FREE DVD

Double-Sided DVD
INSIDE!

FIREFOX
Enabling hardware acceleration

Exploring the iNet Wireless Daemon

LINUX PRO MAGAZINE

ISSUE 243 – FEBRUARY 2021

iNet

Intel's wireless daemon ushers in a new world for Linux networks

Sunflower
New start for a classic twin-panel file manager

Optimizing Vector Graphics
Speed up your websites with these tricks for scrunching SVG files

Extensions in Tiny Core

Escape Room
Build a cool electronic game using an Arduino

MS Visual Studio on Linux
You can even develop code for a Rasp Pi

ExifTool
Analyze file metadata

LINUXVOICE

- Variety: Rotate your favorite wallpaper
- maddog: Weather forecast
- Bpytop: Monitor resources with this bashtop port

FOSSPicks
- Kushview Element
- Artisan Coffee Roaster
- NetSurf

Tutorial
- Ventoy: The easy way to boot from a USB stick

LINUX NEW MEDIA
The Pulse of Open Source

SEARCH DAYLIGHT

Dear Reader,

The big search engines have way too much power over the business world. It almost doesn't matter what your business is. Almost every industry has some kind of online presence – either to sell directly or to provide information to users in need of addresses and product details. If you need acupuncture services, for instance, you will likely type in "acupuncture" with the name of your city to find the services in your area. When the results pop up, the top three names on the list will get most of the clicks, and the first on the list will get the most of all. Getting to the top (or near the top) of that search list is therefore vitally important for any company.

This need to ascend the search ladder has resulted in a system of "search engine optimization" practices that range from the arcane to the comical. We get letters every day from expert "bloggers" offering to post free content on our site if we will link back to their sites. Although they don't mention it, the purpose of this game is to juice up their search rank. Some of these "experts" are actually part of the Open Source community, which might make some sense (if we did that kind of thing), but a few are pretty far afield. One guy said he read all the content on our website, and he knew we would just love his article on interpreting dreams. (Uhhh, we do Linux dude?)

There is a system for how to pump up your search rank, of course, but it has all been derived by something close to trial and error. Search engine optimization is a bit of a black art, because no one but the search vendors knows what the engines are actually doing. A big company can hire experts to ponder and perfect the search rank game. Small companies often get left behind – partly because they lack the expertise, but also, who has time to think about such things with only a few techs running the whole IT operation?

This disparity in time and attention for the mystical game of search optimization is one reason why I applaud the recent guidelines published by the European Commission promoting more transparent search platforms [1]. The guidelines call for the "main parameters" used in search rank algorithms to be made public. Of course, there is much more to the search formula than the main parameters, but the hope is that better knowledge of the parameters will even the playing field, making smaller companies more competitive and reducing the ability of vendors like Google to feature their own products and services at the expense of competitors.

There is some question of what this development will actually mean in the real world. The European Commission has some power for managing and regulating the business environment in the European Union (EU), but the system is quite complex and bureaucratic, and it remains to be seen whether Google and other search vendors can find their way around the need for changes. According to reports, the transparency guidelines are supposedly voluntary, but they represent a roadmap for companies to comply with the transparency requirements of the EU's Platform-to-Business (P2B) regulation, which would prove to be the real enforcement mechanism [2].

But before you pop the champagne corks, keep in mind that this new openness could also have some downsides. In theory, more knowledge of algorithms could lead to more gaming of the system by websites, which could lead to less useful search results, at least until the search vendors adapt and innovate, which they should be doing anyway.

In the long run, it seems better that the Internet, which was built on open standards, should not be under the control of big, weird secret formulas maintained by private, self-interested corporations. The European Commission got this one right. The new guidelines are certainly not a complete solution, but they are a positive first step.

Joe Casad,
Editor in Chief

Info

[1] Ranking Transparency Guidelines: *https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:52020XC1208(01)&from=EN*

[2] EU P2B: *https://ec.europa.eu/digital-single-market/en/platform-business-trading-practices*

LINUX MAGAZINE

WHAT'S INSIDE

With Linux, more innovation is always on the way. This month we take a look at the iNet wireless daemon, a new wireless client that is poised to replace the venerable WPA Supplicant. Also in this issue:

- **ExifTool and jExifToolGUI** – read and manage image metadata from the command line (page 32).
- **Optimizing Vector Graphics** – small but expressive vector graphics files are great for websites, but if you're looking to optimize, get ready for some tweaking (page 50).

Check out MakerSpace for an exotic escape room puzzle built on Arduino, and visit LinuxVoice for a tutorial on Ventoy, an innovative app that lets you build multi-boot USB sticks.

SERVICE

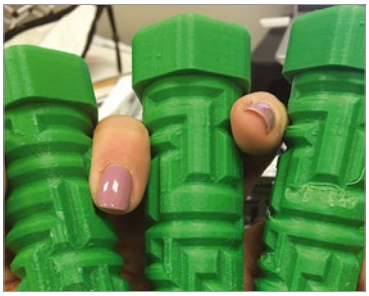
NEWS

REVIEWS

COVER STORIES

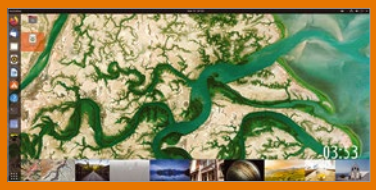
IN-DEPTH

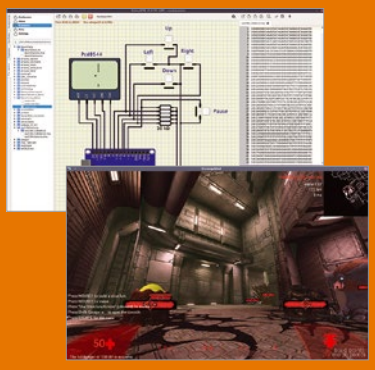
MAKERSPACE

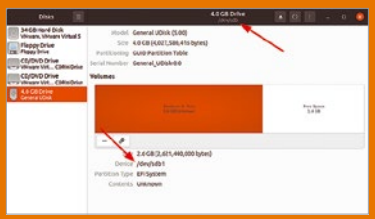


SEE PAGE 6 FOR DETAILS

LINUXVOICE







Linux Mint 20 and Kali Linux 2020.4

Two Terrific Distros on a Double-Sided DVD!

Linux Mint 20
64-bit

Linux Mint became a popular distribution for one overwhelming reason: At a time when Gnome, KDE, and Ubuntu were all trying to innovate on the desktop, Linux Mint listened to what people wanted and gave it to them. The Mint developers forked the popular Gnome 2, keeping it alive under the name of MATE, and, when the Gnome project drifted away from the traditional desktop metaphor with Gnome 3, the Mint team rolled out Cinnamon, an innovative desktop that combines Gnome 3 with a more traditional desktop experience.

Code-named Ulyana, Linux Mint 20 is a Long Term Support (LTS) release based on Ubuntu that will be supported until 2025. Its default installation is a mix of free and proprietary software chosen for ease of use. This month's DVD comes with the version of Linux Mint 20 that boots to the Cinnamon desktop.

Beginner and midrange users should find Ulyana ideal for most purposes. Not only is Ulyana easy to use, but the Mint development team continues to pay close attention to readers' feedback on its forums.

Kali Linux 20.4
64-bit

Unlike many of the distros highlighted on this page, Kali Linux is not primarily intended as a general purpose distribution. Instead, it is designed as a toolkit for forensics and penetration testing. Kali is sponsored by Offensive Security and developed by a thriving community of security experts.

With this intent, Kali's installation is concerned with far more than the tools for a basic desktop environment. The installation includes over 600 tools, ranging from classics like John the Ripper and Wireshark to more recent social- and reverse-engineering tools to Kali's own NetHunter, with new tools added regularly as well as updates. Other features include Kali's own version of the Zsh shell and a makeover of Bash to make it resemble Zsh. In addition, the Kali community and Offensive Security have partnerships with tool authors, providing them with support and forums for their ideas, as well as a broad array of documentation on security issues.

If you decide that it is time to learn about security issues, Kali is one of the best places to start.

Defective discs will be replaced.
Please send an email to subs@linux-magazine.com.

Info

[1] Linux Mint: *https://linuxmint.com/*

[2] Linux Mint Installation Guide: *https://linuxmint-installation-guide.readthedocs.io/en/latest/*

[3] Kali Linux: *https://www.kali.org/*

[4] Kali Documentation: *https://www.kali.org/docs/*

Complete Your Open Source Library with Archive DVDs!

Each fully-searchable archive DVD includes past articles so you can find the content you need quickly.

Save hundreds off the single-copy rate with a convenient archive DVD!

ORDER YOUR DVD NOW!

https://bit.ly/Archive-DVD

NEWS

Updates on technologies, trends, and tools

THIS MONTH'S NEWS

Patreon Project Looks to Bring Linux to Apple Silicon

Developer Hector Martin believes Linux is capable of running on Apple hardware powered by M1 chips. In fact, he believes so much in this goal, that he's created a Patreon project to help fund his efforts.

This is quite a major task for a single developer, but Martin believes he's able to pull it off. The project, however, will require a full-time effort, which is why the developer has created the Patreon page.

Martin's developer experience includes Linux ports of the Nintendo Wii, PlayStation 3, and PlayStation 4.

About this project, Martin says, "Apple just released a new range of ARM-based Apple Silicon Macs that blow every other ARM machine in the same class out of the water. Wouldn't it be nice if they could run Linux too?" As for whether or not the M1-powered Mac hardware can run Linux, Martin makes the claim, "As it turns out, they can, but someone needs to do the work."

As of this moment, Martin has reached his Patreon kick-off goal and the project will start in earnest in January 2021. He plans on documenting his progress and doing the occasional livestreams.

In an interview with ZDNet (*https://www.zdnet.com/article/linus-torvalds-would-like-to-use-an-m1-mac-for-linux-but/*), Linux Torvalds (the creator of Linux) said this of porting Linux to the M1 chip, "The main problem with the M1 for me is the GPU and other devices around it, because that's likely what would hold me off using it because it wouldn't have any Linux support unless Apple opens up... [that] seems unlikely, but hey, you can always hope."

Find out more about Hector Martin's efforts on his official Patreon page (*https://www.patreon.com/marcan*).

A New Chrome OS-Like Ubuntu Remix Is Now Available

If you've used Chrome OS, you know there's a beauty in its simplicity. Over the years, there have been a few attempts at recreating that same simplicity for Linux, but many of those distributions have vanished. From the creator of Ubuntu Unity (*https://ubuntuunity.org/*), comes yet another attempt to create that user-friendly magic. The new distribution, called Ubuntu Web, is based on Ubuntu 20.04 and offers an open source take on Chrome OS.

Photo by Clem Onojeghuo on Unsplash

The developer, Rudy Saraswat, has employed the Gnome desktop (version 3.36, *https://www.gnome.org/*) to pull this off and includes plenty of preinstalled apps to make Ubuntu Web a distribution anyone can use. Included in the app listing you'll find web apps for Mastodon, Twitter, SoundCloud, and a number of others from the /e/ Foundation.

In order to really get the most out of Ubuntu Web, you'll need to have an /e/ account (apply for an invitation at *https://e.foundation/e-email-invite/*). With a free account (which is supported via donations), you gain access to what looks very much like a hosted instance of Nextcloud (which includes an entire suite of tools, similar to that of Google).

Of course, with Ubuntu Web, you can also install any software from the standard repositories, so the distribution isn't limited to web-only.

For more information about Ubuntu Web, check out the channel on the official Ubuntu Community Discourse page (*https://discourse.ubuntu.com/t/ubuntu-web-remix/19394*).

System76 Refreshes the Galago Pro Laptop

System76 (*https://system76.com/laptops/galago*) is known to push the envelope of form and function. But when something works, why reinvent the wheel? That is precisely why the Denver, CO company has given their most popular laptop a bit of a refresh.

The Galago Pro now supports the latest 11th Gen Intel Core i5 and i7 processors and can top out at 64GB of RAM. And although the base model ships with Intel Iris Xe graphics, the laptop can be specified with an optional NVidia GTX 1650 GPU.

Carl Richell, founder and CEO of System76, says of the updates to the laptop, "The Galago Pro has always been a fan favorite of our laptop offerings. The extremely light chassis and well balanced mix of components, all for a very good price, make the Galago an all around excellent computer choice for gamers and engineers alike."

The new take on the Galago Pro will have the same chassis as always (sporting a 14" display), which weighs in at 3.1 pounds and includes an Li-ion 49 Wh battery, 1 × USB 3.2 Gen 2 Type-C/Thunderbolt 4, 1 × USB 3.2 Gen 2 Type-C, 2 × USB 3.2 Gen 1 Type-A, and a microSD card reader.

You can purchase the new Galago Pro, starting at $949 for the base configuration, at System76 (*https://system76.com/laptops/galago*).

Linux Kernel 5.10 is Almost Ready for Release

For a while, Linus Torvalds was concerned about the size of changes for the Linux 5.10 release. However, upon release of the rc6 candidate, that worry has subsided. To this point, Torvalds said, "...at least this week isn't unusually bigger than normal – it's a pretty normal rc6 stat-wise. So unless we have some big surprising left-overs coming up, I think we're in good shape."

Torvalds continued to say, "That vidtv driver shows up very clearly in the patch stats too, but other than that it all looks very normal: mostly driver updates (even ignoring the vidtv ones), with the usual smattering of small fixes elsewhere – architecture code, networking, some filesystem stuff."

As far as what's to be expected in the kernel, there are two issues that have been around for some time that are finally being either given the boot or improved.

The first is the removal of the `set_fs()` feature, which checks whether a copy of the user space actually goes to either the user space or to the kernel. Back in 2010, it was discovered that this feature could be used to overwrite and give permission to arbitrary kernel memory allocations. The bug was fixed, but the feature remained. Since then, however, manufacturers improved the memory management such that on most architecture memory space overloads have been banned.

Image © Igor Negovelov, 123RF.com

Another improvement is the continued work to address the 2038 issue (a bug that has been known for some time regarding time encoding). On POSIX systems, time is calculated based on seconds elapsed since January 1, 1970. As more time passes, the number to represent a date increases. By the year 2038, it is believed 32-bit systems will no longer function.

MORE ONLINE

Linux Magazine
www.linux-magazine.com

ADMIN HPC
http://www.admin-magazine.com/HPC/

Remora – Resource Monitoring for Users
• Jeff Layton
Remora provides per-node and per-job resource utilization data that can be used to understand how an application performs on the system through a combination of profiling and system monitoring.

ADMIN Online
http://www.admin-magazine.com/

DNS Filtering with Authentication
• Dr. Holger Reibold
Filtering HTTP connections and employing traditional proxy servers can protect users from web threats but also increase latency. DNS filters would be a better option, but they have lacked authentication – until NxFilter came along.

Server Administration with Cockpit
• Kristian Kißling and Andreas Reschke
Administer a small server farm, virtual machines, and the Docker alternative Podman with just a web browser.

Tune Your Databases
• Mayank Sharma
Release 2 of Percona Monitoring and Management for open source databases helps reign in inefficient queries and ensure you are making the best use of the available hardware.

As of the 5.6 release, those systems could pass the year 2038. The 5.10 release improves on that reliability.

The new kernel will also see filesystem and storage optimizations and support for even more hardware. It should be released mid-December, 2020.

For more information on the release, check out this message from Linus himself (https://lwn.net/Articles/838514/).

Canonical Launches Curated Container Images

Any admin that has deployed containers understands how important security is for business. The problem with containers is that it's often hard to know if an image is safe to use, especially when you're pulling random images from the likes of Docker Hub. You never know if you're going to pull down an image that contains vulnerabilities or malware.

That's why Canonical has decided to publish the LTS Docker Image Portfolio to Docker Hub. This portfolio comes with up to 10 years of Extended Security Maintenance from Canonical. To that, Mark Lewis, VP Application Services at Canonical has stated, "LTS Images are built on trusted infrastructure, in a secure environment, with guarantees of stable security updates." Lewis continued, "They offer a new level of container provenance and assurance to organizations making the shift to container based operations."

This means that Canonical has joined Docker Hub as a Docker Verified Publisher to ensure that hardened Ubuntu images will be available for software supply chains and multi-cloud development.

For anyone looking to download images, they can be viewed on the official Ubuntu Docker page (*https://hub.docker.com/_/ubuntu*) or pulled with a command like `docker pull ubuntu`.

For more information about this joint venture, check out the official Docker announcement (*https://www.docker.com/blog/canonical-joins-docker-verified-publisher-program/*).

AWS Container Image Library in the Works

In reaction to the new image limiting policy, Amazon is creating what some might call their own internal take on Docker Hub. Why? Because the world's most popular container image repository has started limiting the amount of images users of free accounts (and anonymous users) can pull. Although that number is starting pretty high (5,000 pulls per 6 hours for anonymous and free users) the eventual goal will be limits of 100 image pulls per 6 hours for anonymous users and 200 for free accounts. Although that might sound like quite a large number (even for free accounts), given how complicated some pod deployments can be (with large numbers of container images per deployment), that limit can easily be exceeded, especially when deploying at scale.

That's why Amazon Web Services (AWS) will be offering their own image repository, which will only be available for AWS customers. This new public container registry should hit AWS soon. In their November 2nd announcement (*https://aws.amazon.com/blogs/containers/advice-for-customers-dealing-with-docker-hub-rate-limits-and-a-coming-soon-announcement/*), Amazon intimated the solution would roll out "within weeks."

AWS public images will be geolocated for better availability and offer fast downloads, so you can quickly serve up your applications and services on demand. Amazon will also offer a public website, where anyone can browse the included collection of container images, developer-provided details, and even see pull commands.

Those who pull anonymously will get 50GB of free storage and 500GB of free bandwidth each month and pay nominal charges once that limit has been exceeded.

Make sure to follow the Amazon Containers blog to stay informed (*https://aws.amazon.com/blogs/containers/*).

Zack's Kernel News

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

Debugging Production Systems

In general, a developer might debug the Linux kernel by compiling a bunch of special debugging features, performance analysis features, and whatnot. Then they'll patch any kernel bugs they find. However, the actual users who benefit from those bug fixes would not run the debugging features themselves, because there would be too much overhead. Regular Linux users want fast, low-resource, secure systems that generally kick ass in all directions.

Recently however, Marco Elver of Google submitted a patch implementing Kernel Electric-Fence (KFENCE), a debugging tool that's intended to ship as a default feature of the standard Linux source tree. Why would he turn the world upside down in this way?

KFENCE's goal is to identify memory leaks and boundary violations. Other debugging tools such as KernelAddressSANitizer (KASAN) do this too, but they take a system-wide performance hit in order to find those problems.

KFENCE, on the other hand, sacrifices accuracy in order to achieve virtually no performance hit. However, as Marco explained, given enough time on a large enough set of running systems, KFENCE will eventually accurately detect memory issues. The inaccuracy of any individual test will disappear into statistical background noise as the truth is revealed.

So, why bother? Why not just use tools like KASAN during development and leave debugging tools out of production systems entirely?

Marco explained that "KFENCE will detect bugs in code paths not typically exercised by non-production test workloads."

This is often an issue that frustrates kernel developers. The kernel's process scheduler is a nightmare to maintain, because it has to be developed on a tiny number of systems, but still provide the best possible scheduling performance for hundreds of millions of systems across the galactic quadrant. The amount of pure bile disgorged by developers, one upon another, in debate on that subject alone has hollowed out the nostrils of many a horrified onlooker.

KFENCE takes a different approach. By targeting production systems, it hopes to identify bugs in kernel code that are really and truly along the most-used code paths. Then when those bugs get fixed, they'll be doing the most good for the most people.

There was no debate, but plenty of developers piled in to take a look at Marco's code, point out bugs, and offer suggestions. Jonathan Cameron from Huawei had a few fixes to offer, which Marco said would get into the next version of the patch. Dmitry Vyukov from Google also pointed out some problems with Marco's code. And Alexander Potapenko, on Marco's team at Google, said they'd address these problems in the next version.

SeongJae Park from Amazon was very interested in the KFENCE approach and offered his own evaluation of the patch. Marco took SeongJae's suggestions and said they'd be ready for the version after the next patch release.

Discussion of bugs was soon replaced with suggestions for naming alternatives and code organization preferences, which would seem to indicate that people were generally happy with the direction of the patch.

No one raised any alarm bells, for example, by saying, "we will never allow debugging code in a production release, my friend. Never." Though that doesn't mean no one will raise such objections as the patch gets closer towards acceptance into the main source tree.

For the moment, all seems favorable, at least among the corporate entities

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**.

who in all likelihood would be running the patch on their production systems in millions upon millions of servers housed in secret data centers around the world. So they, at least, seem highly motivated to root out memory issues in the code fast paths upon which they themselves rely. And the rest of us would benefit from those discoveries as well.

Patch Submission Guidelines

Linus Torvalds recently gave some advice to Jakub Kicinski about pull requests. Jakub had sent a pull request for a big pile of bug fixes in the networking tree – deadlocks, authentication, connectivity, the works. There were dozens of patches from a slew of contributors. All of which was fine.

Linus said he used scripts to filter his incoming email, and having the words "git" and "pull" somewhere in the email body (as opposed to the subject line, for example) would make sure he got his eyeballs on it as soon as possible. Without those keywords in the email, he said, the eyeballs would still work, but they might be slower.

Linus also said that for the actual description, the present tense ("Ido fixes failure to add bond interfaces to a bridge") should be replaced with the imperative ("Fix failure to add bond interfaces to a bridge"). The reason, he said, was that these Git log entries would be read by future developers as past events. So if they were written as if they were current at all times, their truth value would likely fluctuate or just fall out of date entirely.

Linus remarked, "Using present tense in particular is very odd when somebody fixed something a year ago and you go back to the description that says 'Ido fixes'. No, he fixed things long ago."

He also explained that for pull requests involving tons of contributors who all deserved credit, "if you want to call out the developer, please do it _ after_ describing the actual fix. Because the commit log (whether for an individual patch or for a merge message) is primarily about what the change is about. Authorship is separate (and generally shows up as such)."

Jakub replied that he'd stick with all those guidelines for next time.

Some of the guidance, as Linus pointed out, was already in the kernel's documentation for submitting patches, though that doc was intended for plain patches rather than pull requests.

New Filesystem for SSD and Flash Drives

Mikulas Patocka announced that he was designing a new filesystem called NVFS. The name apparently derives from the now defunct NovaFS project. NovaFS was intended to mount directory trees on nonvolatile memory, such as flash drives and solid-state drives (SSDs). Mikulas's NVFS project, he said, was smaller and faster than its predecessor; also it had more features and ran on more recent kernels.

Dan Williams was pleased and impressed to see this code. But he did caution Mikulas to avoid some of the pitfalls that had caught earlier projects – not just NovaFS, but ext4fs, the default filesystem in most Linux distributions. Specifically, in Mikulus's original post, he'd suggested that `fsck` ran very slow on NVFS, because the kernel used the buffer cache to map some memory devices. Mikulus felt that the buffer cache simply might be too heavyweight, and that `fsck` could be sped up by 500 percent or 1,000 percent if the kernel would directly map block devices based on Direct Access (DAX), bypassing the buffer cache.

Dan cautioned against this, saying that a whole passel of problems had come up for ext4fs when they tried to bypass the cache and do things directly. Dan acknowledged that those problems all very likely had solutions in some kind of ultimate-truth-of-the-universe sense. However, no one had been able to wend their way to that ultimate truth at the time, and going through the buffer cache solved everything in one fell swoop. Going through that same mess with NVFS, Dan implied, might just lead to the same spiraling misery, and maybe `fsck` didn't really need to run at the absolute highest possible speed after all.

Mikulas was not thoroughly discouraged by this and gave some thought to exactly what might be involved. But he did conclude that "it isn't as easy as it looks." A little while later, he reported that he had "implemented this functionality," although it was a bit of a compromise – essentially, for some files, DAX-based mapping could work, while for others it was necessary to fall back to the standard buffer cache solution. Mikulas was hopeful that he could continue finessing this, gaining more and more speed benefits for `fsck` along the way.

Dan didn't seem optimistic about going down this road, but he offered what help he could. And if Mikulas did end up finding the ultimate DAX-based solution, then all filesystems would likely benefit, so why not try.

Linux supports an uncountable number of filesystems. In fact if you're looking for a fun entry point into kernel hacking, writing your own filesystem could be a good way to go. The number is uncountable because there is such a low barrier to entry. Large numbers of people have whipped up filesystems for their own particular use, using Filesystem in User space (FUSE). This delectable part of the kernel was originally created in response to Hurd's claim that it would be able to offload many core kernel features into user space, including things like networking and filesystems. Hurd developers claimed that monolithic kernels like Linux would never be able to support such things. So some Linux developers got together and did it.

Note that NVFS is not a FUSE-based filesystem. I only mention FUSE here because it's so cool, and because cool filesystems are fun, and because anyone wanting to play around with creating their own filesystems can do it right now without waiting.

Unlocking Memory Access

Thomas Gleixner recently posted a patch to provide a preemptible version of `kmap_atomic()` and similar interfaces. The kernel can't interrupt atomic functions to do things like give CPU time to other processes, but it can interrupt preemptible functions without problems. Preemption is the entire idea behind multitasking. The more preemptible the kernel can become, the smoother and faster the user experience can be, even when there are tons of things running all at once.

The `kmap_atomic()` function is used to access memory that is likely to be re-

mapped soon. You access it, hang on tight while you need it, and only return control to the larger system when you've gotten what you needed. We're generally talking thousandths of a second or faster. But still, more speed is good.

Like the infamous big kernel lock (BKL), `kmap_atomic()` will not go away all at once. There are various implementations, which Thomas's patch seeks to unite under a single umbrella that everyone would use. Then, for cases that can handle the preemptible versions, Thomas will split off separate implementations that avoid atomic locking. In this way, Thomas hopes to whittle away `kmap_atomic()` the same way the BKL was whittled away years ago.

Thomas explained, "This is not a wholesale conversion which makes kmap_atomic magically preemptible because there might be usage sites which rely on the implicit preempt disable. So this needs to be done on a case by case basis and the call sites converted to kmap_temporary." And he also added, "this is only lightly tested on X86 and completely untested on all other architectures." In other words, here's a flame thrower; go play.

Linus Torvalds was more than happy with this. Even aside from atomicity and preemptibility, he liked what Thomas was doing here. He even said the whole kit and kaboodle might be a good replacement for `kmap()` itself, instead of only `kmap_atomic()`.

Instead of Thomas's intended use, Linus suggested that "another solution might be to just use this new preemptible 'local' kmap(), and remove the old global one entirely. Yes, the old global one caches the page table mapping and that sounds really efficient and nice. But it's actually horribly horribly bad, because it means that we need to use locking for them. Your new 'temporary' implementation seems to be fundamentally better locking-wise, and only need preemption disabling as locking (and is equally fast for the non-highmem case)."

Thomas, along with Matthew Wilcox, had some concerns with Linus's idea. As Matthew put it, "people might use kmap() and then pass the address to a different task. So we need to audit the current users of kmap() and convert any that do that into using vmap() instead." To which Linus replied, "Ahh. Yes, I guess they might do that. It sounds strange, but not entirely crazy – I could imagine some 'PIO thread' that does IO to a page that has been set up by somebody else using kmap(). Or similar."

Thomas also replied to Linus's high hopes for a full-on `kmap()` replacement, saying, "I thought about it, but then I figured that kmap pointers can be handed to other contexts from the thread which sets up the mapping because it's 'permanent'. I'm not sure whether that actually happens, so we'd need to audit all kmap() users to be sure. If there is no such use case, then we surely can get of rid of kmap() completely. It's only 300+ instances to stare at and quite some of them are wrapped into other functions."

They went back and forth a bit on the implementation. Though at one point Peter Zijlstra pointed out to Thomas that (as Thomas then explained to Linus), "If a task is migrated to a different CPU then the mapping address will change which will explode in colourful ways." And Linus replied, "Heh. Right you are. Maybe we really *could* call this new kmap functionality something like 'kmap_percpu()' (or maybe 'local' is good enough)."

The discussion continued, with various participants. The clearest thing to come out of it all was that the project had good reason to get much more ambitious, and that the resulting problems would be much more difficult to solve. The whole thing became a deep and dark implementation discussion with many interweaving tendrils, the ultimate goal of which was to eek out that extra little bit of high-speed multiprocessing from every running Linux system.

It's not at all clear where this will all go. At the very least, Thomas's final patch will be something better than what was there before. But ultimately, it's not clear that any of the participants in the discussion are fully clear on what they're trying to do and how it will finally work. This is not unusual in kernel development. Often objections, solutions, and perspective-shifts come from unexpected places in the middle of heated debate – not unlike a merry-go-round whirling at top speed, with developers clinging onto the various bobbing horses, trying to pull each other inward towards the center. ■■■

Exploring Linux's new iNet wireless daemon

Abracadabra

Intel's iNet wireless daemon offers virtually all of the features found in the obsolete WPA Supplicant, and it is smaller by a factor of 10. *By Ferdinand Thommes*

On Linux, a component called WPA Supplicant [1], which has been around since 2003, plays an important role in wireless connections (see the box entitled "Why the Name?"). As the name suggests, WPA Supplicant is a wireless supplicant that supports the WiFi Protected Access Standard (WPA) for secure wireless communication [2]. WPA has been around for over 20 years, and the industry is now on the third major version, which is known as WPA3. WPA Supplicant toils in the background on most modern Linux distros, where users tend to interact with the system through a GUI interface, but if you're using a wireless configuration tool like NetworkManager, Wicd, or ConnMan, WPA Supplicant is probably at work behind the scenes.

WPA Supplicant has seen many improvements through the years, and, in general, it is much easier to connect Linux to a wireless network than it used to be. However, many experts believe that Linux wireless support is due for some reinvention. The world got a scare a few years ago, when WPA Supplicant was shown to be susceptible to the KRACK attack on the WPA2 protocol [3]. Since then, KRACK vulnerabilities have been patched, and WPA3 has taken wireless security to a deeper level, but the complications in implementing a reliable solution underscored the inherent complexity and ungainliness of the WPA Supplicant codebase. That complexity, along with many dependencies, also means that WPA Supplicant is ill-suited for mobile devices and Internet of Things configurations. The need to simplify and provide a better solution for these new technologies explains why efforts have been underway for several years to create a lean alternative to WPA Supplicant.

One alternative that has already arrived, although it still is not installed by default on most Linux systems, is the iNet wireless daemon (iwd) [4]. Intel has been leading the development of iwd for the last four years. In October 2019, the stable 1.0 version was released, and today iwd's version count has reached 1.9. NetworkManager versions from 1.12.0 on can use iwd as their back end. Iwd also works with alternatives such as ConnMan and systemd-networkd. And recently, a small GUI was released for users who want to do without NetworkManager or ConnMan but still want to work through a graphical interface.

The description of the iwd project on *www.kernel.org* highlights simplicity as an important factor behind iwd's recent rise: "The core goal of the project is to optimize resource utilization: storage, runtime memory, and link-time costs. This is accomplished by not depending on any external libraries and utilizing features provided by the Linux Kernel to the maximum extent possible. The result is a self-contained environment that only depends on the Linux Kernel and the runtime C library." [5]

Arch Linux switched to iwd in a snapshot from July 2020. During the installation, you no longer call the `wifi-menu` command to set up WiFi, and the `netctl` network manager has been replaced by `iwctl`. Ubuntu has also been testing iwd and evaluating the possibility of making it the new standard. The developers now consider iwd

Why the Name?

A *supplicant* is one who petitions or asks for something. The 802.11 standards, which provide a vendor-neutral definition for wireless communication, define a role for an *authenticator* (typically a wireless access point) and a *supplicant* (which is the component that asks for the connection – basically, the wireless client).

to be functionally almost on a par with WPA Supplicant. Iwd is likely on its way to your Linux version sometime in the future. In the meantime, we decided to install iwd and take a closer look.

iwd with Ubuntu

We chose a daily build of Ubuntu 20.10 as the test candidate, and we tried to recreate as many of Ubuntu's test requirements as possible. We first used iwd in the terminal, then tested it with the new GUI, and finally used it in combination with NetworkManager, replacing WPA Supplicant as the back end.

Iwd consists of the iwd daemon, the `iwctl` client, and the `iwmon` monitoring tool. The daemon and client were implemented with less than 50,000 lines of code. In comparison: WPA Supplicant weighs in at almost 500,000 lines of code. Iwd uses kernel functions wherever possible (e.g., for encryption). Other benefits include WiFi Protected Setup (WPS) support, simplified network management, fast roaming without unnecessary scanning, and support for multiple profiles per user. For enterprises, iwd also offers support for the following:

- Extensible Authentication Protocol (EAP): A general authentication protocol developed by the Internet Engineering Task Force (IETF) that supports authentication methods such as RADIUS, digital certificates, or SIM cards.
- Trusted Platform Module (TPM): A chip that adds basic security functions to a computer or similar device. In combination with a modified operating system and appropriate software, a Trusted Computing Platform is created.

Iwd basically gets along without configuration because it mainly relies on kernel functions. Only advanced functions like WPA Enterprise require configuration files. Iwd supports WPA3 and Opportunistic Wireless Encryption (OWE). OWE is a standardized procedure for securely encrypting data exchanged on public WiFi networks without a password.

Since version 1.8, iwd has supported peer-to-peer functions via its own API. Bluetooth-style WiFi Direct (WiFi P2P) lets users connect supported devices directly without an intermediate access point.

Listing 1: First Steps

```
01 $ systemctl status iwd.service
02 Unit iwd.service could not be found.
03 $ sudo apt install iwd
04 $ sudo apt purge network-manager
05 $ sudo systemctl stop wpa_supplicant.service
06 $ sudo systemctl disable wpa_supplicant.service
07 $ sudo systemctl mask wpa_supplicant
08 $ sudo systemctl enable iwd.service
09 $ sudo systemctl start iwd.service
10 $ systemctl status iwd.service
```

Getting Started

Before you can get started with iwd, you'll need to take some preliminary steps (Listing 1). First, check if iwd is already installed (line 1). We did not find iwd on the Ubuntu image we tested, which dated from September 12, 2020. We proceeded to install iwd and remove NetworkManager (lines 3 and 4). We

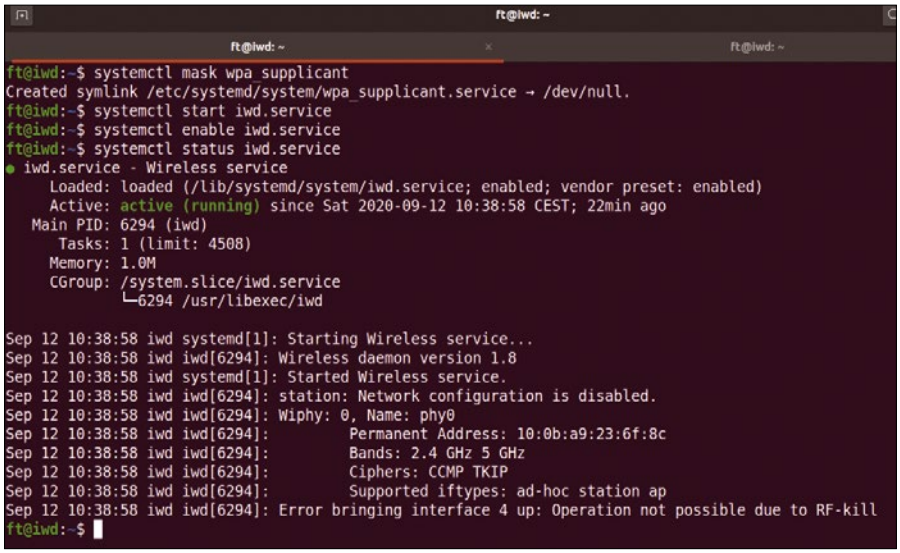


Figure 1: Once WPA Supplicant is shut down, and if iwd always launches at boot time, the status query reports an active service. However, the last line indicates that the device interface cannot be enabled.

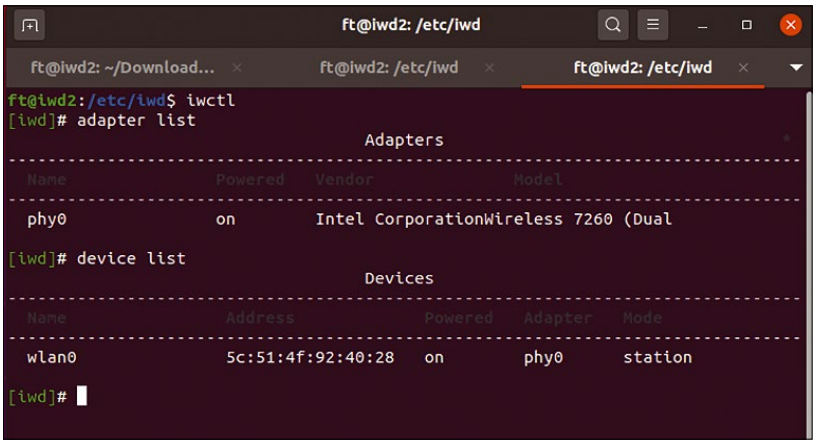


Figure 2: The `adapter list` command displays the available network interface cards with their names and manufacturer IDs.

then disabled WPA and stopped it permanently by masking (lines 5 to 7). Finally, we enabled iwd (lines 8 and 9) and checked if everything was working (line 10).

It is a bad idea to remove the *wpasupplicant* package after the preliminary work is complete, instead of just disabling it. On Ubuntu, removing *wpasupplicant* would also remove the *ubuntu-desktop* metapackage due to many dependencies. On Debian, NetworkManager would be removed as well – which might be a benefit in some cases.

WLAN Setup

Once you have completed the necessary steps, and assuming the status query is positive, you can set up WiFi access. If you get a message about `rfkill` blocking (Figure 1), call the command:

```
sudo rfkill list wifi
```

If *Soft blocked* shows up as *yes*, pressing Fn + F5 might help to switch off flight mode. If this does not help, use:

```
sudo rfkill unblock wifi
```

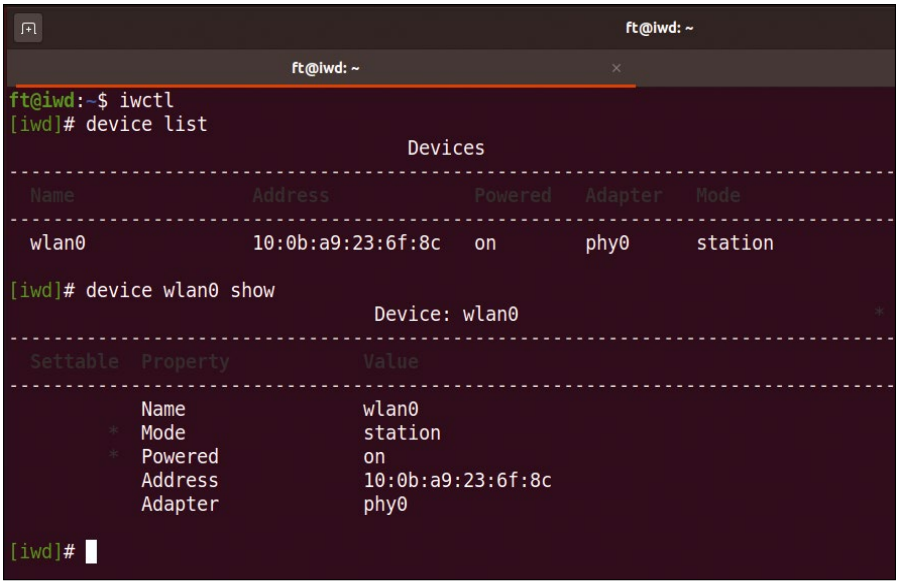


Figure 3: Use the `device list` command to determine the name and state of the interface.

Check if this worked with `rfkill` or a new status request for `iwd.service`.

Now launch an interactive shell as a normal user with the `iwctl` command. Typing `help` lists all the available options. To exit the shell, press Ctrl + D. Iwd can also be used without an interactive shell; you just have to prefix each command with `iwctl`.

Find devices and their names with the `adapter list` command. Use `device list` to discover the name the system is using for the interface (Figure 2). On the test device, the interface goes by the name of `wlan0`. The command

```
device wlan0 show
```

delivers more details about the network interface card (Figure 3). Now scan by typing `station wlan0 scan` before using `station wlan0 get-networks` to display the available networks (Figure 4).

The `station WiFi0 connect your_SSID` command (you need to replace the placeholder with the correct SSID), enables the connection. The requested password is stored in `/var/lib/iwd` when input with the `.psk` suffix.

If needed, check the functionality again by typing:

```
status wlan0 get-networks
```

A check mark, hardly visible against the dark color scheme of the Ubuntu terminal, indicates that the connection was successfully opened. Then use `ping` to check the status of the Internet connection or browse to a website. After rebooting the computer, iwd automatically re-establishes the wireless connection.

DNS Resolution

If the connection fails, or if problems occur when roaming through changing networks, create a `/etc/iwd/main.conf` file with the content from Listing 2 using your preferred editor. The configuration causes iwd to hand over name resolution to `systemd-resolved`. `resolvconf` is also available as an alternative.

As you can see, a computer with iwd can quickly be integrated into a wireless network; graphical tools are not absolutely necessary. Similarly, iwd access can also be set up using WPS, which automatically configures a wireless connection at the touch of a button, PIN entry, or Near Field Communication (NFC) on a device [6]. The `wsc list` command shows whether your device is compatible.

Listing 2: main.conf

```
[General]
EnableNetworkConfiguration=true

[Network]
NameResolvingService=systemd
```

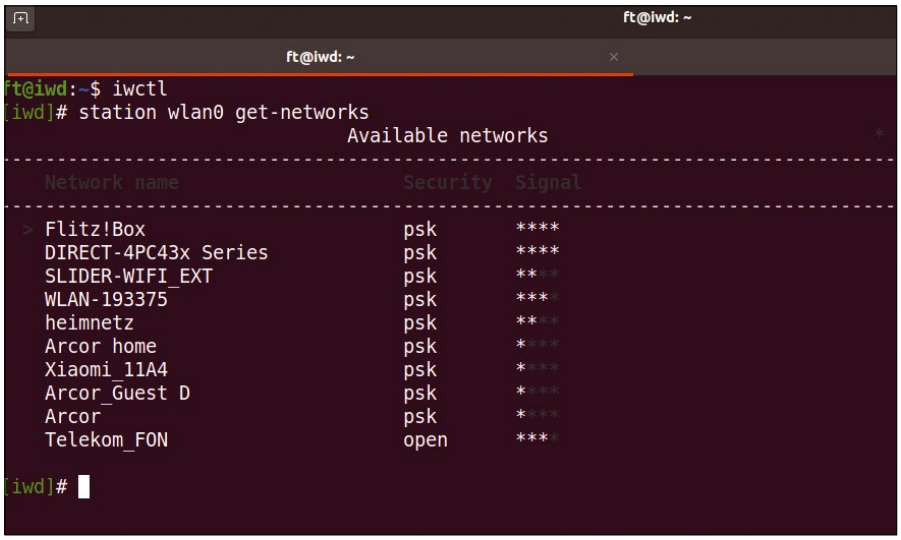


Figure 4: After a scan, `station wlan0 get-networks` displays the available networks.

Table 1: Useful Commands for Iwd

Command	Description
`ad-hoc list`	Lists devices in ad-hoc mode.
`ap list`	Lists nearby access points.
`known-networks-list`	Lists known networks
`known-networks your_SSID forget`	Forget known network.
`station device_name connect-hidden your_SSID`	Connect to hidden network

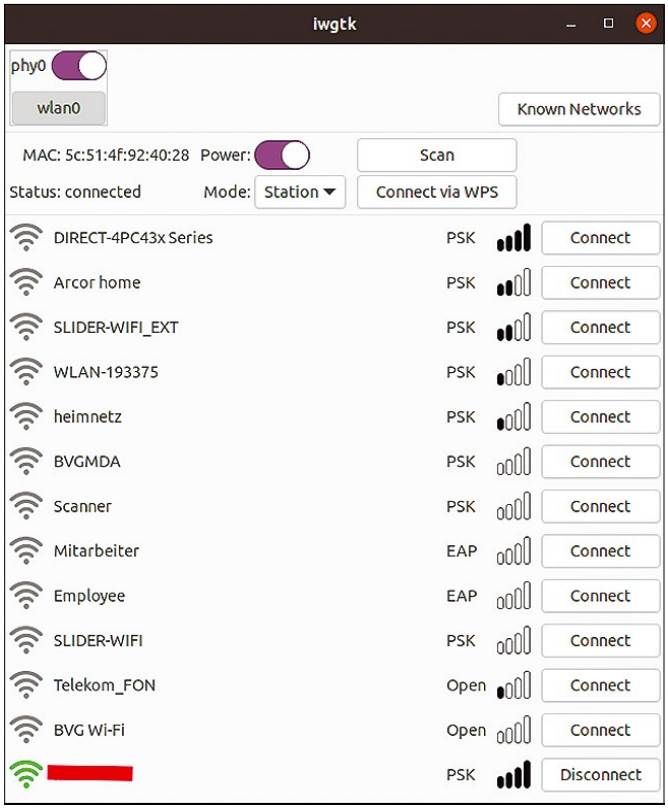


Figure 5: The small iwgtk GUI covers most of the `iwctl` client's command set. However, it cannot and does not seek to compete with the wealth of functions in NetworkManager.

For example, if your router supports WPS, type

```
wsc wlan0 push-button
```

and press the WPS button on the router. If you need a PIN, the command is

```
wsc wlan0 start-user-pin
```

followed by the eight-digit PIN. Other commands supported by `iwctl` are summarized in Table 1.

Small GUI

When you get iwd set up, the next step is to check the iwgtk GUI management tool, which is still very new. Up to now, only Arch Linux (via AUR) and Alpine Linux offer a binary package for iwgtk. However, it doesn't take long to build the package – the only dependencies are iwd and GTK3.

To build iwgtk, download the ZIP file from GitHub [7] while the connection just created with iwd is still active. If you're familiar with Git, you can also clone it. However, Ubuntu doesn't come with basic developer packages such as *gcc* and *make* out the box, and you need to install *libgtk3-dev* and *libglib2.0-dev-bin*.

Go to the unzipped folder, `iwgtk-master/`, and call `make`, `sudo make install`, and `make clean` in that order. After launching the application by typing `iwgtk`, you will immediately see the existing connection. The GUI is already capable of handling the most common use cases via the D-Bus API. For example, if you now terminate the existing connection, you can restore it with `iwgtk`.

The application lets you scan for networks in the vicinity, display known networks, include hidden networks, and connect devices via WPS. The GUI also displays the MAC address. In addition, you can switch from the default mode designated as *Station* to *Access Point* (*AP*) or *Ad-hoc*. The developers recently added an icon for the system section (Figure 5).

You can't use the iwgtk GUI with WPA Enterprise functions, because they are not available via the D-Bus API but instead require configuration in `/var/lib/iwd/`. WPA Enterprise lets the WiFi base station negotiate connection requests from the clients with a downstream RADIUS server. If you need the WPA Enterprise variants, EAP-PWD, EAP-PEAP, or TTLS-PAP, see the article on iwd in the Arch Linux Wiki [8].

Also with NetworkManager

The last step is to test iwd as a back end for NetworkManager. Start by reinstalling NetworkManager. If WPA Supplicant is still active from the previous tests, there is no need for action; if

Listing 3: Editing NetworkManager.conf

```
$ sudo systemctl stop NetworkManager.service
$ sudo nano /etc/NetworkManager/NetworkManager.conf
```

not, repeat the appropriate steps from Listing 1. Then edit the NetworkManager configuration (Listing 3).

You need to add a new `wifi.backend=iwd` line to the `[device]` section. Then relaunch NetworkManager and reboot your computer; you will have to enter the WiFi password. You have now successfully replaced WPA Supplicant with iwd, which means you can look forward to faster establishment of WiFi connections, and you can also switch to WPA3 Personal.

Commissioning the TP-Link AC750 router as an access point on a Fritz!Box worked in all three tested scenarios. You need root privileges for monitoring with the third component, `iwmon`. The command outputs a log that is more suitable for developers. Typing

```
iwmon -w log.pcap
```

redirects the log to a file, which you can load into Wireshark for analysis and evaluation (Figure 6). The `.pcap` suffix stands for Packet Capture, an interface for recording network traffic.

Conclusions

Iwd 1.9 has been running smoothly on one of our laptops as a back end for NetworkManager for about three months now and is not even fazed by a VPN. Even when testing with a development version of Ubuntu 20.10, no problems occurred.

Iwd is the perfect replacement for the ancient WPA Supplicant in home and small business applications, where it meets advanced requirements for access points and hidden networks. ■■■

Info

[1] WPA Supplicant: *https://en.wikipedia.org/wiki/Wpa_supplicant*

[2] WPA: *https://en.wikipedia.org/wiki/Wi-Fi_Protected_Access*

[3] KRACK: *https://en.wikipedia.org/wiki/KRACK*

[4] iwd: *https://git.kernel.org/pub/scm/network/wireless/iwd.git*

[5] Kernel.org iwd page: *https://iwd.wiki.kernel.org/*

[6] WPS: *https://en.wikipedia.org/wiki/Wi-Fi_Protected_Setup*

[7] iwgtk on GitHub: *https://github.com/J-Lentz/iwgtk*

[8] iwd on the Arch Wiki: *https://wiki.archlinux.org/index.php/Iwd*

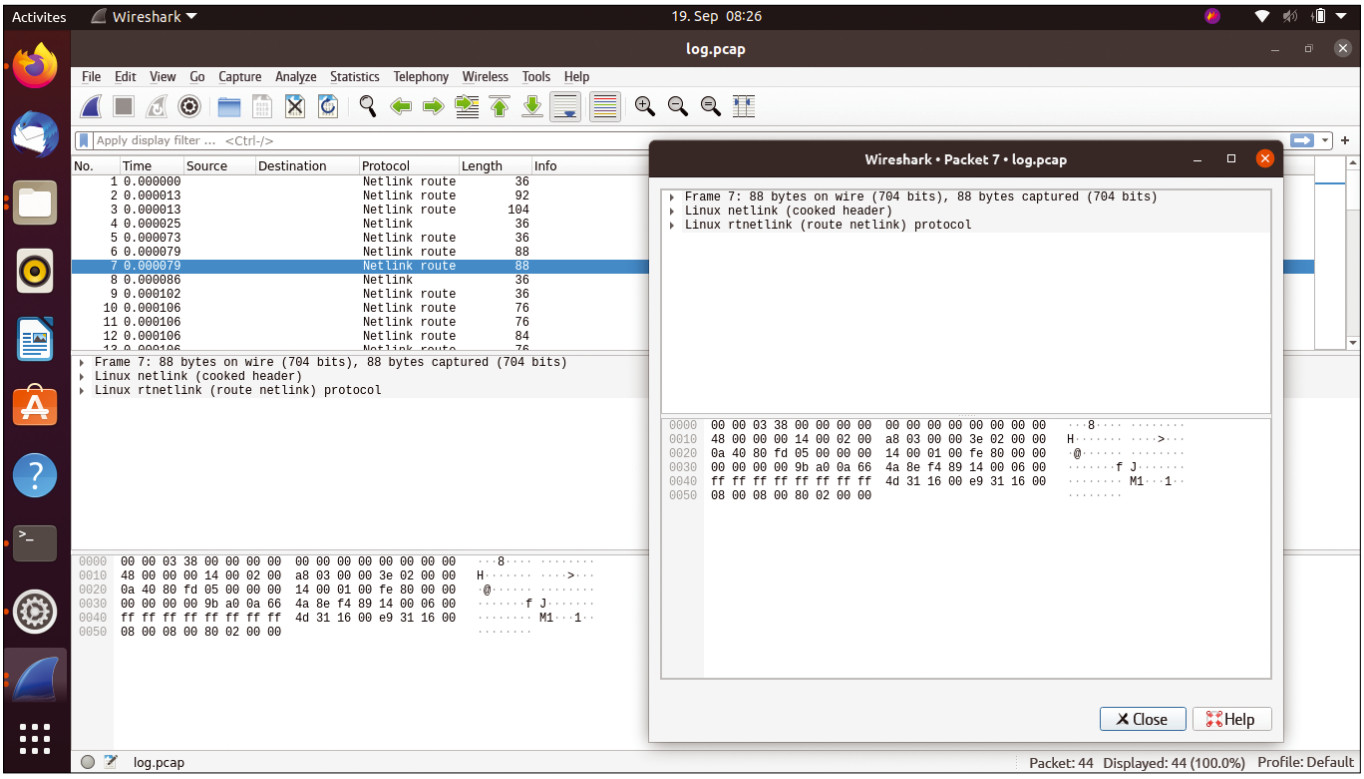


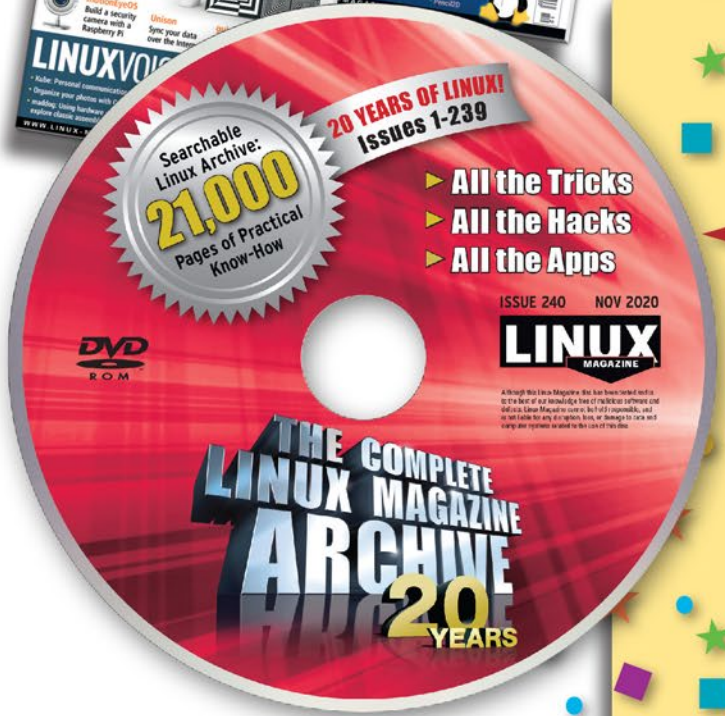
Figure 6: The `iwmon` analysis tool lets you redirect output into a PCAP file, which you can then evaluate using Wireshark.

Help us celebrate 20 years of *Linux Magazine!*

Linux Magazine is your guide to the world of Linux:

- In-depth articles on trending topics
- How-tos and tutorials on useful tools
- 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

Subscribe now and get the Linux Magazine 20th Anniversary Archive FREE with your first issue!



20 YEARS LINUX MAGAZINE

Order today:

https://bit.ly/20th-Anniversary

State of the distros

Trend Watcher

Bruce takes a look at DistroWatch data for a glimpse into the current state of Linux distributions. *By Bruce Byfield*

Most of us experience Linux through the filter of a distribution. The exceptions are those who build their operating system from source files – there is even a project called Linux From Scratch to help that hardy minority. However, like most free software, distributions are hard to track. Many offer download statistics, but often one user does multiple downloads. The only data available is the statistics offered since 2001 by Ladislav Bodnar's DistroWatch site [1]. It is not a precision instrument, since distributions can take some time to appear on the site for one reason or the other, but it is the best we have. Accordingly, every year or so, I like to scan the site to see what trends I can tease from its pages. I mainly see a growing centralization over the last decade, although a few minor trends are also visible.

DistroWatch lists 418 distributions. Of these, 275 are active in 2020, a small decline since 2014, when 285 were active, and a larger decline from the 323 in 2011. Today, 55 are listed as dormant, which means that someone somewhere might consider reviving them at some point, while 88 are officially discontinued. Overall, just under two-thirds of all distributions ever released continue to update. Of those, only 15 are BSD, while 4 are Solaris. The rest, of course, are Linux.

The reason for this decline is probably the dominance of a handful of large distributions, particularly Debian. The

Last 12 months			Last 6 months			Last 3 months			Last 1 month		
1	MX Linux	3991	1	MX Linux	3480	1	MX Linux	3398	1	MX Linux	3670
2	Manjaro	2708	2	Manjaro	2314	2	Manjaro	2354	2	Pop! OS	2338
3	Mint	2373	3	Mint	2131	3	Pop! OS	2145	3	Manjaro	2261
4	Ubuntu	1618	4	Pop! OS	1763	4	Mint	1928	4	Mint	2001
5	Debian	1437	5	Ubuntu	1390	5	Ubuntu	1408	5	Ubuntu	1300
6	elementary	1365	6	Debian	1233	6	Debian	1210	6	Debian	1178
7	Pop! OS	1361	7	elementary	1142	7	elementary	1110	7	elementary	1110
8	Solus	1085	8	Fedora	972	8	Fedora	1066	8	EndeavourOS	1014
9	Fedora	1029	9	Solus	880	9	EndeavourOS	980	9	Fedora	975
10	Zorin	934	10	EndeavourOS	846	10	Solus	789	10	Solus	766

Figure 1: Page hits on DistroWatch are not an exact metric, but they do suggest which distros interest users.

Photo by Tyler Nix on Unsplash

dominance is clearly seen in the list of page hits each distro receives on DistroWatch. These hits only represent popularity among DistroWatch's readers, but the top 10 results seem likely to approximate the popularity among all Linux users (Figure 1). In the last decade, Ubuntu, Fedora, Mint, and Debian have been consistently in the top 10, the only difference being the exact order. Debian consistently hovers around fifth, and Mint somewhere between first and third, but others vary. A decade ago, Ubuntu was consistently first while it was focusing on its desktop. However in 2020, with its move away from desktop development, Ubuntu has slipped to fourth. One thing has been consistent in the last decade: At least 4 of the top 10 distros have been Debian, and sometimes as many as 6. Debian's dominance is even stronger in page hits, with 11,139 for Debian and its derivatives, and only 5,012 for other distributions.

Meanwhile, openSUSE currently has slid to 12th place, while Manjaro has entered the top 10 in the past few years. More recently, Pop!_OS and elementary OS (two Debian derivatives) have also entered the top 10. Farther down the top 100, once-popular expert distros like Slackware, Gentoo, and Arch have slid down the list. By contrast, aesthetic-focused distros like dee-

OS Type	All
Distribution category	All
Country of origin	China
Based on	All
Not based on	None
Desktop interface	All
Architecture	All
Package management	All
Release model (LTS defined)	All
Install media size	All
Install method	All
Multi-language support	All
Init software	All
Status (defined)	Active

Submit Query

Figure 2: DistroWatch's Search page not only helps users decide between distros, but it also indicates some of the current trends.

pin and Zorin have also slid from the top 10, but hover just outside it.

Plying the Search Field

The link to DistroWatch's Search page [2] is hidden in its homepage header. It is intended to help users choose a distribution and includes the ability to track the releases of packages in distributions. However, more relevantly, it also gives a rough indication of preferences and technological trends (Figure 2).

After the list of page hits, a place to begin looking at on the Search page is the *Based on* field. Right away, the field gives proof of Debian's importance by allowing searches based on the Stable, Testing, or Unstable repositories, a feature not given for any other distribution. A search for distributions based on Debian returns 121 results, with 8 based on Testing and 1 on Unstable. While Linux users are sometimes assumed to care most about the latest releases, these numbers suggest that for many users the priorities are what Debian Stable is best known for: stability and secure and timely updates.

A search for distros based on Ubuntu, Debian's most popular derivative, returns another 55 results. Derivatives of Mint, Knoppix, and other derivatives might add another 10 or so. In other words, just under two-thirds of active distributions are first- to third-generation Debian derivatives. No other distribution comes even close to this percentage. Even Fedora, which two decades ago rivaled Debian in popularity, has only 26 derivatives, and then only if CentOS and Red Hat Enterprise Linux's derivatives are counted in Fedora's tally. Other derivatives tend to be based on expert distros, like Arch (21), Gentoo (8), or Slackware (7), often simplifying installation and configuration. Although the market for these derivatives may be small, there is still enough interest to continue their existence. In comparison, only 61 distributions are listed as independent (i.e., not a derivative).

Social and Hardware Trends

According to DistroWatch, distributions are developed in 74 countries. However, they remain primarily a phenomena of the United States (76) and Western Europe (80). However, given online development, the country of origin is not as important as it might be in conventional development. The bias is also lessened by 72 different locales and languages.

The site lists 30 different distribution categories. Of these categories, 158 distributions are listed under *Desktop*, which seems to mean general purpose, followed by 64 for *Server*, and 23 for *Security*. Other well-represented categories are *Old Computers*, *Beginners*, and *Education*, each of which is represented by 14 distros. Continued interest in Raspberry Pi is shown by 30 distros that support it, as opposed to a single distro for Android.

These listings also suggest several trends in development. For instance, 234 current distributions have fixed releases, in which the entire distribution is updated in a general release, whereas 48 have rolling releases, in which packages are updated individually rather than altogether. Some of these rolling releases, like Arch Linux, are entirely rolling, while others, like openSUSE, issue both fixed and rolling releases. Since fixed releases are potentially more secure, while rolling releases are more current, which one your distro supports can be a major concern. Similarly, 15 distributions support Long Term Support (LTS) releases, which are supported for longer than standard releases, making them of interest to corporate users.

Another issue that can be tracked on DistroWatch is support for UEFI Secure Boot [3], a verification system to ensure that only secure code runs a system; this has been a growing concern for several years. Although it does not affect older computers, Secure Boot is oriented towards Microsoft operating systems. When first introduced, it had to be disabled – if possible – to run Linux. However, a growing number of distributions use shim, which (as its name suggests) stands between other operating systems and Secure Boot. Instead of requiring Secure Boot to approve a variety of operating systems, it need only approve shim. DistroWatch includes a search for which distros install shim and which version they support. The list includes not only major distributions like Ubuntu and Fedora, but a growing number of lesser-known ones as well – a total of 46 altogether, with this number slowly increasing as a former hardware issue becomes simply another consideration.

Perhaps the most important issue that can be tracked is the use of systemd [4], the elaborate init system for starting and managing systems. Ask around, and most people who use Linux are apt to assume that systemd has more or less become standard in distros, making the controversy over whether to use it a dead issue. However, DistroWatch's search features soon suggest otherwise. Interest in alternate init systems is strong enough that during the last year or so DistroWatch has had a special search for them – a search that returns 102 distributions. Many of these distributions are minor, or, like Devuan, consciously in revolt against systemd, but the results also include popular distributions like MX Linux, Knoppix, and PCLinuxOS. You can also search based on particular init systems. Apparently, the debate over systemd is not over yet.

Larger Trends

Free software's values include diversity and choice. However, while still healthy, these values appear to be in decline over the last decade, with the number of distros declining by 15 percent. Even more significantly, mainstream Linux has become increasingly synonymous with Debian. Although a crisis is still distant, it appears that free software has less to congratulate itself on than most of us assume.

Whether this loss of diversity will continue is anybody's guess. Some of the development trends, like the increased interest in non-systemd inits, suggest that diversity can re-emerge even after an issue appears to be resolved. But there is still enough diversity that little is certain. The information DistroWatch collates to help users choose their distros is not an exact mirror of trends, although it suggests more than might be expected. While imperfect, DistroWatch offers one of our few glimpses into the current state of distributions. ■■■

Info

[1] DistroWatch: *https://DistroWatch.com*

[2] DistroWatch Search: *https://DistroWatch.com/search.php*

[3] Secure Boot: *https://en.wikipedia.org/wiki/Unified_Extensible_Firmware_Interface#Secure_boot*

[4] systemd: *https://en.wikipedia.org/wiki/Systemd*

A twin-panel file manager based on Python 3/GTK3

Double Flower

Following in the footsteps of Norton Commander and its clones, the Sunflower twin-panel file manager takes a giant leap forward with a switch to Python 3 and GTK3. *By Christoph Langner*

Some programs, such as Netscape, Adobe Photoshop, and Microsoft Office, set the benchmark for future products. For file managers, the discontinued Norton Commander paved the way to easier copying, moving, and editing files at the MS-DOS command line. Norton Commander's view, which was divided into two halves (hence the umbrella term "twin-pane file manager") let users select files from a directory list in one pane by pressing Insert and then transfer them to the directory opened in the other pane by pressing F8.

Norton Commander quickly spawned numerous clones, including Total Commander [1], Midnight Commander [2], Gnome Commander [3], PCManFM [4], and Krusader [5], as well as several other candidates. One of the classics, the graphical Sunflower [6] file manager, has been under continuous development for many years. With version 0.4, the project has switched to Python 3 and GTK3, a radical change that cost the project a great deal of time, which explains the four years between versions. Sunflower 0.4 offers an easy-to-use, highly customizable twin-pane file manager that integrates seamlessly with (but is not limited to) the Gnome desktop.

Installation

Sunflower is not available in the package sources of most popular distributions. But Arch Linux users will find two packages, *sunflower* and *sunflower-git*, in the Arch User Repository (AUR) organized by the community. With an AUR helper like Yay, you can install the program by typing:

```
yay -S sunflower
```

For other distributions, the Sunflower project offers deb and RPM installation packages, including ones for Debian, Ubuntu, and Fedora [7]. In testing, the installation worked without problems on Ubuntu 20.04 LTS and Fedora 32.

After launching, Sunflower offers a typical twin-pane file manager window (Figure 1). On both sides, you will find your home directory's contents. With a mouse click, you can navigate through the folder structure, or you can open files in the associated applications by double clicking. To launch a selected file with a different program, right-click to open the context menu and select *Open with*. In the menu, you will also find the usual functions for creating folders or files, functions for file operations, and options for changing the file permissions or associations (via *Properties*).

Operations

In the file list, you can use the mouse or the keyboard to navigate. Use the Up and Down Arrow keys to select an entry. In the usual style for Commander clones, you can reverse the selection of a file or folder by pressing Insert. Alternatively, hold down the Shift key and use the Up and Down Arrow keys to highlight a selection.

The Right Arrow key expands the folders' contents in a tree view, while the Left Arrow key collapses the tree. Clicking on *Mounts* takes you directly to your

Photo by marty brixen on unsplash

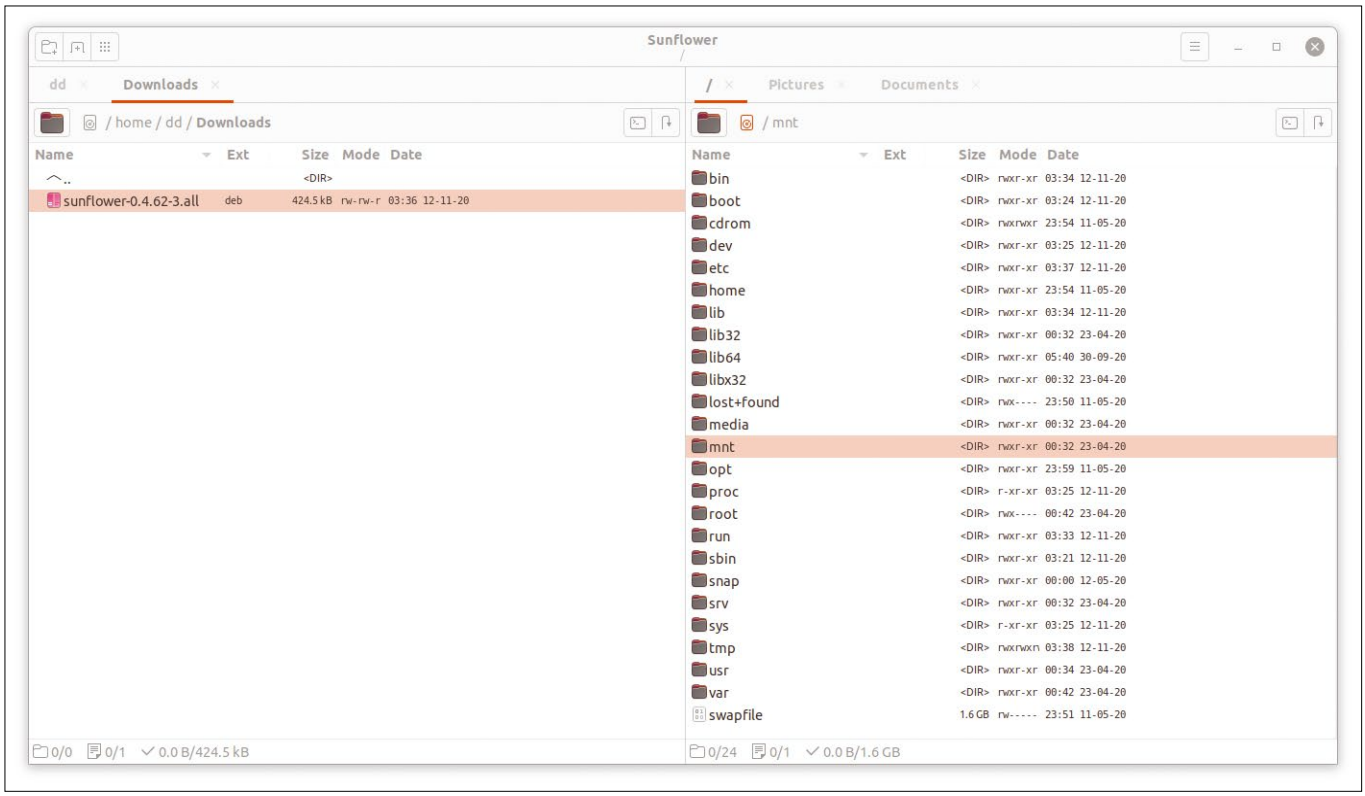


Figure 1: Sunflower 0.4 uses modern components like Python 3 and GTK3, while retaining the classic structure of two panels, tabs, and a toolbar at the bottom.

home directory or to mounted external drives like USB memory sticks or Android smartphones.

Choosing *Selection* from the hamburger menu in the window's upper right corner gives you further possibilities for selecting files or folders, including selecting or deselecting files according to a pattern or their file extension. *Compare Directories* in the same menu is also practical if you want to synchronize a backup directory. This option selects the files that do not exist in the other opened subwindow, but it ignores files that have just changed.

Use Ctrl + Right Arrow to expand the selected folder on the left in the right pane, while Ctrl + Left Arrow collapses the selected folder. Press Ctrl + T to open the current folder in a new tab, while Ctrl + Shift + T opens the selected folder in a new tab (see Table 1 for additional shortcuts).

The *Display* menu lets you customize the application window. The *Horizontal Split* option divides the file manager into two areas arranged one above the other. *Dark theme* lets you enable your desktop

Table 1: Sunflower Shortcuts

Abbreviation	Function
File operations	
F2	Rename selection
F3	View selected element
F4	Edit selected element
F5	Copy selection
F6	Move selection
F7	Create new directory
Del or F8	Move selection to trash
Actions in file list	
Insert	Invert selection
Shift+Up Arrow	Select top element
Shift+Down Arrow	Select bottom element
Ctrl+Right Arrow	Expand selected folder
Ctrl+Left Arrow	Collapse selected folder
Tabs	
Ctrl+T	Open current folder in new tab
Ctrl+Shift+T	Open selected folder in new tab
Ctrl+Tab	Move to the next tab
Ctrl+Shift+Tab	Switch to previous tab
Ctrl+W	Close current tab
Terminal	
Ctrl+Z	Open current folder in terminal
Ctrl+Shift+C	Copy selection in terminal to clipboard
Ctrl+Shift+V	Paste content from clipboard to terminal

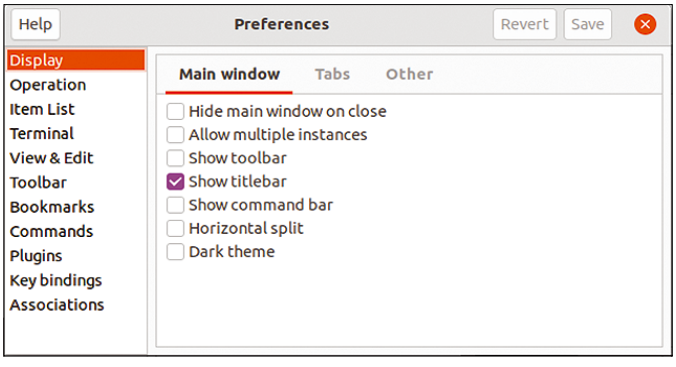


Figure 2: Sunflower can be extensively customized in the settings: A mouse click lets you arrange the panels horizontally or activate a dark theme.

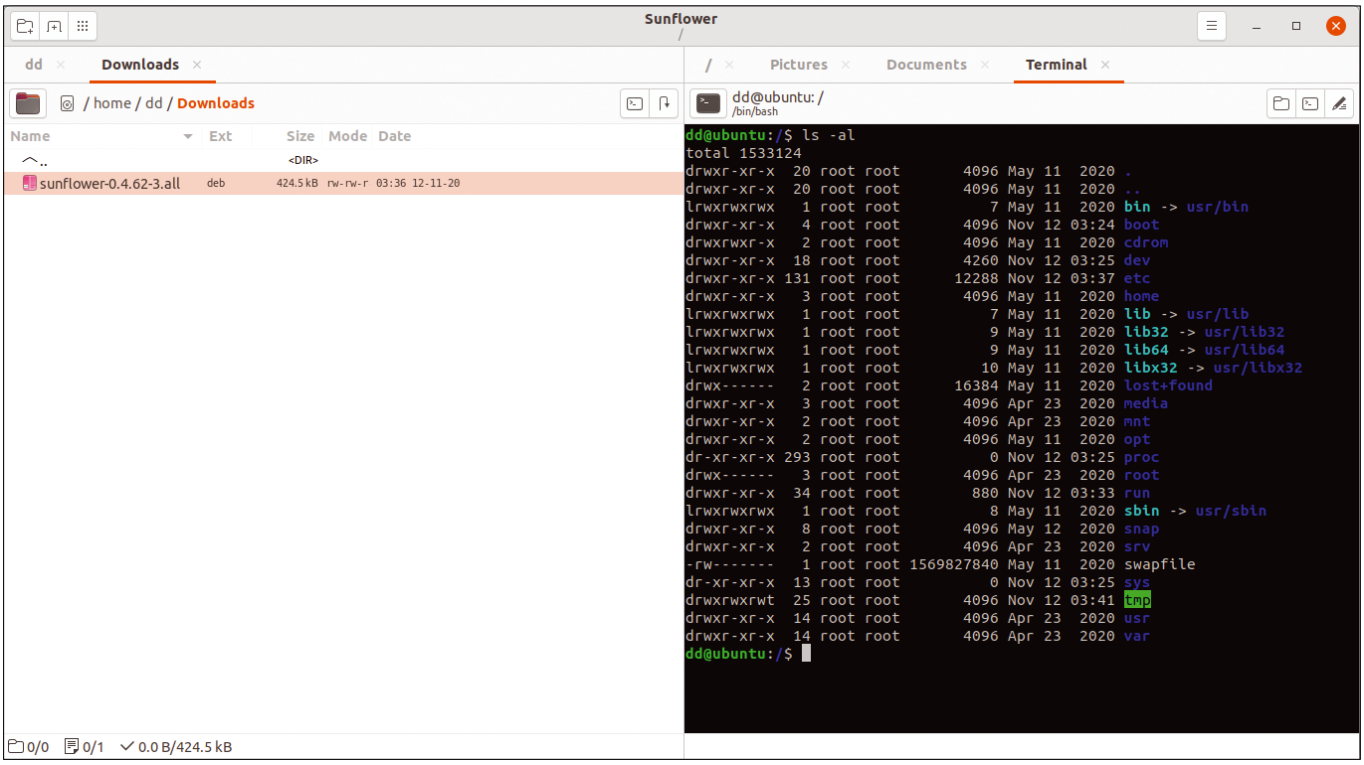


Figure 3: Use Ctrl+Z to display a terminal window in the currently active file manager pane.

environment's default theme for multimedia applications (Figure 2).

One of Sunflower's strengths lies in the terminal integrated directly in the interface. You can access it by clicking on the terminal icon in the buttonbar above the two panes or with the keyboard shortcut Ctrl + Z (Figure 3). By default, Sunflower uses the VTE terminal library, upon which other terminal applications, such as the Gnome terminal or Terminator, are also based. To adjust the font size and other details, go to *Preferences* | *Terminal* in the hamburger menu. Optionally, Sunflower will also launch an external terminal program, but it does not offer a context menu. For example, to insert path-

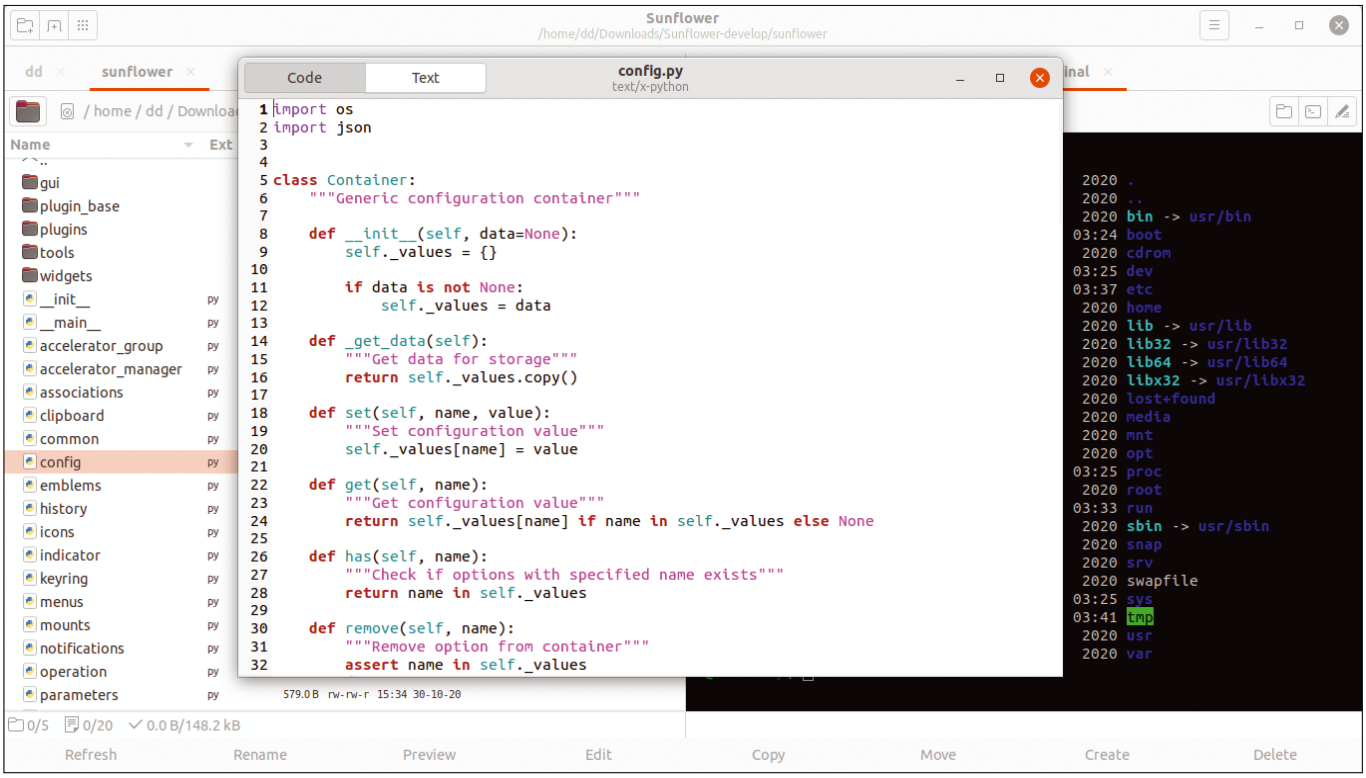


Figure 4: Sunflower opens files with text or source code in a preview. The preview function automatically highlights variables or commands.

names, you have to click on the terminal icon and select *Insert*.

Sunflower offers an integrated preview function for text and source code files (Figure 4). To view a file, select the file in question and click on *Preview* in the lower menubar or press F3. The preview automatically highlights variables and instructions in the source code. You can also disable this highlighting by clicking on *Text* in the preview window's header.

To modify a file, select *Edit* or press F4. Sunflower then opens the selected file in the desktop environment's standard editor. You can optionally choose a different editor in the settings in *View & Edit* or enter an external command.

Conclusions and Outlook

The switch to Python 3 and GTK3 sets Sunflower on course for the future. With version 0.4, it is now far easier to develop the application and expand the range of functions. In testing version 0.4, I found a few areas still under construction. For example, selecting plugins for finding or renaming files results in an error message that tells you to enable a plugin – but a plugin with the stated name does not exist.

Sunflower still needs to do a bit of catching up in terms of network support. While other file managers, such as Gnome Files (formerly Nautilus) or KDE's Dolphin mount network shares directly via a URL such as `smb://<User>@<Server>` or `ssh://<User>@<Server>`, Sunflower completely lacks this function. Network drives can only be accessed if they have previously been permanently mounted in the folder structure or integrated into Nautilus. Afterwards, the contents can be viewed via the corresponding mount point or, in the case of Gnome's GVfs virtual filesystem, in `/run/user/<UID>/gvfs`. ■■■

Info

[1] Total Commander: *https://www.ghisler.com*

[2] Midnight Commander: *https://midnight-commander.org*

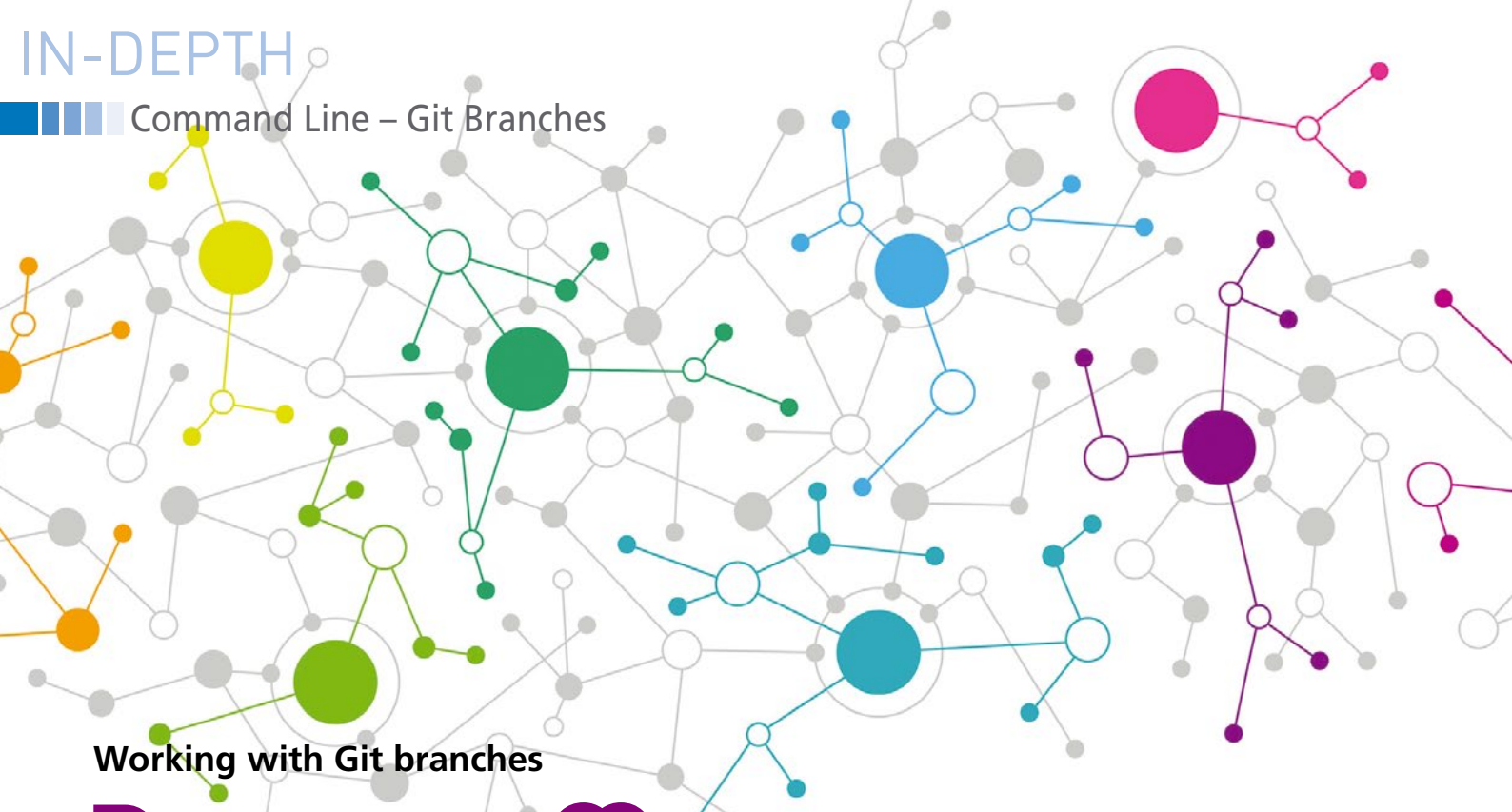
[3] Gnome Commander: *https://gcmd.github.io*

[4] PCManFM: *pcmanfm.sourceforge.net*

[5] Krusader: *https://krusader.org*

[6] Sunflower: *https://sunflower-fm.org*

[7] Download packages: *https://sunflower-fm.org/download*



Working with Git branches

Branch Management

Git makes version control as simple as possible. To manage your Git branches successfully, you need to learn the ins and outs of git branch and git merge. *By Bruce Byfield*

In a previous issue of *Linux Magazine* [1], I outlined how to set up a Git Repository (Table 1). Once a repo is set up, you are ready to use Git [2] for version control.

Version control in Git is about managing and creating branches (collections of files that are variations of each other). By making a branch, you can edit files without altering the original. Later, if you choose, you can merge branches to create a more advanced copy.

The process of managing branches is so simple that branches tend to proliferate in projects that use Git, with branches being created for different users, subsystems or hardware architectures, each release, experiments, or any other criteria that may useful. In fact, while other version control systems also use branches, using Git repos is so flexible that it is widely considered Git's defining feature. Two Git sub-commands are the main tools for managing branches: `git branch` [3], which sets up and edits branches, and `git merge` [4], which combines branches.

A Basic Overview

When a repository is created, a default called `master` is created. It is the top-level branch, analogous to the root directory on a filesystem, with its commits the equivalent of the root directory's files. Otherwise, `master` is no different from any other branch. However, ordinarily, the commits in `master` are not edited – the whole point, after all, is to have a clean set of files in case a recovery is needed. The most common workflow is to create a branch below `master` and edit the files there.

Git manages the branch tree with a series of pointers. The current branch is kept track of with a pointer called `HEAD`. You create a new branch locally with the command:

```
git branch NAME
```

If you want to move to the new branch immediately after creating it, instead use

```
git checkout -b NAME
```

or

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/*.

Table 1: Basic Commands for Creating a Repository

`git clone URL`	Copy remote repository to your local hard drive.
`git pull URL`	Sync a local repository with the remote original.
`git init`	Create a local repository in the current directory.
`git add FILE1 FILE2`	Stage (prepare) files in the repository.
`git commit -m "MESSAGE"`	Add staged files to the repository.

Note: Each of these commands has other options, as well as its own man page. These are only the basic commands.

```
git switch -c
```

Otherwise, use `git checkout NAME`, a basic navigational command for a repo. A new alternative for navigation is `git switch NAME`, which has the advantage of allowing you to return to your previous repo with `git switch`.

To transfer a local branch to a new remote repository, the remote repository must first be recognized by the local one with:

```
git remote add ↵
  REMOTE-REPO-URL://LOCAL-HOST/USER/↵
  repo.git
```

Then a local branch can be uploaded remotely with:

```
git push REMOTE-REPO-URL LOCAL-BRANCH
```

The new branch is automatically created as a sub-branch of the current one. At the same time that you move to the new branch, so does `HEAD` (the current branch pointer). Keep in mind, too, that when you change branches, you change the versions of the files available to you. Should you forget, your Git repository can abruptly become a puzzling place.

Especially with large projects or numerous contributors, Git repo trees can have considerable depth and breadth. In such cases, navigating – just knowing where you are – can be a challenge. One solution is to use the command `git branch --list` (Figure 1), which marks the current local repository with an asterisk and prints any checked out local repositories in cyan. Use `-r` with `--list`, and remote repositories are listed, while `-a` lists both local and remote branches. Another navigational tool is to install the *git-big-picture* utility [5], which opens a map of the repository in a separate window (Figure 2).

Especially when merging (see below), you may also want to use `git tag` to give each commit a human-friendly name and refer to Git's log. The log has an effective search function, and you can filter the branches with:

```
git log INCLUDED-BRANCH..EXCLUDED-BRANCH
```

Editing Branches

A commit's actual content is edited elsewhere – generally in a text editor. However, branches can also be edited and managed using the command:

```
git branch OPTIONS STRING
```

In this structure, `STRING` can be a branch name or a commit, depending on the context. For instance, if a regular expression is used by itself, rather than a single branch name, then you could possibly create other branches – which could cause a lot of unwanted branches. To guarantee that you simply filter with a regex, `--list` must be used as well. Another way to filter is to use `--contains` to include only branches that contain the named commit (i.e., branches that are descendants of the commit) or `--no-contains` to exclude descendant branches. Similarly, `--merged` lists only branches merged into the named commit, while `--no-merged` excludes merged branches.

One of the more common ways to edit branches is with deletion. The simplest choice is `--delete` (`-d`), but it only works if the branch is merged in its upstream branch, in `HEAD`, or has a branch marked as its upstream (see below). If you are sure you want to remove the branch, use `-D` instead, or combine `--delete` with `--force` (`-f`).

Branches can also be copied with `--copied` (`-c`) or moved with `--move` (`-m`). The `--move` option has the same restrictions as `--delete`, but it can be forced with `-M` or by being accompanied by `--force`. If you want to rename a branch, you can use `-m` (`-M`) `NEW-BRANCH OLD-BRANCH`.

```
bb@nanday:~/wip$ git branch --list
  1st
  2nd
  Final
  checkout
  master
  new-characters
  outtakes
* sub-plot
```

Figure 1: A color-coded list of branches.

A more complicated edit is to add `--track` when a new branch is created. This option marks the current branch as being upstream from the new branch and is the default when working from a remote branch. In effect, it is a navigational marker, shown in `git status` and `git branch -v`. After a branch is created, you can give it an upstream with `--set-upstream-to BRANCH` (`-u BRANCH`) or remove an upstream with `--unset-upstream`.

More trivially, you can set how output is displayed. `--color=WHEN` color-codes current, local, and remote repos, defaulting to `always` but also taking `never` or `auto` as a value. To ignore whether characters are lowercase or uppercase, use `--ignore case`. Another option is to display output with `--column` or to use the verbose modes (`-v`, `--verbose`, or `-verbose`) to see more information – although at the cost of having no columns.

Merging Branches

Merging is the joining of branches and the resolving of any differences between them. Before you merge, use `git fetch` to make sure that any local or remote repos are in sync, and check that all changes are committed. Then check out the branch into which you want to merge and run `git pull` to make sure it

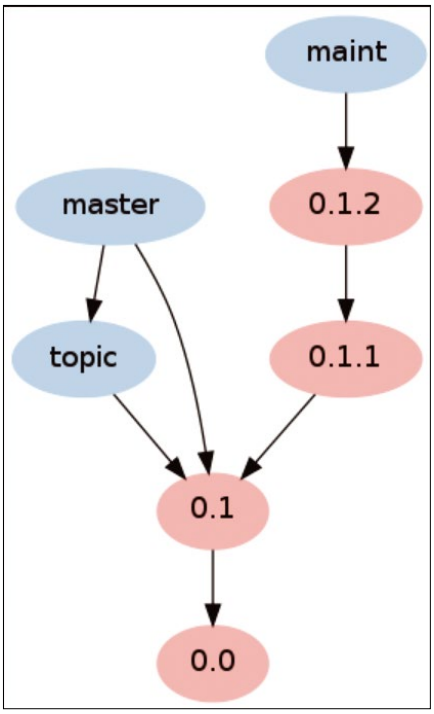


Figure 2: The *git-big-picture* utility provides a handy navigational aid, especially for large repositories.

has the latest updates. Then enter the command:

```
git merge branch OTHER-BRANCH
```

If only one branch is specified, it will be merged into the current branch. When one branch is newer than the other, it will be marked as `Fast-forward`, meaning it is a simple two-way merge (Figure 3) that will mark the differences in the branch being merged into and use minus signs to note differences in the branch being merged. In other words, the results are shown much as they are in the results of the `diff` command.

Git does its best to resolve any conflicts, and, all going well, the result is a new commit into the parent branch that contains the original commits plus a log message detailing the changes. However, if there is no direct connection, Git tries to create one with a three-way merge, assigning a third merge as a route between the other two, which makes conflicts more likely. This is why a clear map of a repo can be useful. If you think a three-way merge is likely, you can add the option `--no-commit` as a dry run to see what problems may emerge.

Since conflicts can be tedious to fix, especially in bulk, experts also suggest that you merge frequently. You may also decide to stop the merge with `git merge --abort`, and Git will try to reconstruct the original change to all branches, although any uncommitted changes may not be restorable.

To fix conflicts:

1. Run `git status` (Figure 4) to get a list of files that need to be resolved. In the files, `<<<<<<<` marks the start of a conflict and `>>>>>>>` marks the end. Use `=======` to mark the start and end of your manual changes as you work. Contents from the first branch are marked in plus signs and from the second batch in minus signs. Delete the conflict markers when you finish editing.
2. Run the `git add` command on the conflicting files when ready to merge.
3. Run `git commit -m "MESSAGE"` or `git merge --continue` to generate the merge commit.

If the conflicts are numerous or complicated, you may need to try a merge more than once to be successful.

Learning by Doing

These operations are not as complex as they may seem from my explanation. Working with Git involves doing the same thing over and over again, and these operations soon become second nature. Learn the ins and outs of `git-branch` and `git-merge`, and you will soon learn most of what you need to use Git day by day.

You'll understand, too, why Git was so quickly and widely accepted. Being originally written by Linus Torvalds helped, but the greatest reason for Git's popularity is that it takes the complex task of version control and makes it as simple as possible. ■■■

Info

[1] "Command Line – Git Version Control" by Bruce Byfield, *Linux Magazine*, issue 237, August 2020, pp. 54-57

[2] Git: *https://git-scm.com/*

[3] git branch: *https://git-scm.com/docs/git-branch*

[4] git merge: *https://git-scm.com/docs/git-merge*

[5] git-big-picture: *https://github.com/esc/git-big-picture*

```
bb@nanday:~/wip$ git merge draft2
Updating 0d15a9d..94f66be
Fast-forward
 part1.txt | 8 ++++++--
 1 file changed, 6 insertions(+), 2 deletions(-)
bb@nanday:~/wip$ vi part1.txt
bb@nanday:~/wip$ vi part1.txt
```

Figure 3: A merge where Git has figured how to merge two branches, each containing one file.

```
bb@nanday:~/wip$ git status
On branch 1st-draft
Untracked files:
  (use "git add <file>..." to include in what will be committed)

        .directory

nothing added to commit but untracked files present (use "git add" to track)
```

Figure 4: The command `git status` reports the merge results.

Too Swamped to Surf?

ADMIN
Network & Security

ADMIN offers additional news and technical articles you won't see in our print magazine.

Subscribe today to our free ADMIN Update newsletter and receive:

- Helpful updates on our best online features
- Timely discounts and special bonuses available only to newsletter readers
- Deep knowledge of the new IT

bit.ly/HPC-ADMIN-Update

Reading and writing metadata from multimedia files

Hidden Data

ExifTool lets you modify and analyze metadata in multimedia files from the command line, but its comprehensive feature set results in a lengthy learning curve. Luckily, jExiftoolGUI offers an intuitive interface that makes using ExifTool easier, even for less experienced users. *By Karsten Günther*

Multimedia files, such as pictures, videos, or music tracks, often contain metadata (additional information not found in the file name or file attributes). Exchangeable Image File Format (EXIF), the standard for storing this metadata, provides details such as recording date, shutter speed, and aperture, among other things. Introduced in 2010, EXIF was originally used for digital camera photos. Today, EXIF has other applications and includes far more data than originally intended. For instance, applications such as Geeqie and digiKam use an image's metadata to control the display order or to narrow a search.

Usually, metadata is defined once and then not edited, because it contains important information about the multimedia data. In practice, there are exceptions where metadata access is both useful and necessary. For instance, many digital cameras still do not store location data in the images. However, if the data exists in the form of a track recorded with a GPS device, it makes sense to include the location data as additional metadata in the file.

For these and many other tasks, Phil Harvey has developed ExifTool [1]. Implemented as a Perl library, ExifTool is the most comprehensive free software for reading, adapting, or adding metadata in multimedia files. Over the years, many users have contributed to ExifTool's development, making it an extremely extensive, but also complex, piece of software [2].

In addition to EXIF, ExifTool also supports Extensible Metadata Platform (XMP), the ISO 16684-1 standard originally developed by Adobe to embed metadata in digital media or store it as a sidecar file, and IPTC-Information Interchange Model (IPTC-IIM), the first multimedia news exchange format. ExifTool can even access makernotes, the meta-

Table 1: General Options

Option	Function
`-a`	Output tag duplicates (normally suppressed)
`-ee`	Extract metadata from embedded files
`-g`	Organize output by tag group (insert header)
`-G`	Print group name for each tag
`-F`	Repair makernotes
`-i DIR`	Ignore specified directory name
`-if [NUM] EXPR`	Conditionally process files
`-m`	Ignore minor errors or warnings
`-o OUTFILE`	Set output file or directory name
`-overwrite_original`	Overwrite original file by renaming temporary file (the default is to save an original file with the `_original` extension)
`-P`	Preserve file modification date/time
`-r`	Recursively process subdirectories
`-s`	Show tag names instead of tag descriptions
`-z`	Read/write compressed information
`-wm MODE`	Set mode for writing tags
`-progress`	Show file progress count

data not standardized by digital camera manufacturers. ExifTool is also capable of reliably processing RAW formats.

Learning all of ExifTool's options and tag names takes time. This article covers the basics of ExifTool and then introduces a graphical user interface (GUI) for ExifTool, jExifToolGUI, that makes using ExifTool easier.

Options

ExifTool's command-line interface offers precise control via many different options. ExifTool's general options (see Table 1) start with a minus sign, followed by one or more letters, and sometimes an argument. Table 2 shows additional options for special cases.

For multiple options, use the syntax `-r -s`. If you use `-rs`, ExifTool will not process the options. Some options are followed by a metadata keyword (a tag). Additionally, the tag may be followed by assignment operators and values.

If you specify a directory as the last argument in the command line, ExifTool automatically processes all files contained in the directory. If you specify individual files in the last argument, ExifTool processes only those files.

EXIF Tags

ExifTool analyzes the EXIF tags in the specified files, evaluates them, or assigns new values to them. The (partially) normalized EXIF tags consist of a name (or tag description) and one or more values. In many cases, the tag descriptions shown in the output do not correspond to the tag names specified as options. You can use `-s` to find out the correct tag names. Tag names are not case-sensitive.

Due to the different metadata families and tag groups, tags with the same name may occur multiple times. Normally, ExifTool shows only the first occurrence of a tag unless you instruct it to show all occurrences with the `-a` option.

Without further options, `exiftool` shows which EXIF tags a multimedia file contains. In the example in Listing 1, the output for the file is very sparse – normally there is much more metadata in a file. The output depends on the software used to create or edit the data.

Many tag names are self-explanatory. For a list of available tags, see the Perl module Image::ExifTool::TagNames. Some distributions offer a special man page (e.g., `/usr/share/man/man3/Image::ExifTool::TagNames.3pm.gz`). You can also find more information on the ExifTool website [3].

ExifTool classifies tags into seven families (0-6), with each family consisting of several groups. The first three families are of general interest, while the remaining families are for specific uses.

You can organize ExifTool's output based on these family groups by using `-g` or `-G` with the family number (0-6) to make the output details more precise. If you don't use the family number after these options, `-g` inserts lines and subtitles (e.g., `---- File ----`) into the output, while `-G` writes a group identifier at the beginning of each line (e.g., `[EXIF]`).

ExifTool normally displays a huge amount of information. To limit the output to certain tags, use the syntax shown in Listing 2. You can specify the `-TAG` parameter as often as you like. If you enter a tag at the command line that is missing in a file's EXIF tags, the corresponding line is omitted in the output.

The syntax for tags consists of a minus sign followed by the tag name. Tag names are often identical to the descriptions, but spaces can be omitted (e.g, `Focal Length` is the same as `FocalLength`, and `Circle Of Confusion` corresponds to `CircleOfConfusion`). However, sometimes tag names and descriptions don't match (as with `Depth Of Field`); in that case, search for the correct name with `-s`. When reading and writing metadata, it is important to exactly adhere to the syn-

Table 2: For Special Cases

Option	Function
`-@ ARGFILE`	Read command-line arguments from file
`-geotag TRKFILE`	Import geodata from file
`-globalTimeShift SHIFT`	Shift all formatted date/time values
`-delete_original`	Delete `_original` backups
`-restore_original`	Restore from `_original` backups (if they were saved in advance with the `_original` suffix)
`-overwrite_original_in_place`	Overwrite original by copying temporary file
`-common_args`	Define common arguments
`-execute [NUM]`	Execute multiple commands on one line
`-FileOrder [NUM]`	Set file processing order
`-if EXPR`	Conditionally process files

Listing 2: Limiting Output

```
$ exiftool -TAG... FILE|DIR
```

Listing 1: EXIF Tag Output

```
$ exiftool N1-30km.png
File Name                       : N1-30km.png
Directory                       : .
File Size                       : 1038 kB
File Modification Date/Time     : 2020:03:11 11:18:42+01:00
File Access Date/Time           : 2020:03:11 11:18:42+01:00
File Inode Change Date/Time     : 2020:03:13 11:24:57+01:00
File Permissions                : rw-r--r--
File Type                       : PNG
File Type Extension             : png
MIME Type                       : image/png
Image Width                     : 1920
Image Height                    : 1080
Bit Depth                       : 8
Color Type                      : RGB with Alpha
Compression                     : Deflate/Inflate
Filter                          : Adaptive
Interlace                       : Noninterlaced
Significant Bits                : 8 8 8 8
Image Size                      : 1920x1080
Megapixels                      : 2.1
```

tax, since small deviations often have unexpected effects.

There are two call variants for tags: `-TAG` enables a tag, while `--TAG` ignores it. The command

```
exiftool -keywords .
```

shows the keywords of all files in the current directory. In contrast,

```
exiftool --keywords .
```

returns all tags except for the keywords of all files in the current directory. Table 3 shows possible tag modifications.

Some values cannot be overwritten. For example, if you try to change the megapixels value, ExifTool outputs *Warning: Sorry, Megapixels is not writable*.

Using ExifTool

The best way to get to know ExifTool is to use it. The automatic backup option, which backs up the original files, prevents serious damage in most cases. Nevertheless, it is advisable to first work with copies of the original files.

To view the tags of a current directory's files, use line 1 of Listing 3. If the output is difficult to read, use the `-htmlDump` option to output an HTML-formatted binary dump that can be redirected to a file, which you can then view in your web browser.

Conveniently, ExifTool accepts values from tags as input for actions. For example, the command in line 2 of Listing 3 sorts the images into directories based on their timestamps. This action leaves the file names unchanged; `-P` also preserves the files' attributes.

If, on the other hand, you want to rename the files based on their timestamps, use the command in line 3 of Listing 3. In the argument for `-d`, `%Y%m%d` and `%H%M%S` represent the same values as the `date` command in the shell. `%e` stands for the suffix and `%f` for the previous file name. In this way you can customize file names individually.

For both examples in Listing 3, ExifTool evaluates the EXIF tags contained in the files. In the first example (line 2), `-Directory` creates new directories based on the value of *DateTimeOriginal*; in the second example (line 3), ExifTool builds the file name from the *CreateDate* tag. Since the redirection character (" < ") is included in ExifTool's syntax, you must put the entire expression in quotation marks – otherwise the shell will interpret the operator.

For some situations, you may need to change the EXIF tags. For example, if the camera clock was set incorrectly, it is often useful to correct the timestamps afterwards. You should be able to fix this by changing the *DateTimeOriginal* tag (Listing 4, line 1).

However, several other EXIF tags contain the timestamp. Because not all software used for viewing images works correctly with a certain specification, it is often better to use the *AllDates* pseudotag (Listing 4, line 2).

ExifTool is especially useful for adding geodata from external tracks for recordings that do not yet contain location data. Two options are important here: `-geotrack=TRKFILE` and `-geosync=MIN:SEC`. ExifTool can interpret most common geodata formats, especially GPX.

GPS devices and camera clocks are often not in sync. Since the time is used as a reference for the location, `-geosync=CORRECTION` lets you correct the times when assigning the positions without changing the timestamps in the images. A helpful tip: Before starting to take pictures, take a reference picture of the GPS screen with the time display to show the time difference between the camera and the GPS device.

You can correct fast camera times by using negative values for `-geosync=`, but it often makes more sense to change the wrong image timestamp directly in the images using `-AllDates`.

There are six standard and three optional geotags (see Table 4). ExifTool extracts these values – usually only the first six – from the GPS tracks, interpolating the location based on the time.

A simplified call for geodata looks like line 1 of Listing 5; it shows the camera clock is 12 minutes and 34 seconds slower than the GPS device. If you want to use multiple track files, specify them individually as arguments (Listing 5, line 2). Using a wildcard like `-geotag=*gpx` usually causes many error messages. The command from line 3 of Listing 5 deletes geodata from the files.

The message *Warning: Time is too far before track in File:Geotime (ValueConvInv)* indicates a problem with the time zone. GPS times are always given as Coordinated Universal Time (UTC), whereas camera clocks usually give the local time. You may have to take this into account when correcting the time difference [4].

Table 3: Tag Modifications

`-TAG=VALUE`	Assigns a value to the tag
`-TAG=`	Deletes all occurrences of a tag when = if followed by a space
`-TAG+=VALUE`	Adds the value to entries
`-TAG-=VALUE`	Removes value from entries

Listing 3: Sorting and Renaming by Timestamps

```
01 $ exiftool .
02 $ exiftool -P "-Directory<DateTimeOriginal" -d "%Y/%m/%d" DIRECTORY
03 $ exiftool -P "-FileName<CreateDate" -d "%Y%m%d_%H%M%S-%f.%%e" DIRECTORY
```

Listing 4: Changing Timestamps

```
01 $ exiftool "-DateTimeOriginal0PERATOR=YY:MM:DD d:m:s" .
02 $ exiftool "-AllDates0PERATOR=YY:MM:DD d:m:s" .
```

Table 4: Geotags

Standard	
`GPSDateTime`	Date and time
`GPSDateStamp`	Date
`GPSTimeStamp`	Time
`GPSLatitude`	Latitude
`GPSLongitude`	Longitude
`GPSAltitude`	Height above sea level
Optional	
`GPSTrack`	Direction
`GPSPitch`	Pitch angle
`GPSRoll`	Roll angle

Listing 5: Extracting Geodata

```
01 $ exiftool -geotag=mytrack.gpx -geosync=+12:34 .
02 $ exiftool -geotag=1.gpx -geotag=2.gpx .
03 $ exiftool -geotag= .
```

An ExifTool GUI

Several projects have attempted to make the complex system of EXIF tags more accessible with a graphical user interface. At least three of these projects can be found in the Arch Linux repositories:

- DigiKam is up-to-date and works quite well, but sometimes it uses special tags that are not fully compatible with ExifTool's tags.
- PyExiftoolGUI is a Python 2-based (now obsolete) application that causes problems during installation.
- jExifToolGUI [5] ports pyExifToolGUI to Java. The developer provides a Debian package, an AppImage, and a JAR archive with the current version [6]. The universal JAR should launch on all platforms.

Please note that jExifToolGUI's developer, Harry van der Wolf, has been rapidly implementing many new features during 2020. The features described here are as of Version 1.4.

The jExifToolGUI interface (Figure 1) is fairly easy to use. The *Load Directory* and *Load Image* buttons open images. To view the embedded preview in RAW files, jExifToolGUI requires the `dcraw` command-line program. A double-click on a file name or a thumbnail opens the image in a display program of your choice.

The five tabs on the right side of the program window organize jExifToolGUI's actions. *View Data* only displays an image's data, *Edit Data* lets you edit some of the data, and *Copy Data* supports transferring data between images. In *Your Commands*, you can define and execute your own ExifTool command lines. The *Exiftool Database* tab helps you search for tags.

When you start, the *View Data* tab is active: In the right-hand pane, with the *All* button selected, jExifToolGUI shows you all output generated by ExifTool when you call it with a file name and without options. Usually this results in far too much information, so it is often more useful to limit the output to certain groups.

You can limit output with the *Common Tags* button, which limits the information accordingly in the field to the right. Many images do not actually contain information in all groups, so the output remains empty. The *By Group* and *By Camera* options have similar effects.

Editing EXIF tags is one of jExifToolGUI's main focuses. The *Edit Data* tab has seven categories that give you direct access to the data. Using the first tab, *Exif*, you can load the standard EXIF tags from one image onto the clipboard and then transfer them to all selected images.

The *xmp* tab is used for editing XMP tags. The tags summarized here let you describe the subject, not the image's technical properties. In particular, the *Keywords* tag is accessible here; you can enter several keywords separated by commas.

The *gps* tab lets you manually transfer the location data of one image to several others. To do this, first select the source image and copy the data to the buffer with *Copy from selected image*. Then select the target images and transfer the buffer's contents by selecting *Save to selected image(s)*. *Open MapCoordinates*.

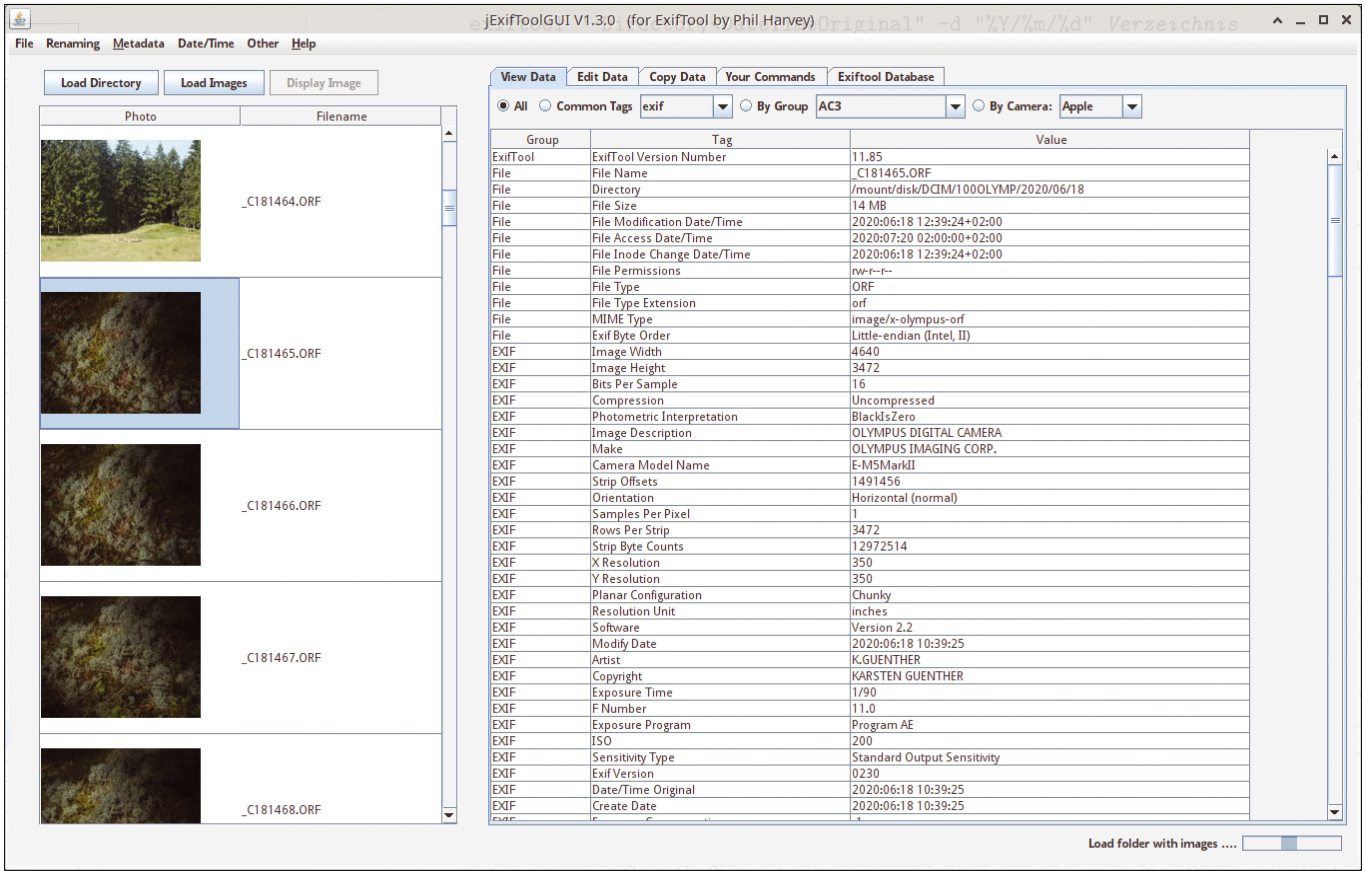


Figure 1: jExifToolGUI presents the images and their EXIF tags in a clear and concise way.

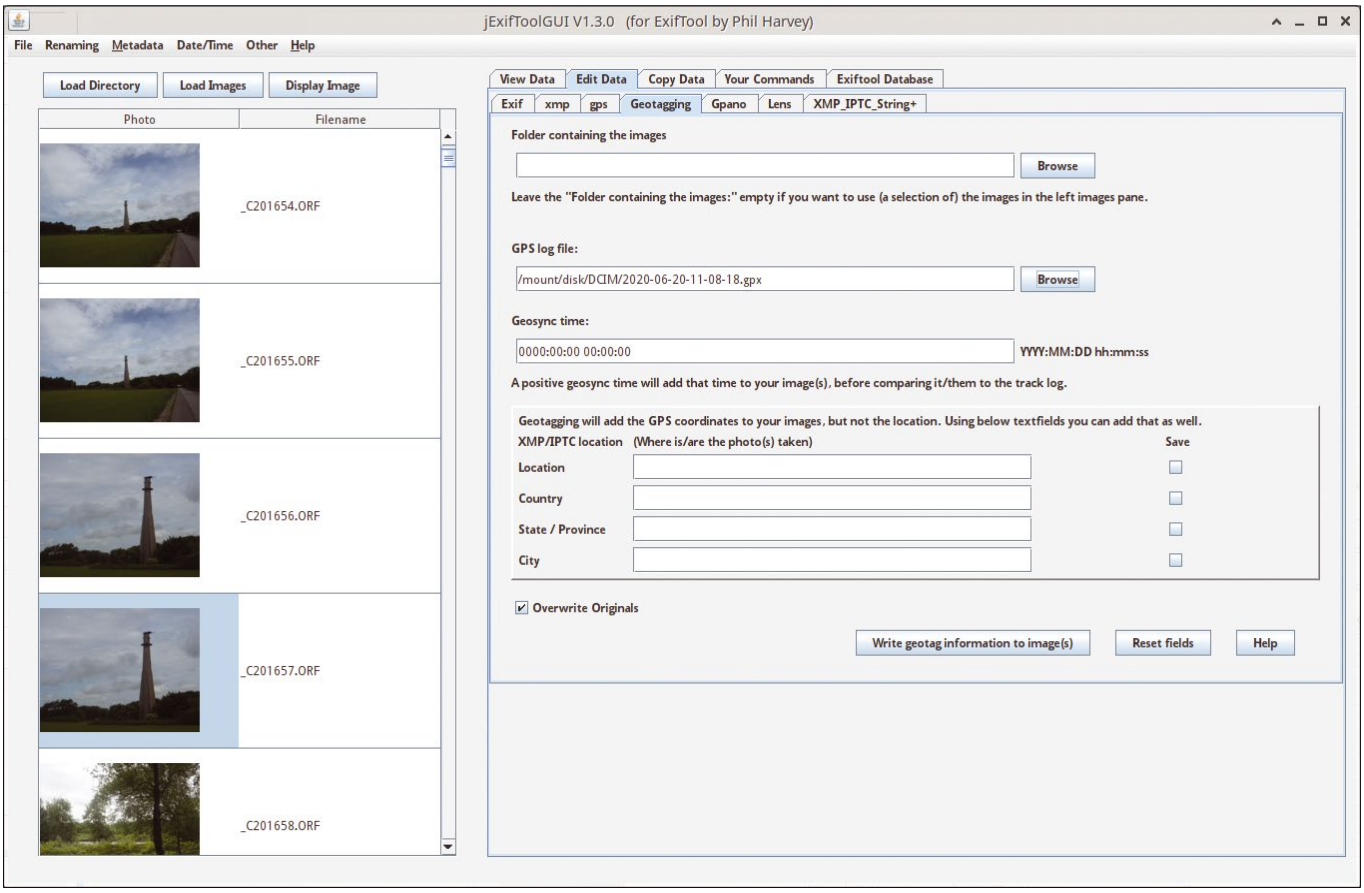


Figure 2: ***Geotagging*** **makes it easy to link geodata from external tracks to images.**

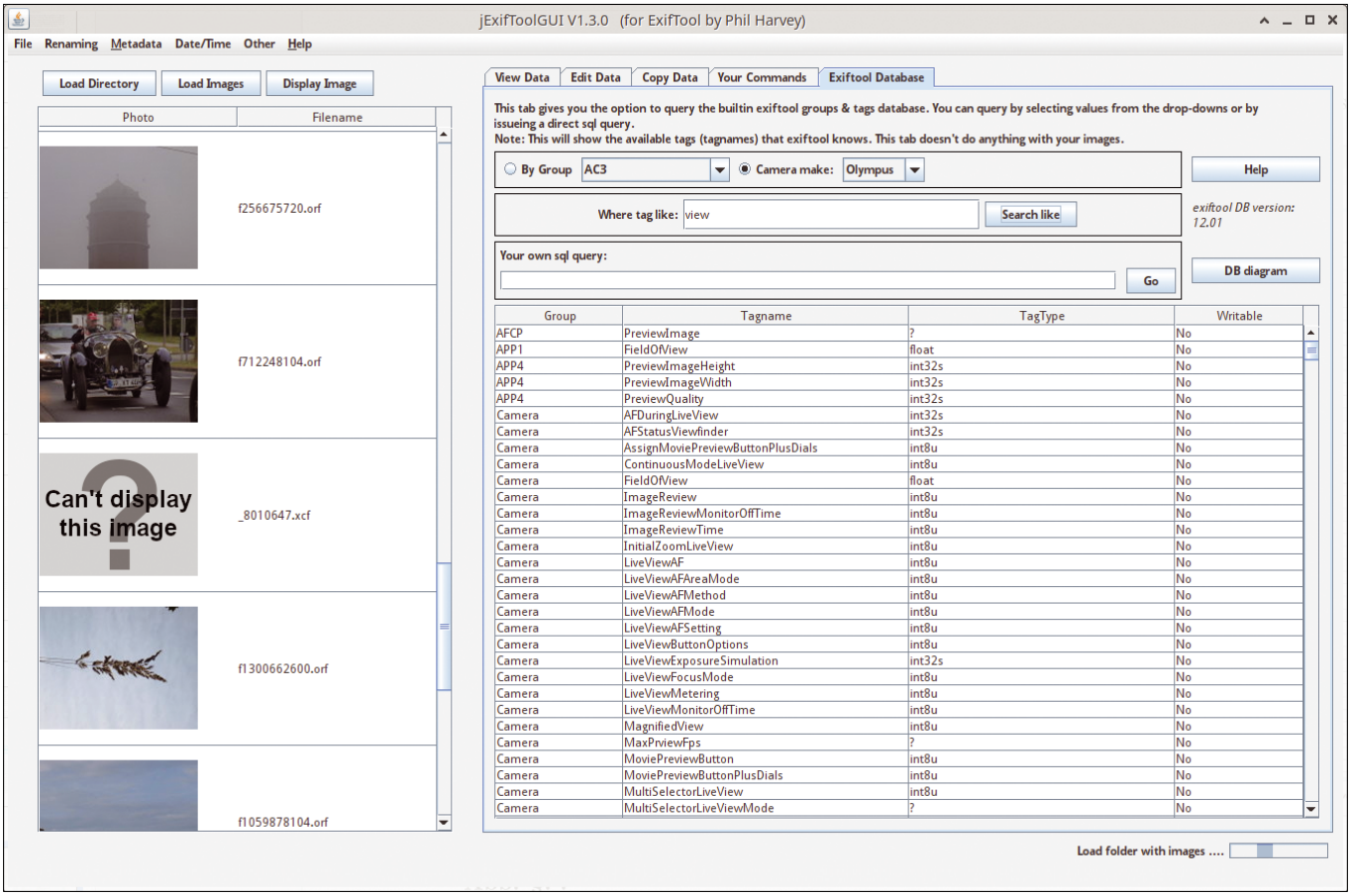


Figure 3: **Search all tags of all groups with** ***Exiftool Database*****.**

net gives you an additional option for defining an image's coordinates via the web browser.

Geodata from external tracks can be added to the images in the *Geotagging* tab. To do this, either load a folder with images in advance or use the currently loaded images. Then load a file with the track data in a format that can be interpreted by ExifTool in the *GPS log file* field.

Geosync time lets you compensate for the time offset between the camera and tracker as described above. *Write geotag information to image(s)* then writes the position data to the image files (Figure 2).

Additional tabs under *Edit Data* (*Gpano*, *Lens*, and *XMP_IPTC_String+*) serve very specialized purposes.

The *Your Commands* tab lets you execute any command line. The developer was planning significant changes for this dialog in upcoming versions, including an additional history.

The last tab *Exiftool Database* (Figure 3), which is more for advanced users, grants access to the internal ExifTool tag database, with all currently known tags of all families and groups. This is especially easy to use thanks to the *Search like* input box, which performs a full text search. The *By Group* and *Camera make* buttons let you restrict the results. jExifToolGUI emulates the internal database in the form of an SQLite database, which enables structured queries.

Conclusions

As Phil Harvey moves ahead with ExifTool's development, he is maintaining compatibility with new standards and image formats. However, this development inevitably increases the complexity, making it more difficult to learn. So far, however, Harvey has succeeded in ensuring backward compatibility. Command lines that used to work successfully in the past will usually still work today.

Thanks to the jExifToolGUI user interface, which is also progressing in giant leaps, working with ExifTool is easier, especially for less experienced users. ■■■

Info

[1] ExifTool: *https://exiftool.org*

[2] Additional documentation and resources: *https://exiftool.org/#links*

[3] Tag names: *https://exiftool.org/TagNames/*

[4] Geotag examples: *https://exiftool.org/geotag.html#Examples*

[5] jExifToolGUI: *https://github.com/hvdwolf/jExifToolGUI/*

[6] jExifToolGUI releases: *https://github.com/hvdwolf/jExifToolGUI/releases*

The sys admin's daily grind: History

History Lesson

For admins like Charly, who try to avoid typing at all costs, the shell offers an excellent opportunity to avoid wear on your fingertips in the form of built-in history. *By Charly Kühnast*

There are commands that I type several dozen times a day – `grep <something> /var/log/syslog` is such a classic. The shell keeps a history of all my entries; thanks to the `history` command, I can always see in a numbered list which commands I typed last.

The `history` command is not a separate tool; typing `which history` at the command line just drops you into a black hole. Instead, `history` is a part of the shell, a built-in keyword. `history`'s killer feature, for which lazy people like me are eternally grateful, is the interactive search. You enable it with Ctrl + R, changing the command-line prompt to `(reverse-i-search)`':`.

If you start typing now, for example, the word *net*, the shell will show you the last command typed containing *net*. When you press Ctrl + R again, the history feature shows you an increasing number of older commands that contain *net* (Figure 1).

There are a number of other ways to execute commands stored in the history one more time. To repeat just the last command entered, you can do any of the following:

- Press the up arrow
- Press Ctrl + P ("previous" on keyboards without arrow keys)
- Type `!!`
- Type `!-1`

Sometimes using relative addressing backwards through history proves helpful. In the example from Figure 2, I reran the third-to-last command from the history by typing `!-3`. If you wanted to repeat the last command that started with *echo*, you would just need to type `!echo`.

You can also access the parameters from previous commands. If you just typed `ls .bashrc`, you can enter `vim !!:$` to open `.bashrc` in the editor. If you have a command that requires root privileges, `sudo !!` does the trick. In the meantime, I defined `but` as an alias (Figure 3).

Occasionally, however, I find the history's length problematic, as it only stores 1,000 entries on my test system. This is not enough for me, so I added a `HISTSIZE=10000` line to the `/.bashrc` file to multiply the history size by 10. I also added `HISTCONTROL=erasedups` to `/.bashrc`. This means that the `history` command, which I type several times, is only saved once – this saves space and gives a better overview. ■■■

```
(reverse-i-search)`net': sudo nethogs enp1s0
```

Figure 1: Interactive search with Ctrl+R.

```
charly@glas:~$ echo "brave"
brave
charly@glas:~$ echo "new"
new
charly@glas:~$ echo "world"
world
charly@glas:~$ !-3
echo "brave"
brave
```

Figure 2: Going back three commands.

```
charly@glas:~$ alias but='sudo $(history -p !-1)'
charly@glas:~$ ls -lha /root/.bashrc
ls: cannot access '/root/.bashrc': Permission denied
charly@glas:~$ but
-rw-r--r-- 1 root root 3.1K Oct 22  2015 /root/.bashrc
```

Figure 3: No? But! Oooh…

Author

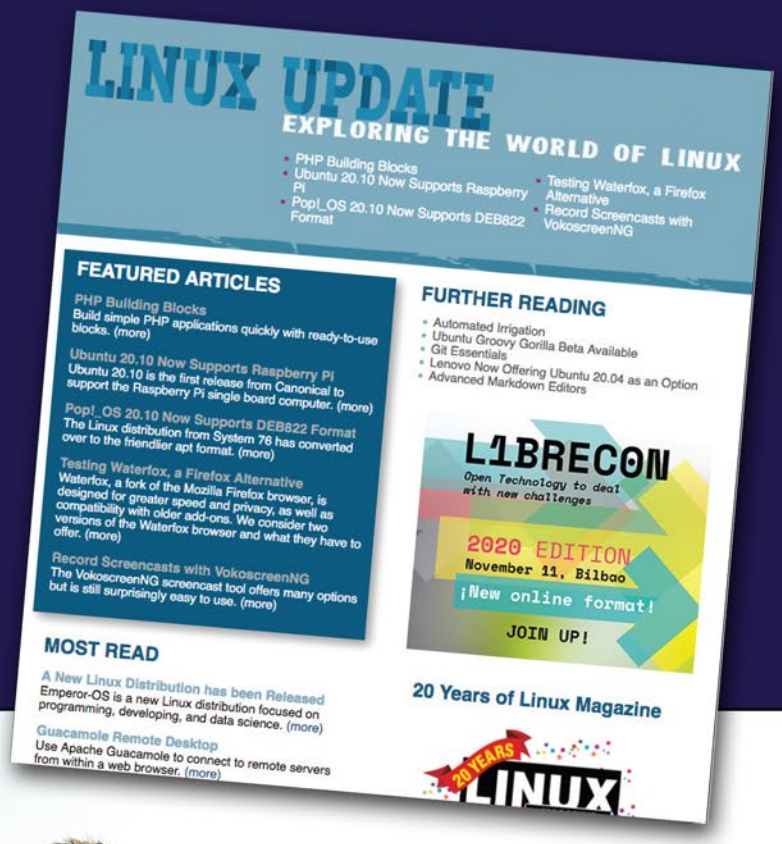
Charly Kühnast manages Unix systems in the data center in the Lower Rhine region of Germany. His responsibilities include ensuring the security and availability of firewalls and the DMZ.

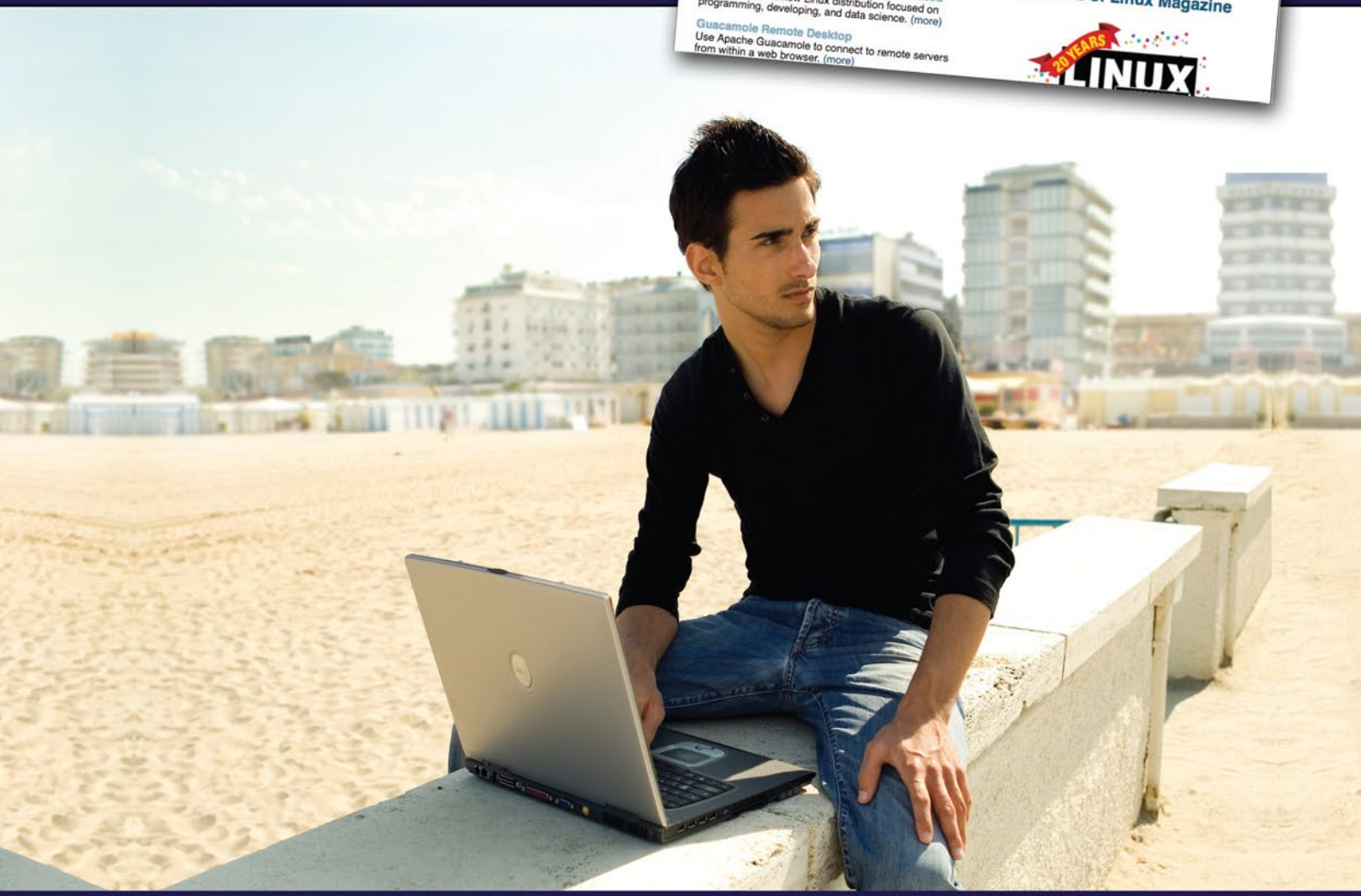
LINUX UPDATE

Need more Linux?

Our free Linux Update newsletter delivers insightful articles and tech tips to your mailbox twice a month. You'll discover:

- **Original articles on real-world Linux**
- **Linux news**
- **Tips on Bash scripting and other advanced techniques**
- **Discounts and special offers available only to newsletter subscribers**





bit.ly/Linux-Update

Decoding videos in Firefox with VA-API

Full Potential

Today's graphics cards not only specialize in quickly drawing graphics on the screen, but they can also speed up video playback, reducing the CPU's load. While Firefox supports this optimization, you must manually enable hardware acceleration to quickly decode video on the browser. *By Paul Menzel*

For awhile, graphics cards have contained dedicated components for decoding and encoding video formats such as MPEG2, H.264, and H.265. In chip developer jargon, these embedded units are called application-specific integrated circuits (ASICs) [1].

CPUs without ASICs have a hard time decoding videos. However, graphics cards with their specialized circuits can decode video even on low-powered systems like the Raspberry Pi without jerking and with lower power consumption, thus shifting some of the load off the main processor. Lower power consumption means longer battery life and less heat generated, which in turn makes the system quieter.

Interface

On Linux, programs access the graphics chip's acceleration functions via the Video Acceleration API (VA-API) [2]. The `vainfo` tool (included in the *vainfo* or *libva-utils* packages) shows you the algorithms supported by the graphics card. The `VLD` suffix indicates the decoding capability, while `EncSlice` denotes the encoding capability.

For example, the integrated Intel 5500 HD graphics chip in the Intel CPU i7 5600U processor (based on the Broadwell architecture) supports decoding and encoding of MPEG2 and H.264, but not VP9 (Listing 1). Intel's eighth generation Core microprocessor ("Coffee Lake") – for example, an i7 9700 with UHD Graphics 630 – can already decode and encode VP9 (Listing 2). An older AMD graphics card only speeds up de-

Listing 1: Intel 5550 HD Graphics Chip

```
$ vainfo
[...]
vainfo: Driver version: Intel iHD driver for Intel(R) Gen Graphics - 20.2.0 ()
vainfo: Supported profile and entrypoints
[...]
    VAProfileMPEG2Simple : VAEntrypointVLD
    VAProfileMPEG2Simple : VAEntrypointEncSlice
[...]
    VAProfileH264Main : VAEntrypointVLD
    VAProfileH264Main : VAEntrypointEncSlice
    VAProfileH264Main : VAEntrypointFEI
[...]
    VAProfileVP8Version0_3 : VAEntrypointVLD
```

Listing 2: Intel Coffee Lake Graphics Card

```
$ vainfo
libva info: VA-API version 1.3.0
[...]
vainfo: Driver version:
   Intel i965 driver for Intel(R) Coffee Lake - 2.3.0.pre1 (2.3.0.pre1)
[...]
    VAProfileVP9Profile0 : VAEntrypointVLD
    VAProfileVP9Profile0 : VAEntrypointEncSlice
    VAProfileVP9Profile2 : VAEntrypointVLD
```

Photo by Joshua Earle on Unsplash

Listing 3: AMD Graphics Card

```
$ vainfo
[...]
vainfo: Driver version: Mesa Gallium driver 20.0.1 for AMD OLAND (DRM 3.39.0,
  5.8.0.mx64.339-13250-g66bc99e8bbd14, LLVM 9.0.1)
    VAProfileMPEG2Simple : VAEntrypointVLD
    VAProfileMPEG2Main : VAEntrypointVLD
    VAProfileVC1Simple : VAEntrypointVLD
    VAProfileVC1Main : VAEntrypointVLD
    VAProfileVC1Advanced : VAEntrypointVLD
    VAProfileH264ConstrainedBaseline: VAEntrypointVLD
    VAProfileH264Main : VAEntrypointVLD
    VAProfileH264High : VAEntrypointVLD
    VAProfileNone : VAEntrypointVideoProc
```

Listing 4: Checking Graphics Card Use

```
$ mpv --hwdec=vaapi /<path>/<to>/video.mkv
[...]
VO: [gpu] 1920x1080 vaapi[nv12]
VO: [gpu] 1920x1080 yuv420p
[...]
```

coding of H.264, but this GPU does not yet work with VP8 and VP9 (Listing 3).

Because not all required software components are under a free license, you have to enable VA-API support on Ubuntu or Debian (for example) for fifth generation Intel chips (Broadwell) by installing *intel-media-va-driver-non-free* from the package sources.

Then you use the mpv media player to check if the graphics card is used during decoding. To do this, call the player with the `--hwdec=vaapi` option. If the program reports

```
VO: [...] vaapi
```

when it starts playing back a video, the interface is working (Listing 4). Involving the dedicated decoder chip reduces the entire system's energy consumption. For example, PowerTOP now reports only 6.32W (Figure 1) instead of 7.87W (Figure 2).

Web Browser

Today many Internet sites – not least YouTube – integrate videos, resulting in web browsers often functioning as video playback programs. Users can benefit in particular if the browser can access the chips on the graphics card during decoding. With the WebRender [3] compositor, Mozilla laid the foundation for using the graphics card to display web pages. WebRender lets you use hardware acceleration in addition to the GL Compositor. However, even the current Firefox 80 only enables the function for Windows 10 [4] by default, because Linux developers complain that there are too many problems with the graphics card drivers.

As is so often the case, a Red Hat developer has taken the Linux desktop another step forward with Linux graphics card drivers. Martin Stransky pro-

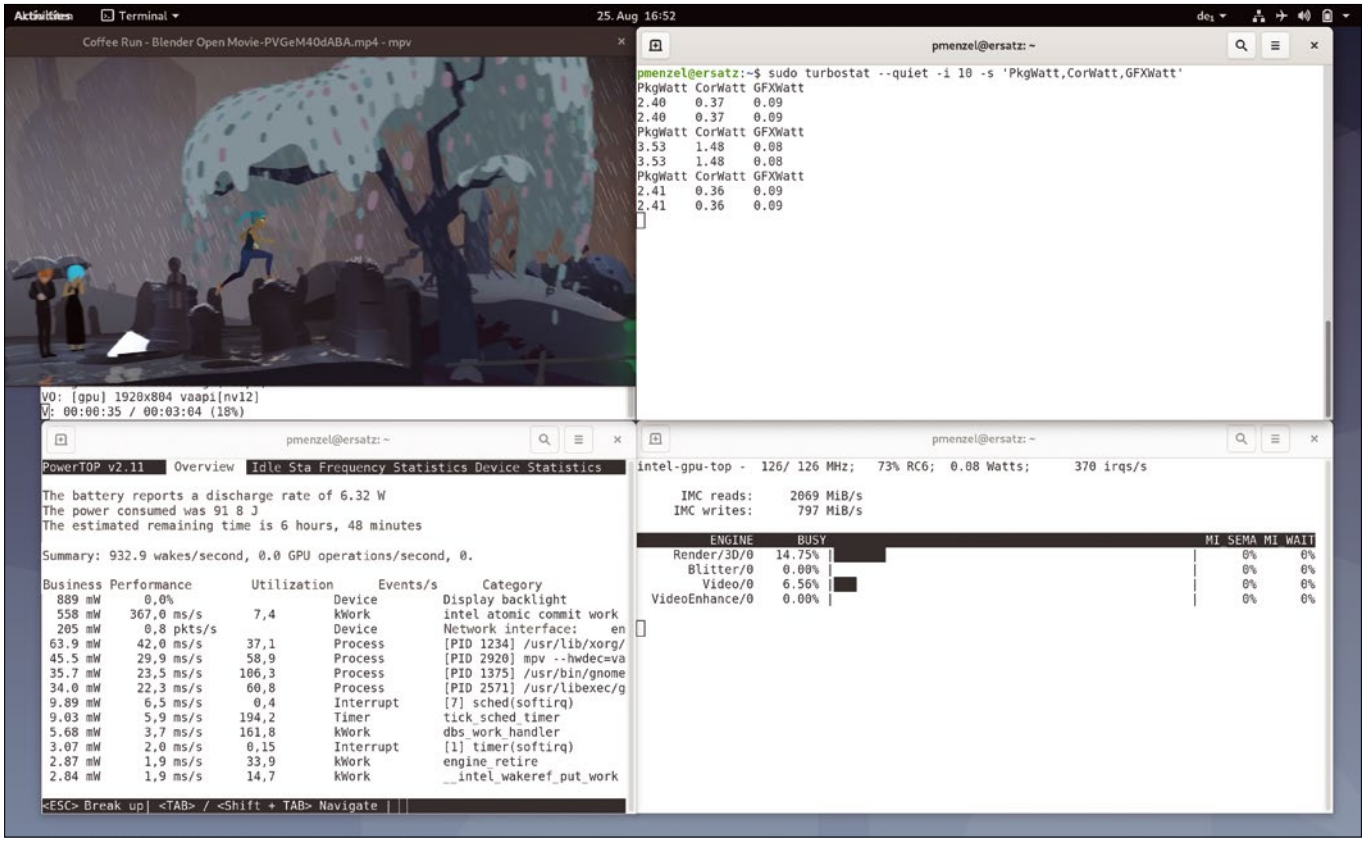


Figure 1: Launched with the options `--vo=gpu --hwdec=vaapi`, mpv accesses the specialized decoder chips on the graphics card.

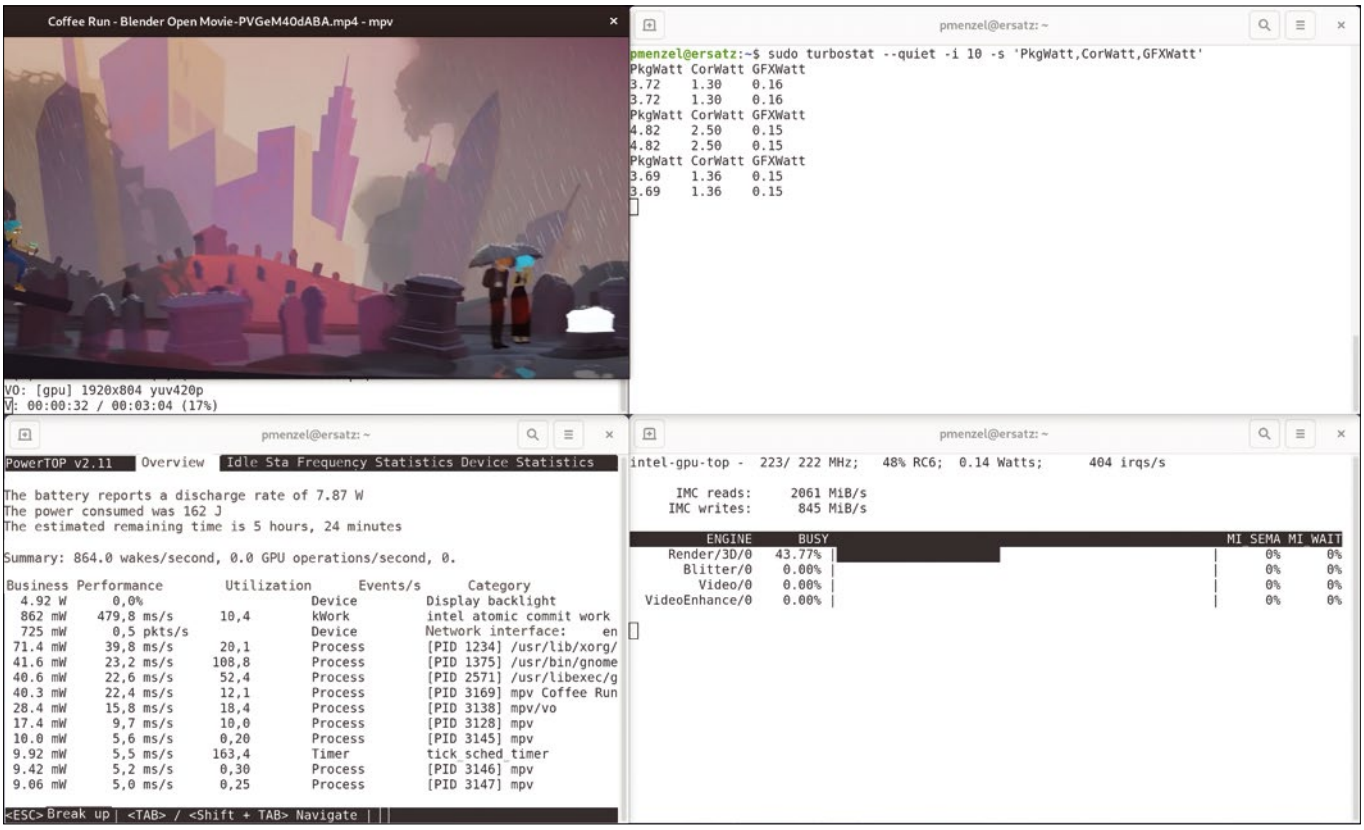


Figure 2: The system's energy consumption decreases noticeably when VA-API is activated. Without the optimization, the system draws 1.5W more battery power.

grammed the changes for Firefox; from Fedora 31 on, they are implemented in Firefox 77.0. In his blog post [5], Stransky reports that thanks to Wayland, many errors in the Linux graphics card drivers have now been discovered and fixed, as Wayland makes intensive use of the acceleration functions. For hardware-accelerated decoding and encoding, the developer resorts to the proven FFmpeg [6], for which Firefox writes the material directly to the graphics card memory via the dma-buf subsystem, thus removing the need to copy the data. By default, the functions are disabled.

In a Wayland desktop environment, calling

```
MOZ_ENABLE_WAYLAND=1
  MOZ_WEBRENDER=1 firefox
```

from the terminal activates WebRender, as well as the VA-API interface. After appropriate configuration, Listing 5 shows the browser call with VA-API in an X11 environment, which Firefox has supported since version 80. The `MOZ_LOG` variable controls the output messages.

Next, open the configuration editor with the pseudo-URL *about:config* and enable VA-API support by double-clicking on *media.ffmpeg.vaapi.enabled* to switch from *false* to *true*. To find out if WebRender works, check the URL *about:support* in the *Graphics* section. This is where Firefox should output information about WebRender.

Currently, YouTube delivers VP9-encoded movies by default due to the smaller size for the same quality. For hardware-accelerated VP8/VP9 playback in Firefox, do not load the *ffvpx* library provided by Firefox, because it cannot use the VA-API interface. Setting the *media.ffvpx.enabled* key to *false* in *about:config* tells Firefox to use the system FFmpeg libraries [7]. Some distributions already configure Firefox accordingly.

If the graphics card does not support H.264, the Firefox h264ify [8] or enhanced-h264ify [9] extensions block formats encoded with VP8 and VP9. If you find messages in the logs such as

```
D/PlatformDecoderModule DMA-Buf/VA-API
  can't be used, WebRender/DMA-Buf is
  disabled
```

please check the settings again.

Listing 5: Browser Call with VA-API

```
$ MOZ_X11_EGL=1 MOZ_WEBRENDER=1 MOZ_LOG="PlatformDecoderModule:5" firefox
[...]
[Child 19296: MediaPDecoder #3]:
    D/PlatformDecoderModule Choosing FFmpeg pixel format for VA-API video
    decoding.
[Child 19296: MediaPDecoder #3]:
    D/PlatformDecoderModule Requesting pixel format VAAPI_VLD
[h264 @ 0x7f01f5720000] Format vaapi_vld chosen by get_format().
[h264 @ 0x7f01f5720000] Format vaapi_vld requires hwaccel initialization.
[h264 @ 0x7f01f5720000] Considering format 0x3231564e -> nv12.
[h264 @ 0x7f01f5720000] Picked nv12 (0x3231564e) as best match for yuv420p.
[...]
```

Troubleshooting

If problems occur when playing videos, you should first check whether the errors still occur in the nightly build of the browser with the latest changes. If the bugs are confirmed, you can contribute to the project by reporting the problem to Mozilla's bug tracker [10], appending the output of `MOZ_LOG="PlatformDecoderModule:5"`.

Download the current nightly build [11] for Linux to your computer as shown in Listing 6 and unpack the archive; the last command starts the browser. Make sure that you close the regular Firefox instance completely before doing this. After completing testing, you can delete the `firefox/` directory created by unpacking.

Listing 6: Downloading the Firefox Nightly Build

```
### Download 64-bit version:
$ wget 'https://download.mozilla.org/?product=firefox-nightly-latest-
    ssl&os=linux64&lang=en-US' -O firefox-nightly64.tar.bz2
### Download 32-bit version:
$ wget 'https://download.mozilla.org/?product=firefox-nightly-latest-
    ssl&os=linux&lang=en-US' -O firefox-nightly32.tar.bz2
### Unzip and call:
$ tar xf firefox-nightly*.tar.bz2
$ firefox/firefox
```

Listing 7: Power Consumption Values

```
$ turbostat --quiet -i 3 -s 'PkgWatt,CorWatt,GFXWatt,RAMWatt'
PkgWatt CorWatt GFXWatt RAMWatt
4.21 1.55 0.13 1.02
4.21 1.55 0.13 1.02
[...]
```

Outlook

Use the `turbostat` tool to determine whether the energy consumption when playing videos using VA-API is now lower (see the "Consumption Display" box). Table 1 shows the averaged values when playing a video via the mpv media player and for Firefox on a Dell Latitude E7250 with the Broadwell chipset (Intel Core i5 5300U). While mpv obviously benefits from the optimized decoder chip, the savings in Firefox with We-

Consumption Display

The `turbostat` tool is part of the Linux kernel and is built into the *linux-cpupower* package on Debian. Other distributions, such as Arch Linux, store the program in the *turbostat* standalone package. The application reads the energy consumption statistics provided by the computer system. The command in Listing 7 displays the power consumption values at intervals of three seconds. `PkgWatt` stands for package (i.e., the chipset), `CorWatt` for the processor, `GFXWatt` for the graphics card, and `RAMWatt` for the main memory. Press Ctrl+C to exit `turbostat` and return to the prompt.

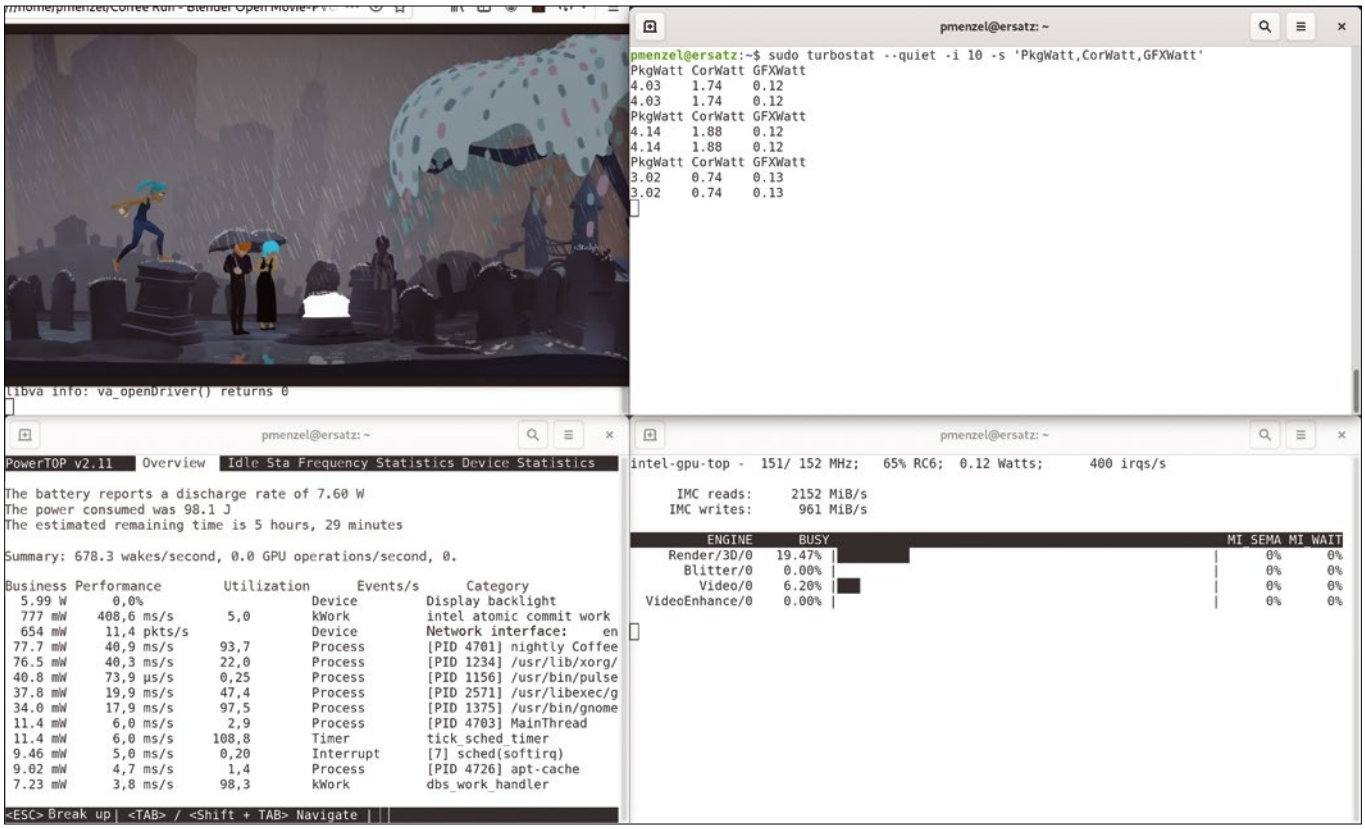


Figure 3: The Firefox web browser can also use hardware acceleration, but support for this function is not yet standard under Linux.

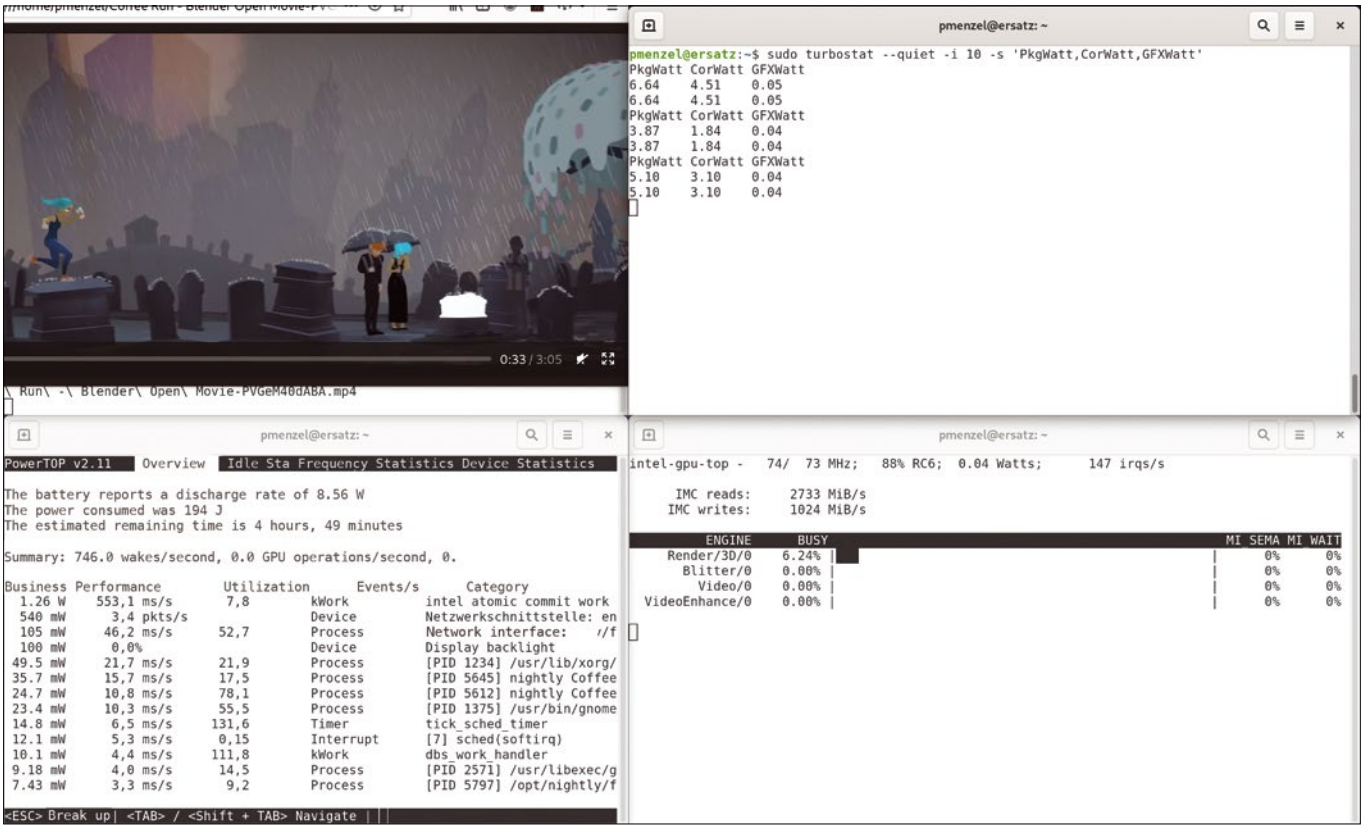


Figure 4: Without VA-API, the system draws nearly 1W more power from the laptop's battery when playing back the video in Firefox's current nightly build.

bRender are somewhat lower. PowerTOP reports a discharge rate of 7.60W (Figure 3) for VA-API; without it, the system draws 8.56W of power from the notebook battery when playing the video in the browser (Figure 4).

Thanks to Red Hat's commitment, Firefox on Linux will soon use the graphics card for decoding videos by default, allowing Firefox to finally catch up with the Microsoft Windows 10 version in terms of functionality. Default VA-API integration removes another obstacle to achieving the year of the Linux desktop (if that year ever happens).

As videoconferencing needs increase, encoding videos (of yourself) becomes just as important as decoding the received video data. Many image streams are encoded with VP8, which many graphics cards support. Firefox 80 or 81 can use VA-API for WebRTC [12] after enabling *media.ffmpeg.low-latency.enabled* in *about:config*. Hopefully, Firefox will soon enable VA-API integration by default, doing away with all of the manual work. ■■■

Table 1: Energy Consumption

Test	PkgWatt	CorWatt	GFXWatt	PowerTOP
mpv (hwdec=no)	4.08W	1.72W	0.15W	7.87W
mpv (hwdec=vaapi)	2.78W	0.74W	0.09W	6.32W
Nightly (plain)	5.20W	3.15W	0.04W	8.56W
Nightly (VA-API)	3.73W	1.45W	0.12W	7.60W

Info

[1] ASIC: *https://en.wikipedia.org/wiki/Application-specific_integrated_circuit*

[2] VA-API: *https://en.wikipedia.org/wiki/Video_Acceleration_API*

[3] WebRender: *https://github.com/servo/webrender/wiki*

[4] WebRender on Linux: *https://wiki.mozilla.org/Platform/GFX/WebRender_Where#Linux*

[5] "Firefox on Fedora finally gets VA-API on Wayland": *https://mastransky.wordpress.com/2020/06/03/firefox-on-fedora-finally-gets-va-api-on-wayland*

[6] FFmpeg: *https://ffmpeg.org*

[7] "Add VA-API decode path to bundled ffvpx": *https://bugzilla.mozilla.org/show_bug.cgi?id=1660336*

[8] h264ify: *https://addons.mozilla.org/en-US/firefox/addon/h264ify*

[9] enhanced-h264ify: *https://addons.mozilla.org/en-US/firefox/addon/enhanced-h264ify*

[10] Mozilla Bugzilla: *https://bugzilla.mozilla.org*

[11] Download Firefox: *https://www.mozilla.org/en-US/firefox/channel/desktop/*

[12] "Use VA-API decoder with WebRTC": *https://bugzilla.mozilla.org/show_bug.cgi?id=1646329*

Run statistics on typed shell commands

Making History

In the history log, the Bash shell records all commands typed by the user. Mike Schilli extracts this data with Go for a statistical analysis of his typing behavior. *By Mike Schilli*

How did the long command to connect to the database server work again? Shell power users know this memory problem and have taught themselves tricks over the years to repeat or resend modified versions of previously sent command lines. For example, the sequence `!!` repeats the last command (useful to prepend a `sudo` you might have forgotten); with Ctrl + R you can find and reuse commands that were sent further back in time by using search patterns. Bash remembers the entire typing history and shows it with the `history` command, which lists the tail end of what it has in memory (Figure 1).

If you have ever taken a look at the `.bash_history` file in your home directory, you will know its little secret. This is where Bash simply appends every command sent, regardless of if it was successful or aborted with an error. If you ask for the last commands you entered, Bash simply looks there (Figure 2).

```
$ history
    1  cd ~/git/articles/history/eg
    2  go build dow.go
    3  ./dow
    4  cd ..
    5  vi t.pnd
    6  make publish
    7  git add t.pnd eg/*.go
    8  git commit
    9  git push
   10  history
$
```

Figure 1: The `history` command displays the last commands that were issued.

Timestamp It!

With a small change, Bash not only logs typed commands, but even adds the date and time when the commands were entered. To do this, you need to set the following environment variable (preferably in the `.bash_profile` init file):

```
export HISTTIMEFORMAT="%F %T: "
```

From then on, Bash adds a comment line with the Unix epoch stamp before each command recorded in the `.bash_history` logfile, while the `history` command prints the human readable date and time for each command in the display list (Figure 3).

This newly discovered data treasure trove begs for some kind of evaluation,

```
#1601840042
cd ~/git/articles/history/eg
#1601840043
go build dow.go
#1601840050
./dow
#1601840057
cd ..
#1601840076
vi t.pnd
#1601840077
make publish
#1601840095
git add t.pnd eg/*.go
#1601840100
git commit
#1601840100
git push
#1601840116
history
~
".bash_history" 20L, 249C    3,1           All
```

Figure 2: In the `.bash_history` file, Bash records which command was executed and when.

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.

```
$ history
    1  2020-10-04 12:34:02: cd ~/git/articles/history/eg
    2  2020-10-04 12:34:03: go build dow.go
    3  2020-10-04 12:34:10: ./dow
    4  2020-10-04 12:34:17: cd ..
    5  2020-10-04 12:34:36: vi t.pnd
    6  2020-10-04 12:34:37: make publish
    7  2020-10-04 12:34:55: git add t.pnd eg/*.go
    8  2020-10-04 12:35:00: git commit
    9  2020-10-04 12:35:00: git push
   10  2020-10-04 12:35:16: history
$
```

Figure 3: If `HISTDATEFORMAT` is set, the history returns the latest commands executed along with the timestamp.

of course. How about determining the days of the week on which a user was most diligent in terms of typing commands? Or how about identifying the most frequently typed commands in order to find new ways to reduce the future typing overhead by using abbreviations or shell scripts?

Callback as an Abstraction

Listing 1 [1] defines the `histWalk()` function, which finds the global history logfile for a user in the user's home directory, browses it line by line, and calls a user-provided callback function for each command entry. In this way, other analysis programs can define their own callbacks, pass it to `histWalk()`, and receive the history data including timestamps this way, without having to worry about the details of the history logfile's location or format.

The logfile opened for reading in line 19 by `os.Open()` is read line-by-line by the scanner created in line 24. To do so, it grabs the reader interface of the opened file, which is offered via the `File` structure, and takes in a new line with every call to `Scan()`; `scanner.Text()` then returns the text line as a string.

If the line starts with a comment character, it is a timestamp for a command in the subsequent log line. Accordingly, Listing 1 stores the seconds field in the `timestamp` variable (after cutting off the leading comment character) and sends the scanner off into the next round. If there is no hash sign at the beginning, it is a command line, and the program jumps to the `else` branch starting in line 34, where it calls the callback function entered by the user. As parameters, it passes in the previously saved timestamp and the currently read shell command.

First Class Functions

In Go, functions are first class citizens: They can be assigned to arbitrary variables or passed into other functions in parameter lists. For example, user programs can use the `histWalk()` function offered by Listing 1, assign it a callback function that is completely adapted to their special needs, and let it fill existing data structures in the local scope with the results.

```
$ go build dow.go histWalk.go
$ ./dow
Sunday: 142
Monday: 327
Tuesday: 275
Wednesday: 301
Thursday: 217
Friday: 218
Saturday: 231
$
```

Figure 4: The binary compiled from Listing 2 shows the typing activity per weekday.

Listing 1: histWalk.go

```
01 package main
02
03 import (
04   "bufio"
05   "os"
06   "os/user"
07   "path/filepath"
08   "strconv"
09 )
10
11 func histWalk(cb func(int64, string) error) error {
12   usr, err := user.Current()
13   if err != nil {
14     panic(err)
15   }
16   home := usr.HomeDir
17   histfile := filepath.Join(home, ".bash_history")
18
19   f, err := os.Open(histfile)
20   if err != nil {
21     panic(err)
22   }
23
24   scanner := bufio.NewScanner(f)
25   var timestamp int64
26   for scanner.Scan() {
27     line := scanner.Text()
28     if line[0] == '#' {
29       timestamp, err = strconv.ParseInt(line[1:], 10, 64)
30       if err != nil {
31         panic(err)
32       }
33     } else {
34       err := cb(timestamp, line)
35       if err != nil {
36         return err
37       }
38     }
39   }
40   return nil
41 }
```

The program logic should be easy peasy moving forward. Because of Go's strict typing, however, converting the previously read time value in seconds into an integer can try your patience, because the scanner reads it as a string. The `ParseInt()` function from the standard *strconv* package expects the string to be parsed as its parameter (`line[1:]` in line 29 cuts off the first letter – that is, the comment character), as well as the base of the integer number (`10` for a decimal number) and the maximum number of bits (`64`). Back comes a `int64` type value; this is an important detail: Otherwise the lights would go out on January 19, 2038, because by that date the supply of 32-bit integers for the seconds is exhausted.

One use for the `histWalk()` function is shown in Listing 2, which runs an analysis of the user's typing activities, broken down for the days of the week (Figure 4). Since the timestamp (`stamp`) passed to the callback is an integer, the standard `time.Unix()` function called in line 12 easily converts it to the Go internal `time.Time` format for time and date values. The `Weekday()` function called with the converted value then determines the day of the week for the given date. The `int()` wrapper converts the value that exists as a structure in Go's standard `time` package into an integer between `0` (Sunday) and `6` (Saturday).

Listing 2: dow.go

```
01 package main
02
03 import (
04   "fmt"
05   "time"
06 )
07
08 func main() {
09   var countByDoW [7]int
10
11   err := histWalk(func(stamp int64, line string) error {
12     dt := time.Unix(stamp, 0)
13     countByDoW[int(dt.Weekday())]++
14     return nil
15   })
16
17   if err != nil {
18     panic(err)
19   }
20
21   for dow := 0; dow < len(countByDoW); dow++ {
22     dowStr := time.Weekday(dow).String()
23     fmt.Printf("%s: %v\n", dowStr, countByDoW[dow])
24   }
25 }
```

Weekdays 0 to 6

The length 7 array created at the beginning of line 9 in Listing 2 provides count values at positions `0` through `6` for events recorded per individual weekday. The callback function increments these values by one for each incoming command according to the weekday found in the timestamp. An interesting property of a callback function defined inline in Go is that it has access to previously defined local variables, like `countByDoW`, which also remains in scope even after the walking phase, further down in the `for` loop from line 21.

The loop iterates over the index positions `0` to `6` and digs out the counters for the individual weekdays. Now how can an integer value be converted back into a weekday string? The `time` package contains the `Weekday` data type, which defines integer constants from `0` to `6`, which correspond to the weekdays Sunday through Saturday. Furthermore there is a `String()` function that converts the constant values into English weekday strings. Line 22 determines the weekday following this

Listing 3: top3.go

```
01 package main
02
03 import (
04   "fmt"
05   "sort"
06 )
07
08 func main() {
09   cmds := map[string]int{}
10
11   err := histWalk(func(stamp int64, line string) error {
12     cmds[line]++
13     return nil
14   })
15
16   if err != nil {
17     panic(err)
18   }
19
20   type kv struct {
21     Key   string
22     Value int
23   }
24
25   kvs := []kv{}
26   for k, v := range cmds {
27     kvs = append(kvs, kv{k, v})
28   }
29
30   sort.Slice(kvs, func(i, j int) bool {
31     return kvs[i].Value > kvs[j].Value
32   })
33
34   for i := 0; i < 3; i++ {
35     fmt.Printf("%s (%dx)\n", kvs[i].Key, kvs[i].Value)
36   }
37 }
```

Listing 4: The Winning Trio

```
$ ./top3
make (72x)
vi histWalk.go (57x)
vi top3.go (23x)
```

scheme. Line 23 then only needs to output the strings, along with the cumulated counters. Figure 4 shows how to compile the listings to create a binary and a call to it. Running the program reveals that the user seems to do the most typing on Mondays.

Hit of the Week

Listing 3 shows another evaluation of the history data, revealing the three most frequently typed commands. As a data structure for counting identical commands, line 9 creates a hash map that assigns commands as strings to integer counters. The callback function starting in line 11 has access to the data structure and receives from `histWalk()` both a timestamp and the string with the typed command line for each history line that passes by.

More Work Due to Strict Types

At the end of the program run, the winners with the highest counter values are determined. But how do you separate the top three from the rest of the field? Due to their weak typing, scripting languages offer far more convenient methods for sorting a hash map by the values it contains. Go has much stricter typing rules, on the other hand, and requires a certain rigamarole to get the job done. First, line 20 creates a new data structure, a combination of a string named `Key` and an integer named `Value`.

Then the `for` loop starts at line 26 and builds a sortable array from the hash map with the hash structure's keys and values. The `sort.Slice()` function then sorts the array numerically in descending order by the `Value` field (i.e., the integer values). After that, it is easy for the `for` loop from line 34 on to output the top three as the first three elements of the sorted array slice.

Since Listing 3 does not need any extra packages, it is simply compiled with

```
go build top3.go histWalk.go
```

Shortly after this, a binary named `top3` is available, which scans the history file and displays the winning trio of the most frequently typed commands (Listing 4).

With some algorithmic tricks (e.g, by using a heap structure), the work for determining the top *N* from a list could be made even more efficient than through sorting the whole list. However, since the number of hand-typed commands is fairly manageable by computer standards, Listing 3 does not bother to do this.

Split Brain

If you try the presented programs right away, you might wonder why the history you see depends on the terminal window instance where you're currently located.

Bash has the questionable habit of creating separate histories for separate terminal windows opened by the same user. If you type a command in one window, the history in another window will not know about it. But if you shut down the shell in a terminal window (e.g., by closing the window), the running shell sends its history to the `.bash_history` file in your home directory. Every newly started shell then has access to the newly added data moving onward.

If you want to keep the global history up to date, you can use the command sequence shown in Listing 5. If you store it in the `PROMPT_COMMAND` environment variable, the shell will trigger it after each command you type.

The three `history` commands in Listing 5 write the command trail of the current Bash session to the global file (`-a` for "append"), delete the local history (`-c` for "clear"), and load the global history into the local session (`-r` for "reload"). This setting, however, means accessing the hard disk for each command; depending on the number of typed commands, this may result in a loss of speed.

Furthermore, Bash limits the number of commands created in the history to 500 by default. If you want to resort to older commands, set the variables in Listing 6 to the values shown [2].

The first value sets the maximum number of entries Bash can remember in a running session. The second value specifies the maximum number of lines in the global history file. In any case, access and analysis of history data offer the opportunity to discover shell commands that you type too often. This analysis might help you to develop better, as in more efficient, methods. The `histWalk()` function presented here will hopefully animate readers to try out further practical applications. ■■■

Listing 5: Keeping the History Up to Date

```
export PROMPT_COMMAND="history -a; history -c; history -r; $PROMPT_COMMAND"
```

Listing 6: Set Variables

```
export HISTSIZE=100000
export HISTFILESIZE=100000
```

Info

[1] Listings for this article: *ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/243/*

[2] HISTSIZE vs. HISTFILESIZE: *https://stackoverflow.com/questions/19454837/bash-histsize-vs-histfilesize*

Optimize SVG files for websites

Slimming Down

Inkscape creates W3C-compliant SVG files, but they are usually larger than they need to be for the web. We'll show you how to optimize SVG files for faster load times. *By Sirko Kemter*

Graphic images can give a website clarity and visual interest, but the web designer must contend with the tax that heavy graphic images place on bandwidth and load time. Photo images are typically scaled down to a lower resolution (and smaller file size) to run on the web, and for other types of image files, smaller is also almost always better.

One way to reduce the size of web images is to rely on vector graphics whenever possible. The Scalable Vector Graphics (SVG) format was originally developed for use on the Internet. Although SVG typically leads to significant economies of scale when compared to pixel graphics, it has taken decades for web developers to become reasonably comfortable with it.

The most popular tool for vector graphics on Linux is Inkscape. You can certainly save some kilobytes by generating graphics files with Inkscape rather than an alternative raster graphics tool like Gimp. But using Inkscape doesn't mean your files will automatically be optimized for the web. This article highlights some steps you can take to reduce the size of vector graphics image files.

Turning Big into Small

There are two ways to display a simple circle using HTML graphics: either as a path (Listing 1) or as an SVG element (Listing 2).

Two nodes are all you need for the path, but most programs use at least four, and that includes Inkscape. These extra nodes are due to the fact that slight deviations from the original path occur with only two nodes, but this extra level of precision is often unnecessary in the web context.

As you can see, the source code from Listing 1 needs far more characters and accordingly more space. As long as you can work with elements defined in SVG (Listing 2), you should do so.

Clone or Duplicate?

Graphic artists often use a duplicate of an object instead of a clone. The duplicate repeats the complete description of

Listing 1: Circle as Path

```
<path d="M 162.66582,125.05696 A 70.917999,70.917778 0 0 1 91.747818,195.97474 70.917999,70.917778 0
0 1 20.829819,125.05696 70.917999,70.917778 0 0 0 1 91.747818,54.139183 70.917999,70.917778
0 0 1 162.66582,125.05696 Z" style="fill:#000000;stroke:none;stroke-width:7.99999;stroke-linecap:round;
stroke-linejoin:round;stop-color:#000000" id="path10" />
```

Listing 2: Circle per SVG

```
<ellipse ry="70.917778" rx="70.917999" cy="125.05696" cx="91.747818" id="path10" style="fill:#000000;
stroke:none;stroke-width:7.99999;stroke-linecap:round;stroke-linejoin:round;stop-color:#000000" />
```

the object; only the coordinates and the ID change (Listing 3). A clone, on the other hand, uses the use tag with the ID of the original (Listing 4).

The code example from Listing 3 needs 488 characters, whereas the cloned variant from Listing 4 needs only 406 characters – with the same result. The difference might seem relatively small, but this is only a single simple object. In the case of more complex graphics, several kilobytes can quickly accumulate.

Some people might think that a few kilobytes do not matter today, but with many users accessing the Internet via smartphones, traffic is still relatively expensive. Some organizations also pay for data traffic on the server end, and a few kilobytes for a single image can add up to several gigabytes by the end of the month. But regardless of the traffic issues, optimization is a healthy habit for any website. Your visitors benefit from a compact website with faster load times and will be more likely to come back.

You should always remove unneeded nodes from paths, but this can only be done manually. An ideal path looks like the one shown in Listing 5.

Inkscape

The majority of Linux users use Inkscape to create SVG graphics. Many users don't realize that Inkscape inflates graphics with a lot of unnecessary stuff. This starts with unused definitions and continues with the Inkscape and Sodipodi document types and ends with a heap of metadata. The web browser does not need all this information to render the graphics.

If you fill an object with a gradient, a definition will appear in the `<defs>` section at the beginning of the document. If you change the fill, the old definition will remain in the file. You can remove these unused definitions in Inkscape with *File* | *Clean Document*.

Save Correctly

If you use *Save as…* instead of *File* | *Save as…*, you can select a different file format. The output options include different SVG variants, such as Inkscape SVG, plain SVG (and its zipped variants), and optimized SVG.

If you save in Inkscape SVG, everything remains the same. Saving as plain SVG, on the other hand, removes all Inkscape/Sodipodi proprietary elements. If you use optimized SVG, a dialog box with four tabs appears. The last tab reports that Inkscape uses the SVG optimizer Scour [1].

In the first tab, *Options*, you can specify the significant digits for the coordinates (Figure 1). In the code example for the circle, you can see that the coordinates have six digits after the decimal point; however, you will not usually need this kind of accuracy to display a graphic on the web. If you enter a *6* in the *Options* tab, and the number has three digits before the decimal point, it

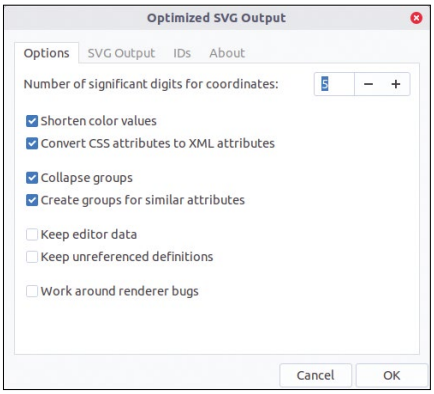


Figure 1: If you save your file as an optimized SVG in Inkscape, this dialog will open.

Listing 3: Duplicate

```
<ellipse
    ry="8.595953"
    rx="8.5959797"
    cy="62.735138"
    cx="29.425798"
    id="path10"
    style="fill:#000000;stroke:none;stroke-width:0.96968;
           stroke-linecap:round;stroke-linejoin:round;stop-c
           olor:#000000" />
<ellipse
    style="fill:#000000;stroke:none;stroke-width:0.96968;
           stroke-linecap:round;stroke-linejoin:round;stop-c
           olor:#000000"
    id="ellipse102"
    cx="49.26012"
    cy="62.735138"
    rx="8.5959797"
    ry="8.595953" />
```

Listing 4: Clone

```
<ellipse
    ry="8.595953"
    rx="8.5959797"
    cy="62.735138"
    cx="29.425798"
    id="path10"
    style="fill:#000000;stroke:none;st
           roke-width:0.96968;stroke-linecap:round;stroke-li
           nejoin:round;stop-color:#000000" />
  <use
    height="100%"
    width="100%"
    transform="translate(21.089236)"
    id="use104"
    xlink:href="#path10"
    y="0"
    x="0" />
```

Listing 5: Optimized Path

```
<path d="M504 256c0 137-111 248-248 248S8 393 8 256 119 8 256 8s248 111 248 248zM212 140v116h-70.9c-10.7
0-16.1 13-8.5 20.5l114.9 114.3c4.7 4.7 12.2 4.7 16.9 0l114.9-114.3c7.6-7.6 2.2-20.5-8.5-20.5H300V140c0-6.
6-5.4-12-12h-64c-6.6 0-12 5.4-12 12z"/>
```

will have three digits after; if you have tow digits before the decimal point, you will have four after.

If you enable the *Shorten color values* option, the short version is used for all color values (i.e., `fill:#f00` instead of `fill:#ff0000`). *Convert CSS attributes to XML attributes* writes color values to the SVG document instead of the CSS attribute. You should not do this, for example, if you are designing icons that need to get their color values from CSS.

The *Collapse Groups* button removes unneeded groupings. Many graphic designers group together all or individual elements of the graphic, which makes the workflow easier. In code, a group is represented by the `g` tag, for example `<g id="g1211"></g>`. This example contains 18 bytes that are not important for correct rendering. Inkscape itself always creates a group for the layers in the document, but this kind of group only makes sense while you are working on the drawing, not when it is displayed.

The *Create groups for similar attributes* option creates groups for objects with the same fill color and outline. The software then writes the values for the attributes to the `g` tag, for example `<g fill="#f00">`. All the objects in this group are therefore black, which removes the need for the corresponding attribute for each object (i.e., a minimum of 10 characters with two objects).

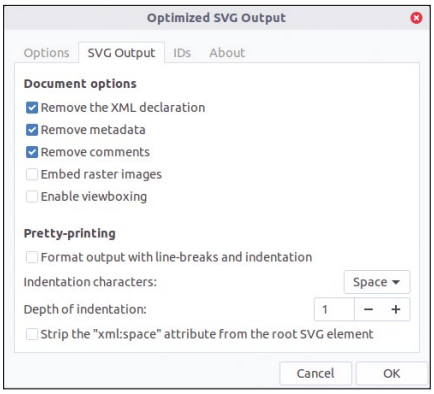


Figure 2: The Optimize dialog with the tab for customized SVG output.

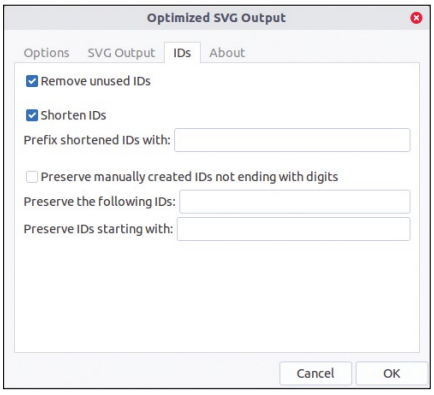


Figure 3: The tab with the ID functions in the Optimize dialog.

The *Keep unreferenced definitions* option keeps the unreferenced definitions. *Work around renderer bugs* tries to work around known rendering bugs in some engines. However, all web browsers render SVG quite well, so you don't really need this switch.

The second tab *SVG Output* (Figure 2) also contains some options. The option *Remove the XML declaration* is not needed to render the graphic. However, it should be kept if the graphic contains text with special characters.

It makes sense to enable the next option *Remove metadata* – there is sometimes quite a large volume of metadata, such as the author and license entries. *Remove comments* deletes comments contained in the document; *Embed raster images* writes existing raster images as Base64 code into the SVG document.

The *Enable viewboxing* checkbox is interesting; you can often see faulty files with it. The viewport is the visible image area and can be compared with a window where all or only part of the view is shown.

The size of the viewport is defined by the attributes `width` and `height`. The dimension for the viewbox is often the size of the document, so you first need to adjust the page size accordingly. To adjust the page size, open the *File* | *Document Properties* dialog and enable *Resize page to content…*. Then click on *Resize page to drawing or selection*.

The *IDs* tab of the Optimized SVG output dialog(Figure 3) lets you remove or shorten unused IDs. In SVG, every object is given an ID, but this ID is often longer than necessary. You usually only save one or two bytes, but the sheer mass of the data can add up.

If you now save the file as SVG, it will be far smaller. By how much depends on the complexity of the graphic: The more complex, the greater the savings – usually around 50 to 65 percent.

Alternatives

Optimizing vector graphics in the Inkscape GUI can take some time. It is also possible to call Inkscape at the command line, although the command-line options are limited to cleaning up definitions.

Inkscape uses the file extension on the command line to determine what to save the file as. SVG stands for Inkscape SVG only.

The Scour command-line tool is a better choice for optimizing an Inkscape file at the command line; Listing 6 shows its syntax.

There are other tools for editing and optimizing SVG files, for example SVGO [2]. However, very few distributions have SVGO in their repositories. To install it anyway, use the Node.js package manager:

```
$ sudo npm install -g svgo
```

SVGO follows a philosophy that is different from Scour, so the call is less complex:

```
$ svgo -i filename.svg
```

The command overwrites the `filename.svg` file with the optimized version. If you want to save it in a separate file, use the `-o new_filename.svg` switch.

SVGO performs some operations that Scour does not offer. For instance, SVGO compares path values and converts duplicates into clones. You can enable this option with the `--reusePaths` option.

SVGO also comes with a function for recursive operations and can therefore be applied to directories. In Scour, you would have to write a small script for recursion. In any case, SVGO is the more modern tool. A plugin exists for using SVGO directly from within Inkscape [3], but currently it does not work.

Listing 6: Using Scour

```
$ scour input_filename.svg output_filename.svg --create-groups
--no-renderer-workaround --remove-descriptive-elements
--enable-comment-stripping --enable-viewboxing --no-line-breaks --strip-xml-space
--enable-id-stripping --shorten-ids
```

Another tool for cleaning SVG files is svgcleaner [4]. Svgcleaner has a graphical user interface called SVG Cleaner [5]. This tool is also rarely found in the repositories of the popular distributions. Since the GUI is written in Qt, you have to compile it yourself.

The developers implemented svgcleaner in Rust, so you can install it using the Rust package manager Cargo:

```
cargo install svgcleaner
```

Svgcleaner takes an even more radical approach than SVGO and creates even smaller files. If you work with CSSs in SVG, make sure you avoid the svgcleaner program, because it also removes CSS attributes.

Since Rust writes the binary file to a separate subdirectory of its home folder, you first need to change the directory, and when you get there, call svgcleaner:

```
$ cd /home/user_name/.cargo/bin/
$ ./svgcleaner input_filename.svg
output_filename.svg
```

Conclusions

Even if you're a vector graphics beginner, the tools and techniques described in this article will help you get started with optimizing your SVG web files. Smaller files will mean faster load times for your website visitors – and could also lead to less traffic and lower costs for you and your customers. However, it is important to note that all the tools described in this article are not designed to work with the next generation W3 vector-graphics standard SVG 2.0, which is currently at the draft stage. ■■■

Info

[1] Scour: *https://github.com/scour-project/scour*

[2] SVGO: *https://github.com/svg/svgo*

[3] svgo plugin: *https://github.com/konsumer/inkscape-svgo*

[4] svgcleaner: *https://github.com/RazrFalcon/svgcleaner*

[5] SVG Cleaner: *https://github.com/RazrFalcon/svgcleaner-gui*

Author

Sirko Kemter has been involved with Inkscape since the program began and has written a book about working with Inkscape. In his spare time, he creates graphics for Openclipart and various open source projects. Because he lives in Southeast Asia, and people there tend to use mobile devices, he has been working for some time on optimizing websites to reduce their size and load times.

Developing Tiny Core Linux extensions

TINY EXTENSIONS

Tiny Core Linux does not boast a big repository. Sooner or later, you'll need to create your own extensions to get the most out of Tiny Core. This article shows you how. *By Rubén Llorente*

Tiny Core Linux [1] offers an extremely lightweight Linux distribution whose default install gives you only the bare essentials. With no web browser, email client, or office suite, Tiny Core Linux gives you just enough to run a graphical session, making it ideal for older computers.

While Tiny Core offers some software in its repository, if you intend to run a specialized application, such as a medical image viewer like the one used at my job, you will need to create your own package.

In this article, I will show how to create your own packages, or extensions as they are called in Tiny Core Linux. Because the general methodology for packaging an application does not change much across distributions, this article also will give you a general overview of what it takes to create a software package.

Understanding Extensions

Packages from popular distributions are usually compressed archives that contain the software, some metadata information, and an install script. When you install a *txz* package using `installpkg` in Slackware, for example, the package manager decompresses *txz*, places its contents directly in the operating system root (/), and then runs any install script contained in the package (Figure 1).

Tiny Core Linux is not a regular distribution. Instead of residing on the hard disk, the operating system's core components are loaded into Random Access Memory (RAM) upon boot (Figure 2). The operating system root (/) exists in a virtual filesystem, which lives in RAM only. In order to install a traditional package in Tiny Core, you must decompress its contents and then load them completely into RAM, which wastes both time and RAM.

As an alternative to traditional packages, Tiny Core Linux uses extensions. Each extension is a Squashfs filesystem image instead of a compressed archive (see the box "What is Squashfs"). Tiny Core loads an extension by mounting this filesystem in a folder in `/tmp/tcloot`. Then, it creates symbolic links from its contents to the designated places in the operating system's file hierarchy. In practice, this means that the distribution's core runs fully in RAM, while extensions are read directly from the CD-ROM or USB used to boot the operating system. As a result, extensions load quickly and do not take up much RAM.

Install scripts are also supported. These are useful for a number of reasons. For example, if an extension contains a system daemon, a script will be necessary to create a system user for this daemon. Install scripts run when the extension is loaded and perform this sort of task. Any script

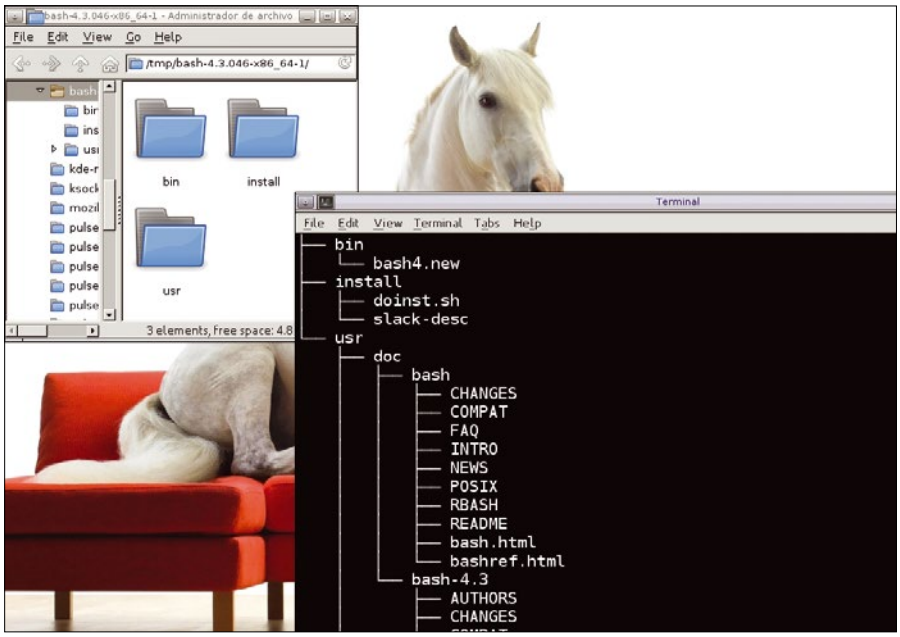


Figure 1: A decompressed Slackware package: During installation, the package manager decompresses the contents of the package and places them directly under `/`, with the exception of the `install` folder, which contains the package's metadata and a script that is run during installation.

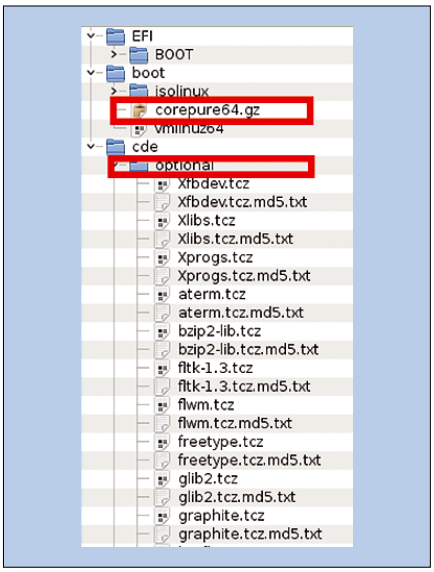


Figure 2: A typical Tiny Core ISO image: The operating system is stored in `corepure64.gz`, which gets decompressed into RAM on boot. The `optional` directory contains extensions external to the core.

named after the extension and placed in `/usr/local/tce.installed` (within the package) will be executed at load time.

Setting the Environment

You first need to install Tiny Core Linux in order to create a proper Tiny Core extension. You can either download a CD-ROM image from the website [1] and burn it, or you can install on a USB stick (see *Linux Magazine*, issue 203, for instructions [2]).

After booting the Live system, you need to load the necessary development extensions. In order to get the essential components for compiling and packaging, install `compiletc`, `squashfs-tools`, and `submitqc`:

```
$ tce-load -wi compiletc ➥
  squashfs-tools submitqc
```

This will download and install a compiler, `mksquashfs`, and a tool for ensuring that extensions are formatted properly. If you intend to compile software with additional dependencies, as is usually the case, you will need to use `tce-load` to download them as well.

What Is Squashfs?

Squashfs is a read-only, compressed filesystem. Files are added to it upon filesystem creation, and its contents cannot be modified while the filesystem is mounted and in use.

Squashfs is a popular tool for Live operating systems, which are usually stored on read-only media (such as DVDs) or devices whose lifespan is reduced by write operations (USB pen drives). For example, Porteus packages, called modules, are compressed Squashfs filesystem images, as are SLAX modules.

Creating Your First Extension

If you are creating an extension for personal or internal use, you can use an existing package from another distribution or official binaries from the upstream developers (if available) and then shoehorn them into a Squashfs filesystem. This is quick but inefficient. For this article, I will repackage AdoptOpenJDK 12 as an example. The *AdoptOpenJDK-12* extension in the official Tiny Core Linux repository was packaged using a similar method. Keep in mind that this software is only supported on AMD64, so you will need to use the x86 Pure 64 version of Tiny Core Linux [3] in order to replicate my instructions.

First, download the tarball you intend to package and place it in `/tmp` (in this example, the official AdoptOpenJDK release [4]).

Next, decompress the AdoptOpenJDK binary distribution. Create a folder to serve as the Squashfs filesystem root. Since I want to install the AdoptOpenJDK binaries under `usr/local/lib`, I will create this directory in the extension tree and decompress the official binaries there (as shown in Listing 1).

You may add as many custom files as needed to your extension. For example, users need to have certain shell variables set in order to make use of an installed Java Development Kit (JDK). Most distributions place scripts in `/etc/profile.d` in

Listing 1: Decompressing AdoptOpenJDK

```
$ cd /tmp
$ mkdir -p package-adoptopenjdk-12/usr/local/lib/adoptopenjdk-12
$ cd package-adoptopenjdk-12/usr/local/lib/adoptopenjdk-12
$ tar -xvzf /tmp/OpenJDK12U-jdk_x64_linux_hotspot_12.0.2_10.tar.gz
--strip-components=1
```

order to set these variables for login shells, but Tiny Core's extensions are not supposed to access `/etc/profile.d` directly. Instead, create the folder `usr/local/etc/profile.d` within the extension file tree as follows:

```
$ cd /tmp
$ mkdir -p package-adoptopenjdk-12/usr/local/etc/profile.d
```

Then place the script shown in Listing 2 inside this folder.

Scripts that must run at load time are placed in `usr/local/tce.installed`. Create this folder with:

```
$ mkdir -p package-adoptopenjdk-12/usr/local/tce.installed
```

Then place the script from Listing 3 in the newly created folder. The script tells the extension to generate a symbolic link from `/usr/local/etc/profile.d/adoptopenjdk.sh` to `/etc/profile.d/adoptopenjdk.sh` at load time.

Lines 14 and 15 in Listing 3 serve as a hack to make the dynamic linker available to the extension (see the "The Lost Dynamic Linker" box).

Next, you need to make the package tree's permissions consistent. System directories such as `/usr/local` should belong to `root:root` and have a permission mode of `755`. Files that are intended to be executable should be set to `755`, while files that are not executable should be set to `644`. As an exception, `usr/local/tce.installed` must belong to `root:staff` and be set to `775`. The scripts located in this folder must belong to `tc:staff` instead. Listing 4 shows how to make permissions consistent.

Finally, make an extension out of the package tree. To create a Squashfs filesystem with the contents of `/tmp/package-adoptopenjdk-12` and place it under `/tmp`, use the following command:

```
$ mksquashfs /tmp/package-adoptopenjdk-12 /tmp/adoptopenjdk-12.tcz
```

Packaging AppImages

The method shown for packaging AdoptOpenJDK can be adapted for AppImage files. The only difference is that you won't be able to unpackage them with `tar`, since their contents are encapsulated as filesystem images instead. AppImage files can be extracted using the `--appimage-extract` switch. For example, in order to extract *Nextcloud-Client* after downloading it, run:

```
$ chmod +x Nextcloud-2.6.4-x86_64.AppImage
```

Listing 2: adoptopenjdk.sh

```
01 #!/bin/sh
02
03 # Place this script in usr/local/etc/profile.d
04 # It will be copied to /etc/profile.d at load time.
05 # This script sets variables needed by the JDK
06 # for login shells.
07
08 export JAVA_HOME=/usr/local/lib/adoptopenjdk-12
09 export MANPATH="${MANPATH}:${JAVA_HOME}/man"
10 export PATH="${PATH}:${JAVA_HOME}/bin"
```

Listing 3: adoptopenjdk-12

```
01 #!/bin/sh
02
03 # Place this script in usr/local/tce.installed
04 # A script placed in this folder gets executed
05 # at load time. It must be named after the
06 # extension.
07
08 # This script makes adoptopenjdk.sh available at
09 # /etc/profile.d and makes the dynamic linker
10 # available at /lib64. This is needed to run
11 # many prebuilt binaries in Tiny Core x86_64.
12
13 [ -L /etc/profile.d/adoptopenjdk.sh ] || [ -f /etc/profile.d/adoptopenjdk.sh ] ||
     ln -s /usr/local/etc/profile.d/adoptopenjdk.sh /etc/profile.d/adoptopenjdk.sh
14 [ -d /lib64 ] || mkdir -m 755 /lib64
15 [ -L /lib64/ld-linux-x86-64.so.2  ] || [ -f /lib64/ld-linux-x86-64.so.2  ] ||
     ln -s /lib/ld-linux-x86-64.so.2 /lib64/ld-linux-x86-64.so.2
```

The Lost Dynamic Linker

Most Linux programs need shared libraries in order to work. Shared libraries contain code that is designed to be used by different applications. When you write a program that uses standard output, for example, you do not write the standard output code. Instead, you use a shared library that already contains that code rather than doing the work yourself.

According to the Filesystem Hierarchy Standard (FHS) [5], shared libraries should be located in `/lib` and `/usr/lib`. Operating system installs that support both 32- and 64-bit programs should put the 32-bit shared libraries in `/lib` and `/usr/lib` and the 64_bit libraries in `/lib64` and `/usr/lib64`.

The dynamic linker is what allows a program to load shared libraries. In Tiny Core x86_64, the dynamic linker is a file called `ld-linux-x86-64.so.2`, which resides in `/lib`. In theory, installs that only support 64-bit programs don't need a `/lib64` directory, so Tiny Core's approach is correct. However, many official binaries distributed as tarballs (such as AdoptOpenJDK) expect to find a dynamic linker under `/lib64` and will fail to run if that directory does not exist.

This is the reason for lines 14 and 15 in Listing 3. This code creates `/lib64` in Tiny Core Linux's directory structure if it does not exist already and places a symlink to the default dynamic linker there. With this hack, programs that expect a dynamic linker to exist under `/lib64` will find it in that folder.

Listing 4: Making Permissions Consistent

```
$ sudo sh
# cd /tmp/package-adoptopenjdk-12
# chown -R root:root .
# chown tc:staff usr/local/tce.installed/adoptopenjdk-12
# chmod 755 usr/local/etc/profile.d/adoptopenjdk.sh
# chmod 755 usr/local/tce.installed/adoptopenjdk-12
# find -L . \
  \( -perm 777 -o -perm 775 -o -perm 750 -o -perm 711 -o -perm 555 \
    -o -perm 511 \) -exec chmod 755 {} \; -o \
  \( -perm 666 -o -perm 664 -o -perm 640 -o -perm 600 -o -perm 444 \
    -o -perm 440 -o -perm 400 \) -exec chmod 644 {} \;
# chown root:staff usr/local/tce.installed
# chmod 775 usr/local/tce.installed
```

```
$ ./Nextcloud-2.6.4-x86_64.AppImage
  --appimage-extract
```

Remember: In an AMD64 system, you will need to ensure that `/lib64/ld-linux-x86-64.so.2`, the dynamic linker, is available. Otherwise, AppImages won't work. The best way to do this is to create `/lib64` if it does not exist and then make a symlink from `/lib/ld-linux-x86-64.so.2` to `/lib64/ld-linux-x86-64.so.2`.

Packaging from Source Code

The procedure for packaging a piece of software from its source code is not complex. Instead of taking a precompiled build and decompressing it into the extension file tree, you just compile the software directly into the extension file tree:

```
$ cd source_code
$ ./configure --prefix=/usr/local
$ make
$ sudo make install \
DESTDIR=/tmp/package-extension
```

The `--prefix` switch is required to configure the software to operate under `/usr/local` once installed. The `make install` command is passed a `DESTDIR` parameter to get the software installed directly into the extension folder (rather than the actual operating system doing the building!).

Making Your Extension Official

If you don't intend to share your extensions, you are done. However, if you intend to submit your extension to an official repository, you still have work to do.

Every submission to the official repository needs a file that contains the extension's checksum, a list of the

files contained in the extension, a generic information file, and a list of dependencies.

To generate the extension's checksum, use the following commands:

```
$ cd /tmp
$ md5sum adoptopenjdk-12.tcz >> ➥
  adoptopenjdk-12.tcz.md5.txt
```

Then, you can generate a valid list of the extension's contents with the following commands:

```
$ cd /tmp/package-adoptopenjdk-12
$ find * \! -type d >> ➥
  /tmp/adoptopenjdk-12.tcz.list
$ sort -o /tmp/adoptopenjdk-12.tcz.list ➥
  /tmp/adoptopenjdk-12.tcz.list
```

In order to generate a valid information file, you should download an existing file from the official repository and use it as a template [6].

For the list of dependencies, check the documentation of the software you are packaging. You can also use `ldd` to obtain a list of the libraries a binary uses, which is a great starting point if the documentation is unhelpful:

```
$ find * -type f | ➥
xargs file | grep ELF | ➥
  cut -f 1 -d \: | xargs ldd
```

You should place the list of dependencies in a file with the extension `*.tcz.dep` (e.g., `adoptopenjdk-12.tcz.dep`). Check the official repository for examples.

In order to ensure your dependency list is correct, boot a Tiny Core Linux instance with the `base` and `norestore` cheat codes. Then install the dependencies listed in the `*.tcz.dep` file using `tce-load`. Try to load and use your new extension. If everything works, your list of dependencies is correct.

The official policy for extensions is that they should not touch anything in the filesystem structure's higher levels directly. An extension's files should be placed in `/usr/local`. If you need the extension to make any modification elsewhere in the filesystem, you should use a `tce.installed` script, as shown in this article.

You will also need to strip binaries within the extension. This is easily accomplished with the `strip` command:

```
$ find * -type f | ➥
xargs file | grep ELF | ➥
  cut -f 1 -d \: | xargs sudo strip
```

In order to find out if extensions are fully compliant, Tiny Core Linux offers a *submitqc* extension. Using this extension is as easy as placing the extension, the `*.info` file, the `*.dep` file, and the `*.md5.txt` file in the same folder. Then open a terminal in the folder and run:

```
$ submitqtc --libs
```

The program will generate a log and report any issues it encounters.

If you are packaging graphical applications, it may be a good idea to generate a freedesktop-compliant entry, such as the one shown in Figure 3. This will allow your extension to appear in the desktop environments' menus and in the Tiny Core Linux's default Wbar (Figure 4).

Finally, remember to include the software license! The software that governs the distribution and use of your extension should be installed in `/usr/local/share/doc/$extension` (where *$extension* is the name of your extension).

Conclusion

Tiny Core extensions have a simpler structure than Debian packages, but they accomplish their task. These extensions allow you to load custom software into a Live operating system without wasting RAM or time.

Creating your own extension is easier than it seems. It is worth the effort if only for the experience of being a repository package maintainer. I found that generating an extension is very similar to creating a Slackware package. Despite the formatting differences, the process is pretty much the same: Compile or place the program into a folder and then encapsulate that directory. The biggest difference is the way install scripts are managed.

If you are interested in contributing to the official repository, sign up at the Tiny Core Linux forum [7]. The Tiny Core Linux community keeps a database of build scripts you can consult if you want to see how other users are creating their extensions. To see some of my extensions (official and unofficial), visit the Software directory of my Gopher server [8].

Submissions to the Tiny Core Linux repository are welcome at `tcesubmit@gmail.com`. ■■■

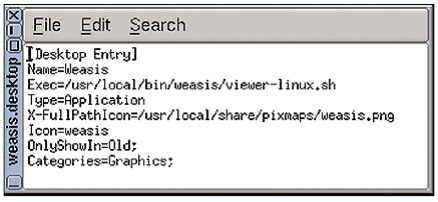


Figure 3: Placing `weasis.desktop` in `/usr/local/share/applications` makes the extension available in the desktop environment's menus.

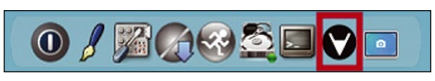


Figure 4: My custom Weasis extension (a medical image viewer) shows up in the Wbar.

Info

[1] Tiny Core Linux: *http://www.tinycorelinux.net*

[2] "Micro Distros: The Tiniest Linux You Can Get" by Mike Saunders, *Linux Magazine*, issue 203, October 2017, *https://www.linux-magazine.com/Issues/2017/203/Tiny-Distros*

[3] x86 Pure 64: *http://www.tinycorelinux.net/11.x/x86_64/*

[4] AdoptOpenJDK 12 with Hotspot VM: *https://github.com/AdoptOpenJDK/openjdk12-binaries/releases/download/jdk-12.0.2%2B10/OpenJDK12U-jdk_x64_linux_hotspot_12.0.2_10.tar.gz*

[5] FHS 3.0: *https://refspecs.linuxfoundation.org/FHS_3.0/fhs-3.0.txt*

[6] Example info file for AdoptOpenJDK 12: *http://www.tinycorelinux.net/11.x/x86_64/tcz/adoptopenjdk-12.tcz.info*

[7] Tiny Core Linux Forum: *http://forum.tinycorelinux.net/*

[8] Extensions on Gopher server: *gopher://gopher.operationalsecurity.es/1/Software/TCE/*

Author

Rubén Llorente is a mechanical engineer, whose job is to ensure that the security measures of the IT infrastructure of a small clinic are both law compliant and safe. In addition, he is an OpenBSD enthusiast and a weapons collector.

MakerSpace

Visual Studio Code for programming a Raspberry Pi

Visual Cues

Professional and casual Python developers save time and keystrokes in a Visual Studio Code development environment for remote coding of a Raspberry Pi. *By Sean D. Conway*

Writing software code is heavy work, and professional code writers leverage software tools to create an effective environment that allows developers to work on code without affecting the users or breaking anything in the live environment.

Development environments are typically set up on a local machine or server that reflects the target production environment and contains the code being developed or modified. An integrated development environment (IDE) is a software application programming tool, deployed in a development environment to assist developers in maximizing their productivity and efficiency.

The IDE needs to be compatible with the programming language of interest and typically contains source code editors, a debugger, a compiler, and designers that can all be accessed through a single interface. Combining all of the development tools into one software suite helps the coder work on multiple tasks through one interface.

The repetitive commands used to write, run, and change code can be optimized in an IDE. Microsoft Visual Studio Code (VS Code) [1] is a tool for professional code developers (but see the "IDEs for All" box) that is usually used on standard Windows, Linux, and macOS machines; however, it can also be deployed to help make the life of a Raspberry Pi programmer easier.

Environmental Decisions

VS Code is a streamlined code editor with support for development operations like debugging, task running, and version control. Although the Raspberry Pi has developed through the years into a fairly powerful single-board computer, it is probably not appropriate to run a full development environment *and* run the Python application being developed all on the same Rasp Pi

VS Code is an editor first and foremost. Instead of installing it on a Raspberry Pi with a keyboard, mouse, and video screen attached [2], you can deploy the VS Code tools for remote debugging. For this exercise, I installed VS Code on a workstation running Ubuntu 20.04; then, VS Code installed its headless server component designed for the ARM7 processor on the Raspberry Pi (Figure 1). This installation setup allows you to write and modify code from a local workstation and run and test the code on a Raspberry Pi (see the "Materials" box).

Preparation

The first exercise in creating a development environment with VS Code is to es-

IDEs for All

If you don't consider yourself a professional coder, why invest in an IDE? Maybe you're a code resurrectionist, or Frankenstein programmer.

Resurrectionists were paid body snatchers who exhumed the bodies of the recently dead for anatomists in the United Kingdom during the 18th and 19th centuries. In Mary Shelley's classic horror novel *Frankenstein; or, The Modern Prometheus*, Dr. Frankenstein stitches together an assortment of stolen body parts and attempts to imbue life into his creation.

Likewise, code resurrectionists look through Python code to find pieces that they can bring back to life by combining them with other pieces of code to make something useful. The cycle is similar to the instructions "wash, rinse, and repeat" found on a shampoo bottle: Find code snippets, combine them, modify them, write code when desperate, deploy the mix, and test. An IDE is the perfect tool for this programming cycle.

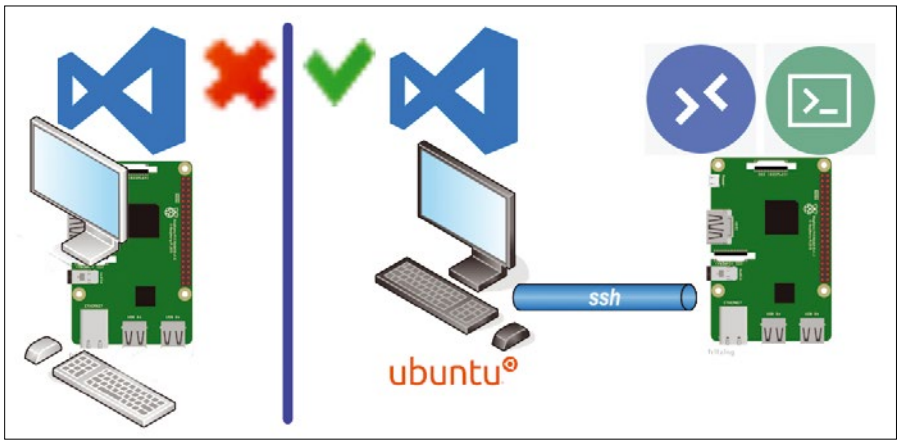


Figure 1: Instead of installing VS Code on the Raspberry Pi, I opted for a workstation installation with a connection to the Pi.

tablish an SSH connection from the Ubuntu workstation to the Raspberry Pi and configure it so that no password is required.

To generate a public and private public key infrastructure (PKI) key pair and transfer the public key to the Raspberry Pi host (Figure 2), issue the following commands from a command-line interface (CLI) on the workstation:

```
ssh-keygen <optional passphrase>
ssh-copy-id -i ~/.ssh/id_rsa.pub ➥
  pi@<Pi IP Address>
```

Being security conscious, best practice is to include a passphrase when prompted. You should retain the file defaults, unless you have the knowledge to customize the configuration.

The command

```
ssh-copy-id -i ~/.ssh/<mykey> ➥
  <user>@<host>
```

logs in to the Raspberry Pi, copies the public key file, and configures the key in the `authorized_keys` file to grant access. The copying may ask for a password or other authentication to connect with the Pi.

After the SSH configuration, the first time you log in requires you to supply a password. After that, SSH connections should be made without password prompting. You will know you are successful when you enter

```
ssh pi@<Pi IP Address>
```

and it drops you into a remote prompt without the need for a password.

Now that you have a Raspberry Pi SSH connection working without password prompting, you need to tailor the Pi for a development environment. Some additional software needs to be added so that it is prepared to support the changes that VS Code is going to make to establish a remote connection to the Pi. The following commands entered from the CLI on the Raspberry Pi adds the needed software:

```
sudo apt update
sudo apt upgrade
sudo apt install ➥
  git ➥
  python-pip ➥
  python3-pip
```

The `update` and `upgrade` commands should always precede any Raspberry Pi software install requests. The commands refresh the environment and prepare it for the new installs.

The `apt install` command loads the Git distributed version control system and `pip`, the package manager for Python packages for the Python 2 and 3 environments. Packages contain the files needed for a module, which is a Python code library. If VS Code wants to install Python software, it is going to call `pip` to make it happen.

Materials

- Workstation: 64-bit Ubuntu 20.04.1 LTS
- Visual Studio Code: version 1.47.3
- Pi3B+, 8GB SD card
- Raspberry Pi OS (previously called Raspbian) *2020-05-27-raspios-buster-lite*

Host Workstation Installation

On the Ubuntu workstation, you should install VS Code from the Ubuntu Software Center. For Ubuntu 20.04, the Software Center will list two versions of VS Code: Visual Studio Code and Visual Studio Code Insiders. The description provided in the window for both are identical, so what is the difference?

Visual Studio Code is the stable version, whereas Visual Studio Code Insiders is the daily release with the most recent code pushes. The Insider builds have new features, bug fixes, and other recently closed issues. If you like working on the edge of software development, consider a side-by-side install (i.e., Insiders installed next to the stable

```
sconway@ubuntu20:~$ ssh-keygen
Generating public/private rsa key pair.
Enter file in which to save the key (/home/sconway/.ssh/id_rsa):
Enter passphrase (empty for no passphrase):
Enter same passphrase again:
Your identification has been saved in /home/sconway/.ssh/id_rsa
Your public key has been saved in /home/sconway/.ssh/id_rsa.pub
The key fingerprint is:
SHA256:LZ+y9eJvpSZxkALDZwCTYPINYnpf27cRfXtsSPs0+BU sconway@ubuntu20
The key's randomart image is:
+---[RSA 3072]----+
|o.+.o+..         |
|o= o..+ o  .     |
|. o . .=  ... oE |
| . . . o..o. o.=.|
|    . . S.+. .+.*|
|         +.+. o=o|
|        . =o o ..|
|         +o.+    |
|        ...*o    |
+----[SHA256]-----+
```

Figure 2: Output from an SSH key generation command, retaining default file names.

build), so you can use either version independently.

If using the latest releases of VS Code appeals to you, you might want to consider removing the *Remote-SSH* extension and replacing it with the *Remote-SSH (Nightly)* extension. The *Remote-SSH* extension lets you use any remote machine with an SSH server as your development environment. The nightly build version of the extension provides the developer early feedback and testing. This extension works best with VS Code Insiders. You must uninstall the stable version of the extension before using the nightly version. Be aware that bleeding edge builds may lead to the occasional broken build.

After firing up VS Code, you will see a welcome screen. Typically, I would start by exploring all the options of the VS Code interface, but I will not use that approach for the moment and pick it up later when VS Code is connected to the Raspberry Pi. For now, I will consider how to configure the application environment for Python.

Some changes are needed in the VS Code installation before proceeding. To begin, at the bottom of the left-hand vertical Activity Bar, select Extensions (the Tetris, or falling block, icon). Extensions add features to VS Code, and you can install those you need or uninstall those you don't require. Recall that the goal is to create a development environment for Python coding on the Raspberry Pi.

In the *Extensions Search* field, search for extensions that have "python" in the name, and click the *Install* button for `ms-python.python` to install the Microsoft *Python* extension (Figure 3). For a development environment, you should add support for Python code formatting (discussed in greater detail later) by installing the *Python-autopep8* (`himanoa.python-autopep8`) extension.

Recall that you did some configuration work earlier on the Raspberry Pi to enable SSH, so you should add that feature to VS Code by installing the *Remote-SSH* (`ms-vscode-remote-ssh`) extension.

Remote Pi Installation

VS Code is now ready to reach out and connect to the Raspberry Pi. The remote connection from the Ubuntu workstation to the Pi must be configured to support

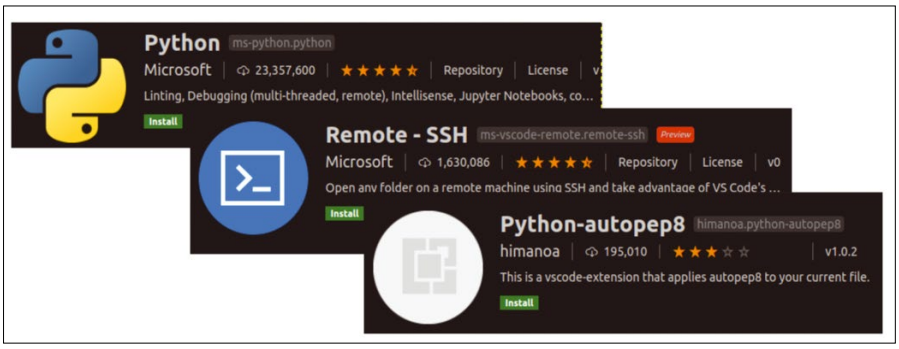


Figure 3: The VS Code extension trifecta for Python support.

the SSH connection without a password. From the VS Code console, press F1; in the search field, enter *remote* and select *Remote-SSH: Add New SSH Host*.

The prompt displays the standard command format `ssh hello@microsoft.com` to make an SSH connection. Enter the connection details for the Raspberry Pi. Now, press the F1 key again and select *Remote-SSH: Connect to Host*; the entry you made for the Pi connection is displayed, so select that item.

If all goes well, VS Code opens another console. At the bottom of the screen it displays the connection details and indicates that it is busy using the connection to bring the Raspberry Pi into the VS Code fold (Figure 4). When complete, the console will remain, and VS Code will be ready for you to program on your Pi (Figure 5).

The Suspended Tour

Now that VS Code is installed on the Raspberry Pi development environment,

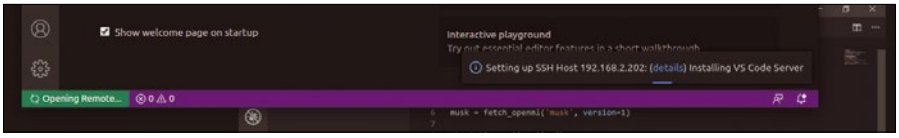


Figure 4: VS Code sets up the Raspberry Pi to communicate with the host remotely over SSH.

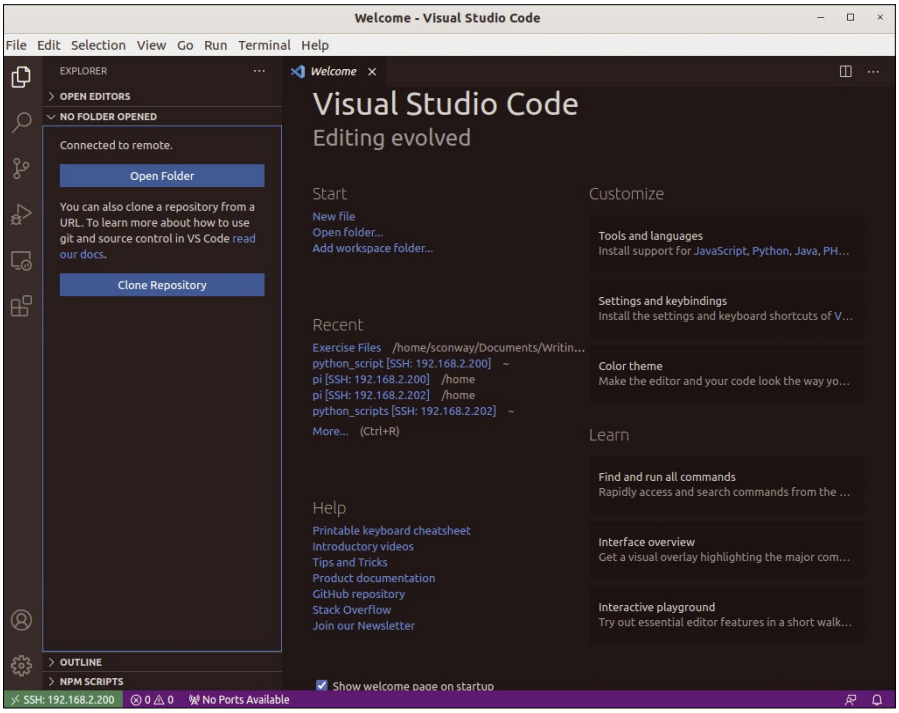


Figure 5: The VS Code Welcome page. This article is based on version 47. At the time of publication, the software had advanced to version 51. The instructions are still valid for the Raspberry Pi VSC install, but the "Start" page has changed in the most recent version.

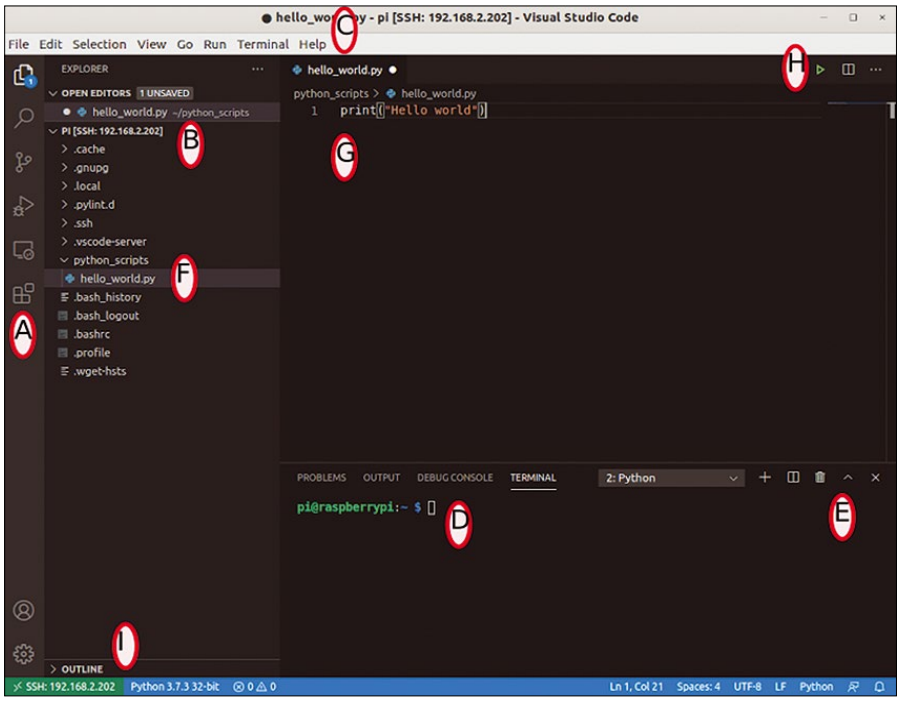


Figure 6: VS Code workspace: (A) Activity Bar, (B) Sidebar, (C) Menubar, (D, E) Panels, (F) Explorer view, (G) Editor workspace, (H) Run arrow, (I) Outline view.

I'll review the interface. Instead of describing the interface options (Figure 6), I will introduce them by performing some tasks to condition the Pi development environment.

In the interface, the Activity Bar (A), the vertical bar on the left side of the screen, contains the icons for the various views (from the top): Explorer, Search, Source Control, Run, Remote Explorer, and Extensions. From the Explorer view of the Welcome page, select *Open Folder*, choose the directory `/home/pi`, and press the *OK* button.

VS Code opens the folder and creates a panel in the workspace (B), which is basically just a collection of related panels. The Explorer view displays the files and folders for the current project workspace, along with any currently open editors.

The left panel of the workspace is a list of the Raspberry Pi's home directory. It also shows how many files have been modified and not yet saved. The `.vscode-server` directory is where VS Code for the remote Pi install was created.

From the main menubar (C), select *Terminal* | *New Terminal* to open a panel (D) at the bottom of the workspace that displays a terminal window for the Raspberry Pi. The plus (+) icon in the terminal panel (E) enables you to open additional terminal windows, the drop-down lets you choose between the named terminal windows, and the trash can icon closes the terminal.

From the Explorer view, create a new folder named `python_scripts` from the icon options and then create a file named `hello_world.py` (F). An Editor panel opens in the workspace (G), where you can modify, add, and delete file content. In the editor, add the line

```
print("Hello World")
```

to create a simple line of Python code that will print *Hello World* when the script is run. Use the *Save* icon on the left in the Open Editors line to save the file.

The `.py` extension has significance for VS Code. The extension is the trigger for the application to customize the environment for the programming language – in this example, Python.

After saving the file, VS Code opens a notification message about recommended extensions for Python (Figure 7). The application indicates that it could use a little support on the Raspberry Pi for Python programming. Ignore the notification message by closing the message window. Instead of letting VS Code do the preparation, you can install the recommended Python extensions manually.

From the Activity Bar, select the Extensions view, which lists all of the installed extensions. This view also shows you which extensions need to be updated and lets you update them. The workspace on the left will indicate that no extensions are found on the *SSH: < IP Address >* connection, because the extensions that were installed to this point were all on the Ubuntu VS Code workstation.

The goal has been to create a Python development environment for the Raspberry Pi. In the Extensions view, enter *python* in the *Search Extensions in Marketplace* field. The search results will display Python extensions that can be installed to the SSH connection. Select the *Python* (Microsoft) extension to install (Figure 8). Recall that you installed this same extension on the VS Code host workstation.

After a brief installation interlude of flashing dots, a *Reload Required* button appears. Now you can reload VS Code by pressing the button to enable this extension on the Raspberry Pi. The screen refreshes the Extensions workspace panel on the left and indicates that the Python extension has been installed.

To see what this manually loaded VS Code recommended extension does for the development environment, select the *Explorer* view from the Activity Bar and open the Python `hello_world.py` file created earlier. If any terminal windows are open in the lower portion of the workspace, select the trash can to close them. VS Code is constantly looking to tailor the environment, so you might

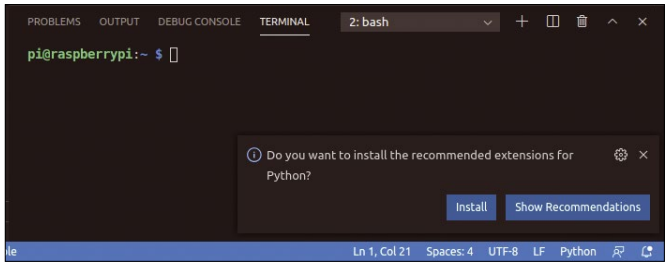


Figure 7: Although you installed Python packages on the Ubuntu host, the Raspberry Pi needs Python support, too.

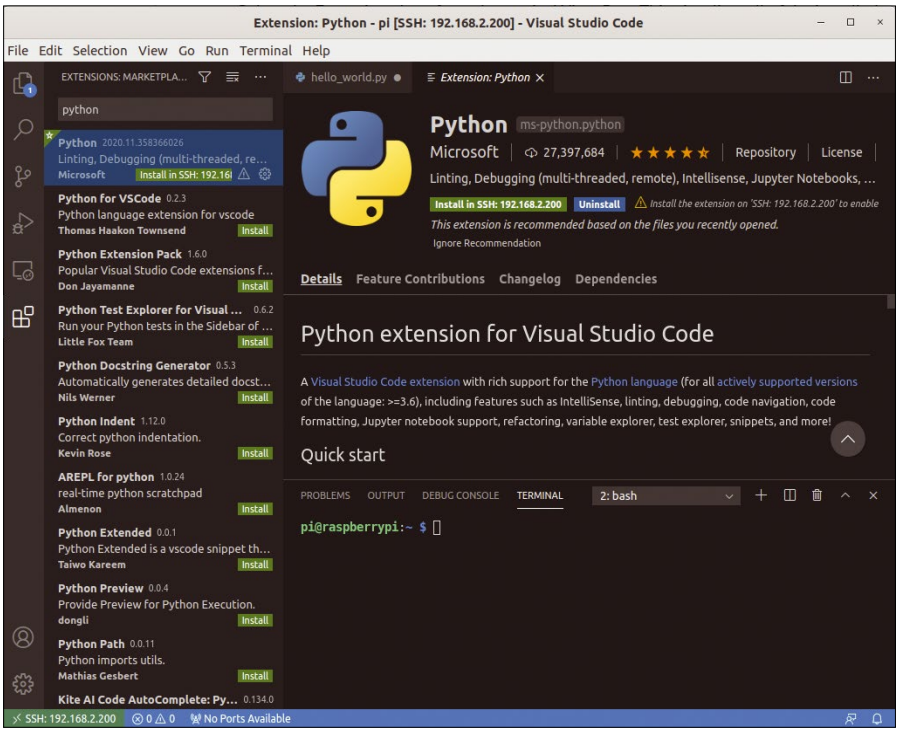


Figure 8: The results of a search for Python extensions available for installation.

also have a notification message (discussed later). At some point, the environment will be stable and the messages will stop. Ignore them for now, but don't close the message.

In the upper right-hand corner of the Editor workspace (Figure 6, H), select the green arrow icon to run the Python file. A terminal panel then opens in the lower right workspace and the Python script is executed, displaying the *Hello World* output.

In addition to the extensions notification message received earlier, you will also get a notice that `autopep8` (which formats Python code to conform to PEP 8 coding conventions [3]) is missing from the development environment on the Raspberry Pi. VS Code attempts to help you by indicating missing software components that aid Python code development.

Go ahead and use the *Install* button in the notification message to install the *pyline* Python linter package. Earlier in the tutorial, when preparing the Raspberry Pi environment, you installed Pip, which VS Code uses to install the required Python packages on the Pi. In the terminal panel, feedback is displayed as the package is installed. For now, close the notification about `autopep8`. After some additional instruction you will be better prepared to deal with that notification message.

So far, the VS Code Remote SSH connection (Figure 6, I) to the Raspberry Pi has provided a graphical file and folder listing, an editor for modifying code, the ability to add or delete Python scripts, and a terminal panel that allows you to run the code. It also informs you when it discovers missing Python packages. VS Code also lets you install Python extensions and packages. As a huckster on a television commercial would say, "but wait, there is more!"

Tour Continued

In the Editor panel (Figure 6, G) use the mouse to select all lines of code, then right-click for the command context menu and choose *Run Selection/Line in Python Terminal*. A Python terminal opens in the bottom panel and runs the code selection. Recall the Python programming language itself has its own terminal or console. VS Code lets you to take a portion of the code from the editor and run it in the Python terminal. Click the trash can in the terminal panel when you are done examining it.

One of the difficulties experienced by professional and casual code writers alike is code formatting. Programming rules help make code easier to read and understand. Python is particularly sensitive to formatting because indentations and white spaces affect code. VS Code provides support through extensions for automatically applying formatting to Python code.

From the Editor workspace, add an octothorpe, another term for the hash sign (#), to the end line of code in the `hello_world.py` file,

```
print("Hello World") #
```

and save the file. The octothorpe triggers the notification regarding PEP 8 formatting. Select *Yes* in the notification message to apply the changes. Now that Python on the Raspberry Pi has the package it requires, you need to configure VS Code to use it.

Style guides are all about code consistency. The guidelines are intended to improve the readability of code and make it consistent across a wide spectrum of Python code. VS Code has other formatters, but for this exercise, I'll explore PEP 8.

To implement the PEP 8 style guide, select the Extensions view from the Activity Bar and search for *autopep8*. Use the *Install to SSH* button to install the *Python-autopep8* extension and press the *Reload Required* button to complete the PEP 8 install on the Raspberry Pi.

From the main menubar (Figure 6, C) select *File* | *Preferences* | *Settings*. In the search bar, enter *python formatting*. From the search returns, scroll down to *Python* > *Formatting:Provider* and confirm that *autopep8* is selected. In the search bar, enter *format on save* as the search criterion. Put a checkmark beside the setting *Editor:Format on Save*. Now VS Code will use the PEP 8 style guide before a file is saved.

To take the PEP 8 formatter for a test drive, select the Explorer view in the Activity Bar and open the Python file `hello_world.py` you created earlier. Modify the line containing the print command as follows,

```
import math,os,time
print("hello world")#this is a comment
```

and save the file.

If Python PEP 8 support is functioning correctly, the script will change after the

save (Figure 9). For example, the `import` statement is divided across multiple lines and the comment line will have some spaces inserted, as per the PEP 8 rules for Python code.

Lesser Detail

To wrap up the VS Code console tour, I'll describe some of the Activity Bar items not used. The magnifying glass icon represents the Search view, in which you can search for and optionally replace content within the current editor or across multiple files.

The Source Control view (the source tree icon) provides access to source control software that tracks and manages changes to code. Recall when first preparing the Raspberry Pi environment that you installed the Git software package. Git is the most commonly used version control system today. VS Code provides support for Git through the Source Control view.

If Git is established in the Source Control environment and the project being worked on is in a Git repository, then the view will show relevant Git information, such as how many files have been changed and which files have been changed and staged, and will even commit your changes.

Source Control is important even to the amateur developer, and it can be relied on as a form of code backup. After you have spent your time writing the code, it would be nice for it to stick around if your Raspberry Pi development environment were to fail. If Git is using a repository, then the code is safe.

The Run view (an arrow with a bug), the final Activity Bar item to be mentioned, establishes a Python debugging environment. When you select the Run view icon, the debugger needs the `launch.json` configuration file. The file will be displayed with some default configurations. A `.vscode` folder is created to hold this file.

VS Code debugging allows you to step through code line by line and fix problems. To stop code at a specific point, you can establish breakpoints and conditional breakpoints. Debugging also enables log points for coding [4].

Figure 9: Python code before and after formatting, courtesy of the *Python-autopep8* extension.

Summary

VS Code provides a development environment that will make you a more efficient Raspberry Pi coder by taking over some of the tasks and freeing time for you to focus on what is important – writing code. Until next time, keep your Pi in that creative oven. ■■■

Info

[1] Visual Studio Code: *https://code.visualstudio.com/*

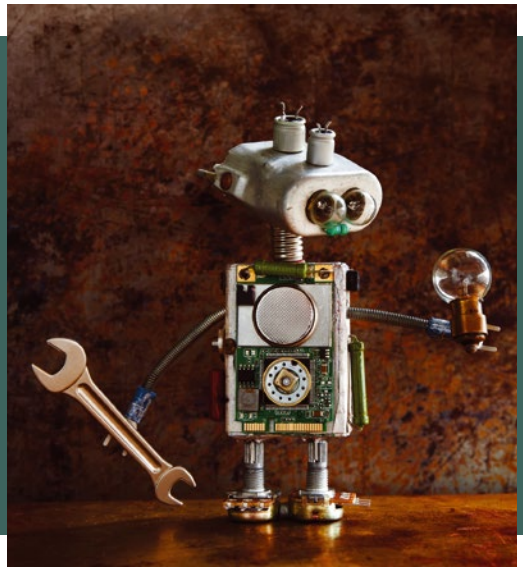
[2] VS Code Raspberry Pi install: *https://code.headmelted.com/*

[3] PEP 8 style guide: *https://www.python.org/dev/peps/pep-0008/*

[4] Visual Studio Code for Python developers: *https://www.lynda.com/Visual-Studio-tutorials/Visual-Studio-Code-Python-Developers/784291-2.html*

Author

Sean D. Conway (the D in the name is to keep his mail from being delivered to his neighbor with the same name) is a retired IT security specialist for a national telecommunication company. In addition to designing, installing, maintaining, and securing computer systems in a telecommunications environment, his 40-year career has spanned engineering ground-based aviation navigation and communications electronic systems and teaching data communication in a community college.



MakerSpace

An ASCII puzzle for an escape room challenge

Jail Break

A digital puzzle presents a challenge for young people in an escape room. *By Scott Sumner*

A teacher recently asked me to help create a couple of puzzles for an escape room she was designing for her classes. Escape rooms have a number of interpretations, themes, and implementations but ultimately comprise a series of puzzles designed around a theme. Solving one puzzle provides a clue to something else. Sometimes a puzzle is just an off-the-shelf combination lock, and as you play other parts of the game, you'll discover the combination (or three numbers that you can try as the combination).

One puzzle I designed starts with a number on a seven-segment display (see the "Seven-Segment Display" box), a bundle of leads clipped onto two rows of electrical connections, and an unfinished set of notes. A previous adventurer has started to decode the puzzle and left notes for whoever might follow. The top set of connections is numbered 128, 64, 32, 16 , 8, 4, 2, and 1. The bottom set of connections isn't labeled, but each post has an associated LED, only a few of which are lit.

Theory of Operation

Built into the code of the display box, the secret word to be revealed is part of an Arduino program. The number displayed is the ASCII representation of the character currently being sought. Players must use clip leads to connect the numbers on the top to the active (lit) connections on

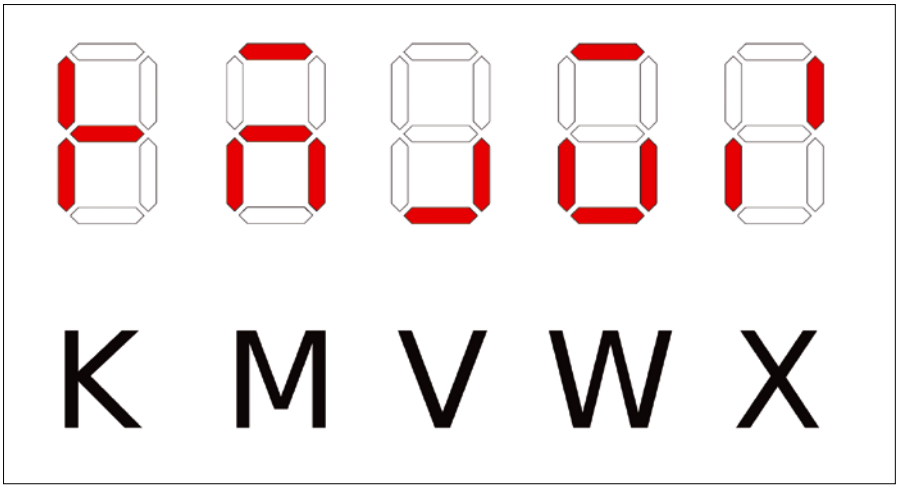


Figure 1: Several letters don't translate directly to seven-segment displays. Here's how I chose to represent them.

Lead Image © besjunior; 123rf.com

Seven-Segment Displays

Seven-segment displays are one of the most prevalent types of displays found in electronics projects. They are easy to control and very readable, even from large distances or in small physical sizes. The downside is that the value 7 is not displayed by seven discreet LEDs but by three LEDs arranged in the shape of a 7.

A number of driver chips can take care of the translation to an LED display. The MAX7219 used in this project can drive up to eight seven-segment displays (or 64 individual LEDs). If you are only displaying numbers, you can pass them directly to the 7219 in binary form, and it will display the appropriate segments to show the corresponding decimal number.

Because I want to show letters, as well, a special "no decode" mode tells the driver to just display the segments requested rather than decoding the value into a number. In this case, I pass an 8-bit binary number, where each bit represents one of the seven segments. The last bit represents the decimal point (not used in this project).

A seven-segment display by its nature has a "font" that's quite blocky. Numbers are no problem to display, and even most of the alphabet can be represented without too much trouble. However, the letters K, M, V, W, and X don't line up well with seven segments. If you use your imagination, though, these exceptions don't hinder things too much (Figure 1).

the bottom so that the connected posts add up to the displayed number. A "check" post is tapped with an extra clip lead, and the display will either say *CORRECT* or *NO*.

After each check, the active posts change so the clip leads will always have to be rearranged; however, the number being sought remains the same until it is discovered. Once the number is discovered, the letter is revealed on the left of the display, and the players can start working on the next number.

Construction

The 3D-printed faceplate (Figure 2) has places for the electrical connections. Each 1-inch #8 bolt connects directly to an Arduino pin (Figure 3); the bottom row also connects to an LED to show whether that post is active. Some 1x3 lumber cut at 45° angles form the rest of the box, with scrap plywood used for the back. Because this is a short-term project, I didn't worry about battery access, but I did put a power switch on the outside.

Code

In Listing 1 [1], the `include` in line 1 brings in an external library. In this case, that is `LedControl.h`, which controls the display driver attached to the seven-segment display.

The next several lines define the string `puzzle`, which I've set as `PASSWORD`, as the word that will be revealed as the game is played; the `iLetterIndex` variable, which is the letter position currently being discovered in the puzzle; and `cLastLetter`, which is the ASCII value of the character currently being discovered. Finally, `lc` is an instance of `LedControl` included on line 1. Its four arguments are the pin numbers for data, clock, chip select, and the number of 7219 chips in the series (in this case, just 1).

The `displayTarget` function (lines 8-19) sends to the display the number that the players are trying to find. The `int i` line is defined inside the function, so it is local to the enclosing function. In defining the `sNumber` string, the character position currently being sought is `iLetterIndex`, and `puzzle` is the secret word. This single-character string is what the players are currently seeking. The argument `DEC` specifies a decimal value of the character, or its value in ASCII, rather than the character itself and will be shown on the LED display.

`Serial.println` prints the same value to the serial monitor (purely for debugging). Because the serial monitor is inaccessible and the physical connection is sealed in a box, I didn't worry about removing it for the final version of the code.

The `for` loop iterates over each character in the string to be displayed, and line 17 sends it to the display. I'll talk about `showDigit` and how it works a little later.

countBits

The `countBits` function (lines 21-29) returns the number of 1s in a binary representation of the number `n`, which it figures out through a series of binary operations. After defining `iCount`, the function sets up a `while` loop with the condition `n`. In C, a `while` loop runs *while* its condition is not zero. Because the provided condition here is just a number, it will run until the number becomes zero.

In the binary AND (`&`) of the number `n` with 1, if the low-order bit (the least significant bit, or bit 1 in this case) is set,

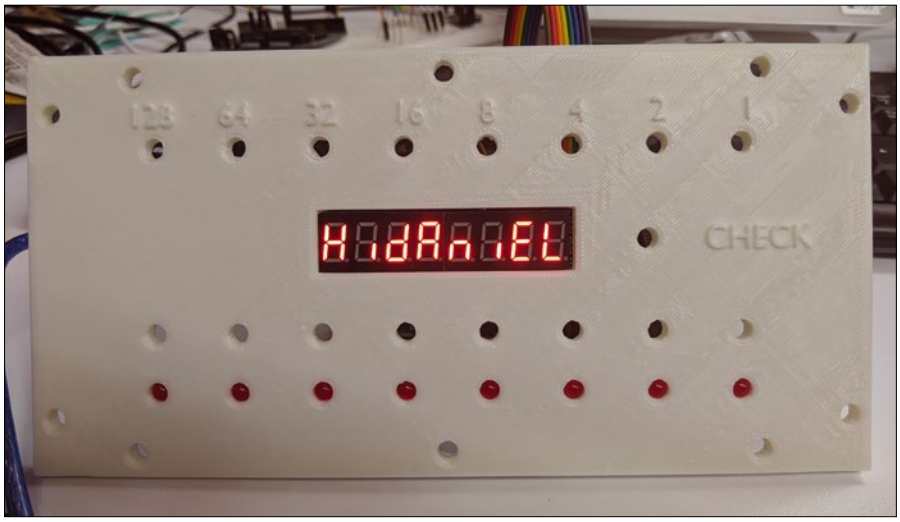


Figure 2: The puzzle faceplate after I installed the LEDs. The empty rows of holes will house bolts as electrical contacts. The holes around the outside are to mount the panel on top of the project box.

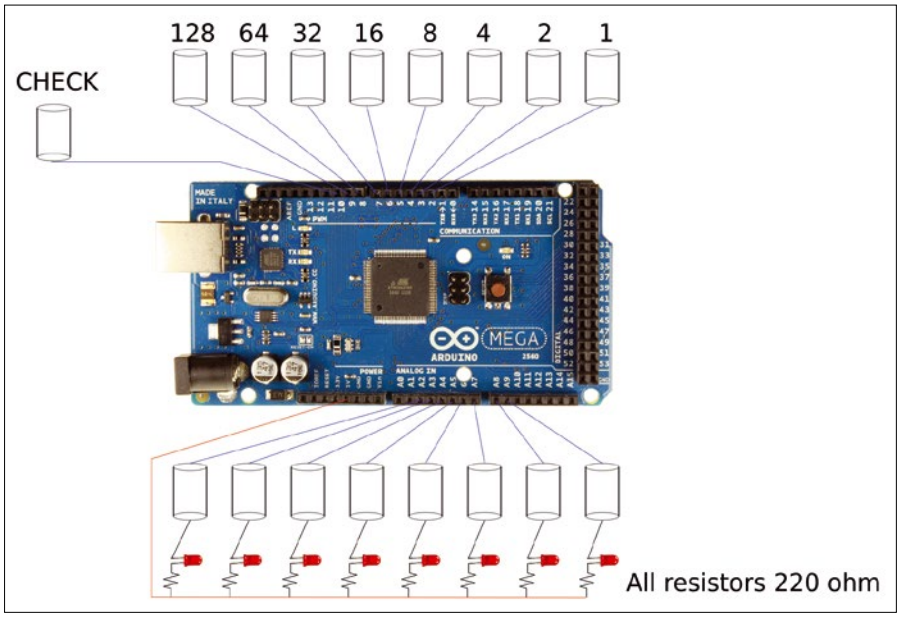


Figure 3: The wiring for the puzzle box. Note that most connections just lead from the GPIO to more robust hardware on the panel. The top set of connections use PWM pins 2 through 9, with the check post on pin 10. The bottom set of connections use analog pins A2 through A9. The red line is the 5V power connection.

Listing 1: cryptexCode.c

```
#include <LedControl.h>

String puzzle = "PASSWORD";
int iLetterIndex = 0;
char cLastLetter = 0;
LedControl lc=LedControl(14,15,16,1);

void displayTarget()
{
  int i;
  String sNumber = String ( puzzle [ iLetterIndex ] , DEC );

  Serial.println ( puzzle [ iLetterIndex ] , DEC );

  for ( i = 0 ; i < sNumber.length() ; i ++ )
  {
    showDigit ( 2-i , puzzle [ iLetterIndex ] );
  }
}

unsigned int countBits(unsigned int n)
{
    unsigned int iCount = 0;
    while (n) {
        iCount += n & 1;
        n >>= 1;
    }
    return iCount;
}

void rotateOutputs()
{
  int iNeededBits = 0;
  int iCurrentBits = 0;
  int iRandom = 0;

  iNeededBits = countBits ( puzzle [ iLetterIndex ] );
  while ( iNeededBits != countBits ( iRandom ) )
  {
    iRandom = random ( 0 , 255 );
  }

  Serial.println ( iRandom, BIN );
  if ( ( iRandom & 1 ) != 0 ) digitalWrite ( 2 , HIGH );
  else digitalWrite ( 2 , LOW );

  if ( ( iRandom & 2 ) != 0 ) digitalWrite ( 3 , HIGH );
  else digitalWrite ( 3 , LOW );

  if ( ( iRandom & 4 ) != 0 ) digitalWrite ( 4 , HIGH );
  else digitalWrite ( 4 , LOW );

  if ( ( iRandom & 8 ) != 0 ) digitalWrite ( 5 , HIGH );
  else digitalWrite ( 5 , LOW );

  if ( ( iRandom & 16 ) != 0 ) digitalWrite ( 6 , HIGH );
  else digitalWrite ( 6 , LOW );

  if ( ( iRandom & 32 ) != 0 ) digitalWrite ( 7 , HIGH );
  else digitalWrite ( 7 , LOW );

  if ( ( iRandom & 64 ) != 0 ) digitalWrite ( 8 , HIGH );
  else digitalWrite ( 8 , LOW );

  if ( ( iRandom & 128 ) != 0 ) digitalWrite ( 9 , HIGH );
  else digitalWrite ( 9 , LOW );
}

void ledYES()
{
   showDigit ( 2 , 'Y' );
   showDigit ( 1 , 'E' );
   showDigit ( 0 , 'S' );
}

void ledNO()
{
   showDigit ( 1 , 'N' );
   showDigit ( 0 , 'O' );
}

void showDigit ( int iPosition , char cLetter )
{
  int iPattern = 0;

  switch ( cLetter )
  {
    case 'A': iPattern = 0b01110111;break;
    case 'B': iPattern = 0b00011111;break;
    case 'C': iPattern = 0b00001101;break;
    case 'D': iPattern = 0b00111101;break;
    case 'E': iPattern = 0b01001111;break;
    case 'F': iPattern = 0b01000111;break;
    case 'G': iPattern = 0b01011110;break;
    case 'H': iPattern = 0b00110111;break;
    case 'I': iPattern = 0b00010000;break;
    case 'J': iPattern = 0b00111100;break;
    case 'K': iPattern = 0b00000111;break;
    case 'L': iPattern = 0b00001110;break;
    case 'M': iPattern = 0b01010101;break;
    case 'N': iPattern = 0b00010101;break;
    case 'O': iPattern = 0b00011101;break;
    case 'P': iPattern = 0b01100111;break;
    case 'Q': iPattern = 0b01110011;break;
    case 'R': iPattern = 0b00000101;break;
    case 'S': iPattern = 0b01011011;break;
    case 'T': iPattern = 0b00001111;break;
    case 'U': iPattern = 0b00011100;break;
```

Listing 1: cryptexCode.c (continued)

```
    case 'V': iPattern = 0b00011000;break;
    case 'W': iPattern = 0b01011000;break;
    case 'X': iPattern = 0b00010010;break;
    case 'Y': iPattern = 0b00111011;break;
    case 'Z': iPattern = 0b01101100;break;
    case '0': iPattern = 0b01111110;break;
    case '1': iPattern = 0b00110000;break;
    case '2': iPattern = 0b01101101;break;
    case '3': iPattern = 0b01111001;break;
    case '4': iPattern = 0b00110011;break;
    case '5': iPattern = 0b01011011;break;
    case '6': iPattern = 0b01011111;break;
    case '7': iPattern = 0b01110000;break;
    case '8': iPattern = 0b01111111;break;
    case '9': iPattern = 0b01111011;break;
  }
  lc.setRow ( 0 , iPosition , iPattern );
}

void setup() {
  // put your setup code here, to run once:
  Serial.begin ( 9600 );

  pinMode ( A0 , INPUT_PULLUP );
  pinMode ( A1 , INPUT_PULLUP );
  pinMode ( A2 , INPUT_PULLUP );
  pinMode ( A3 , INPUT_PULLUP );
  pinMode ( A4 , INPUT_PULLUP );
  pinMode ( A5 , INPUT_PULLUP );
  pinMode ( A6 , INPUT_PULLUP );
  pinMode ( A7 , INPUT_PULLUP );

  pinMode ( 10 , INPUT_PULLUP );

  pinMode ( 14 , OUTPUT );  // display DATA
  pinMode ( 15 , OUTPUT );  // display CLK
  pinMode ( 16 , OUTPUT );  // display CS

  digitalWrite ( 16 , HIGH );

  pinMode ( 2 , OUTPUT );
  pinMode ( 3 , OUTPUT );
  pinMode ( 4 , OUTPUT );
  pinMode ( 5 , OUTPUT );
  pinMode ( 6 , OUTPUT );
  pinMode ( 7 , OUTPUT );
  pinMode ( 8 , OUTPUT );
  pinMode ( 9 , OUTPUT );

  randomSeed ( analogRead ( 12 ) );
  rotateOutputs();

  lc.shutdown(0,false);
  lc.setIntensity ( 0 , 8 );
  lc.clearDisplay ( 0 );
}

void loop() {
  // put your main code here, to run repeatedly:
  if ( cLastLetter != puzzle [ iLetterIndex ] )
  {
    displayTarget();
    cLastLetter = puzzle [ iLetterIndex ];
  }

  if ( digitalRead ( 10 ) == LOW )
  {
    int iPort = 0;

    if ( digitalRead ( A7 ) == 0 ) iPort += 1;
    if ( digitalRead ( A6 ) == 0 ) iPort += 2;
    if ( digitalRead ( A5 ) == 0 ) iPort += 4;
    if ( digitalRead ( A4 ) == 0 ) iPort += 8;
    if ( digitalRead ( A3 ) == 0 ) iPort += 16;
    if ( digitalRead ( A2 ) == 0 ) iPort += 32;
    if ( digitalRead ( A1 ) == 0 ) iPort += 64;
    if ( digitalRead ( A0 ) == 0 ) iPort += 128;

    if ( cLastLetter == iPort )
    {
      iLetterIndex += 1;
      Serial.println ( "CORRECT" );
      ledYES();
      delay ( 1000 );
      for ( i = 0 , i < iLetterIndex - 1 , i ++ )
      {
         showDigit ( 7 - i , puzzle [ i ] );
      }
      Serial.println ( puzzle.substring ( 0 ,
                       iLetterIndex ) );
      rotateOutputs();
    }
    else
    {
      Serial.println ( "INCORRECT" );
      ledNO();
      delay ( 1000 );
      displayTarget();
      rotateOutputs();
    }
    while ( digitalRead ( 10 ) == LOW ) delay ( 100 );
  }
}
```

this operation evaluates to 1; otherwise, it evaluates to 0. Therefore, iCount increments by either 1 or 0 depending on whether the bit is set. The right shift operator (>>=) shifts iCount, discarding the low-order bit and moving everything over one place. The loop repeats, checking the new low-order bit again. Eventually, when the entire value has been right shifted, only 0s are left in n. At that point, n = 0, the while loop exits, and return iCount returns the number of bits that are set. You'll see how this is used in the next function.

rotateOutputs

The rotateOutputs function (lines 31-67) picks a random number until it finds one with the appropriate number of bits; then, it turns on the associate output pins (the lower row of electrical connections in the puzzle). These connections are used in the current round of the game, indicated by LEDs.

The three variables are the number of bits to be set (iNeededBits), the number of bits in the generated random number (iCurrentBits), and the current random number (iRandom).

The number of bits needed (iNeededBits) is set by calling countBits with the current character the players seek. The while (lines 38-41) loops until iNeededBits is not equal to countBits(iRandom), which was initially set to zero, but is set to random(0,255) within the loop. In this way, random numbers are chosen until one is found in which the number of set bits matches the goal; then the loop exits.

Line 43 prints the random number to the serial monitor for debugging, and lines 44-66 update all of the electrical connections so that they are all either HIGH or LOW, depending on the value of iRandom set above. The binary ANDs (&) with 1 (line 44), 2 (line 47), …, 128 (line 65), checking only one bit in the number at a time. For each bit, digitalWrite either turns the appropriate pin on or off.

ledYES and ledNO

The ledYES and ledNO utility functions (lines 69-80) just display *YES* or *NO* on the display. One or the other is called when the player checks to see whether the current wiring is correct. In each case, the showDigit function is called with position (first argument) and character (second argument) hard coded.

showDigit

The showDigit function (lines 82-126) is responsible for drawing the provided character on the display. I start by defining int iPattern and then use switch to find the character provided in cLetter.

On each line of the switch function, case says: If I match '<this>' then do whatever is after the colon (:). If it does not match, it moves on to the next case statement until it finds a match or drops off the end without a match. Once a match is found, iPattern is set to a number represented in binary with the 0b at the front (i.e., much like a preceding 0x denotes hex notation).

The binary number specifies which segments of the seven-segment display should be illuminated to display this character. Once the pattern is defined, the break exits the switch block. Without break, C would keep evaluating the rest of the case statements and possibly find another match. Here, the function exits as soon as a match is found.

Finally, line 125 calls lc.setRow, which is the LED controller's function to turn on a set of LEDs. setRow takes three arguments: the chip number (always 0 in this case), the position in which to place this character (iPosition), and which segments to illuminate (iPattern).

Setup

In the Arduino world, the special setup function (lines 128-164) is called once when the Arduino powers on, allowing you to initialize variables, turn on hardware, and make sure everything is properly configured before your main program runs. To begin, the code initializes the serial monitor in the Arduino IDE with Serial.begin (9600) for debugging messages. The baud rate (speed) of the serial connection is 9600, but this number doesn't really matter, as long as it matches what is selected in the serial monitor.

The pinMode of analog pins A0 through A7 are set to INPUT_PULLUP. As the value suggests, this makes the pin an input and also enables its internal pull-up resistor. Even though I'm using the Arduino analog pins, I'm just using them as digital inputs in this case. Pin 10 is set in the same way and will be the "check" input to determine whether the current wiring is correct.

The next series of pinModes sets a number of GPIO pins to OUTPUT. Pins 14, 15, and 16 are the data, clock, and chip select lines of the LED driver, and pins 2 through 9 are the top row of electrical connections on the puzzle. The digitalWrite sets pin 16 to HIGH, which allows the LED driver to listen to incoming data. Because the project only has one driver, it can be left on indefinitely; in this case, it could even be connected directly to V+, but this arrangement was more convenient.

The randomSeed function initializes the Arduino's random number engine. An analogRead of an unused digital input essentially picks up static, causing the return of a random number, which is used as a seed.

The rotateOutputs was defined earlier, so when the puzzle starts, the lower row of electrical connections is ready to go.

The last couple of lines set up the LED driver. An lc.shutdown set to false means the LEDs should *not* be shutdown. The lc.setIntensity sets the brightness of the display. The range is 0 to 16, so a value of 8 sets brightness at 50 percent. Finally, lc.clearDisplay erases anything that was previously on the display. The first two functions are called with a 0 as the first argument. The 7219 driver chip can be daisy-chained, so you have to specify the chip in the chain to which you are talking. Because this setup only has one chip, the value will always be 0.

Loop

After the setup function finishes, the Arduino loop function (lines 166-210) loops continuously until power is removed. In this function resides the main logic of the program.

To begin, I check to see whether cLastLetter does not equal the current puzzle character, displayTarget (i.e., show the value the player is trying to achieve), and update cLastLetter to the current puzzle character.

Next, I check to see whether pin 10 is LOW. If so, the player wants to know whether their current wiring is correct. To check, I initialize iPort, then digitalRead each of the input pins, and add the appropriate power of 2 if the pin is LOW (lines 178-185). For a pin to be LOW, the player had to connect a clip lead

Figure 4: These 3D-printed maze boxes were in a different part of the escape room. This is how they looked immediately after printing.

between this connection and one of the active connections on the lower row indicated by LEDs.

Once I've calculated `iPort`, I check to see if it equals `cLastLetter`, which means the players have found the correct combination. In that case, I increment `iLetterIndex`, debug print `"CORRECT"`, display *YES* on the LEDs, and then wait for 1 second (line 192).

The `for` loops over the characters discovered so far. I call `showDigit` and provide the position as `7 - i`, which makes sure the drawing starts from the left side of the display. The second argument, `puzzle [ i ]`, is the character at that position. As each character is discovered it is added to the display.

Finally, `rotateOutputs` ensures a different set of connections will be active for the next round. On the next iteration of the loop, `iLetterIndex` has been updated, so a new number will be displayed for the players to find.

If the players did not guess correctly, the `else` block (line 200) runs instead, debug prints *INCORRECT*, displays *NO* on the LEDs, waits for 1 second, redisplays the target value, and finally picks a new set of active connections with `rotateOutputs`.

Line 208 loops infinitely `while` a `digitalRead` of pin 10 returns `LOW` and delays a tenth of a second before checking again. Once pin 10 returns to `HIGH` (the clip is removed), the loop continues.

Designing for Children

Once your adventurers have decoded the puzzle, they can use the clue they discovered to unlock their next challenge. For the same escape room, I 3D-printed a second challenge comprising of a couple of maze boxes (Figure 4) that could be opened by twisting, pushing, and pulling. Young makers are very enthusiastic but often don't know the limits – that come with experience and maturity – of the materials involved. However, they can reveal some interesting failure points that you might not have considered. Figure 5 shows some of the pieces after being used in a classroom for only a couple of hours. ■■■

Figure 5: After several hours of hands-on use, the maze boxes show signs of abuse. Absent direction (and sometimes even then), children tend to resort to the use of extra force to try to solve the problem at hand.

Info

[1] Code for this article: *ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/243/*

Author

Scott Sumner is the Assistant Manager of the Charlie Noble Planetarium at the Fort Worth Museum of Science and History. He enjoys using Python to solve as many problems as possible.

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!
shop.linuxnewmedia.com

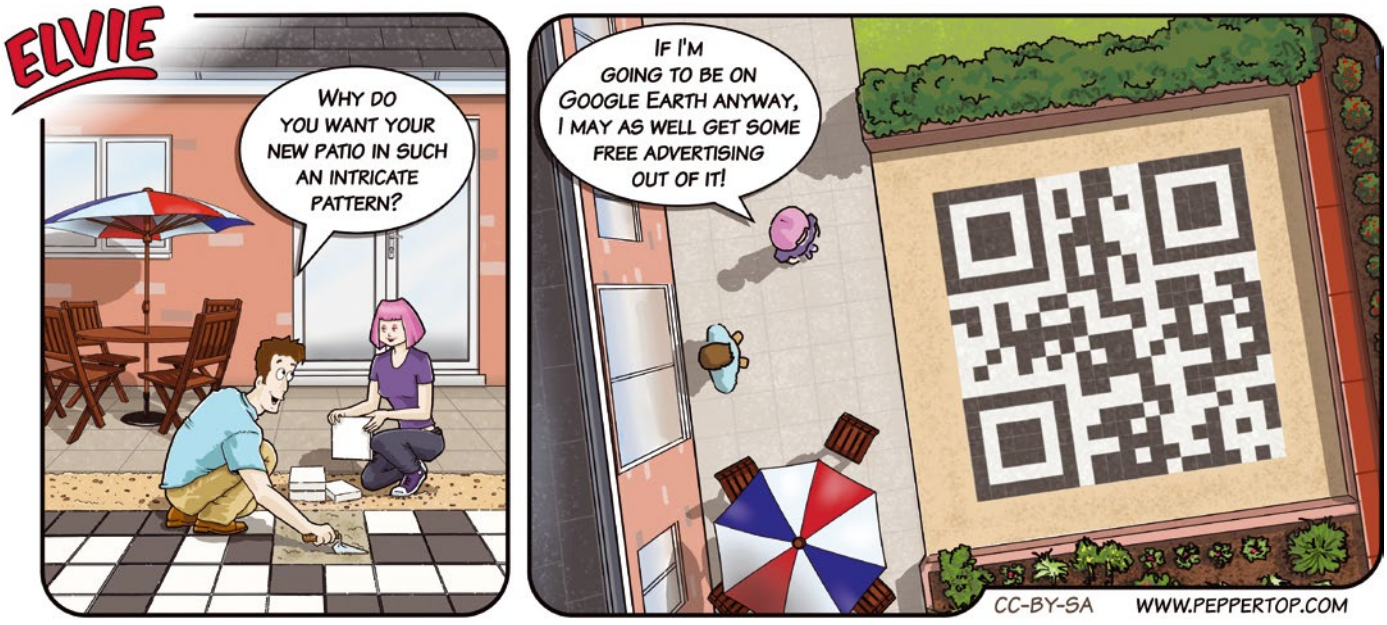
Those of us who remember the old days sometimes long for that bygone era when "desktop" meant the real top of a desk and not an electronic picture on a computer screen that vaguely resembles the top of a desk. But memories can be deceiving. Those old-time desktops used to pile up with clutter, which some of us were much better at organizing than others.

Come to think of it, our new virtual desktops can also get cluttered, and not just with application icons. The system of files and directories at the heart of the "desktop metaphor" can easily get stuffed with lots of eye candy you'll never use – because you'll never slow down long enough to make changes.

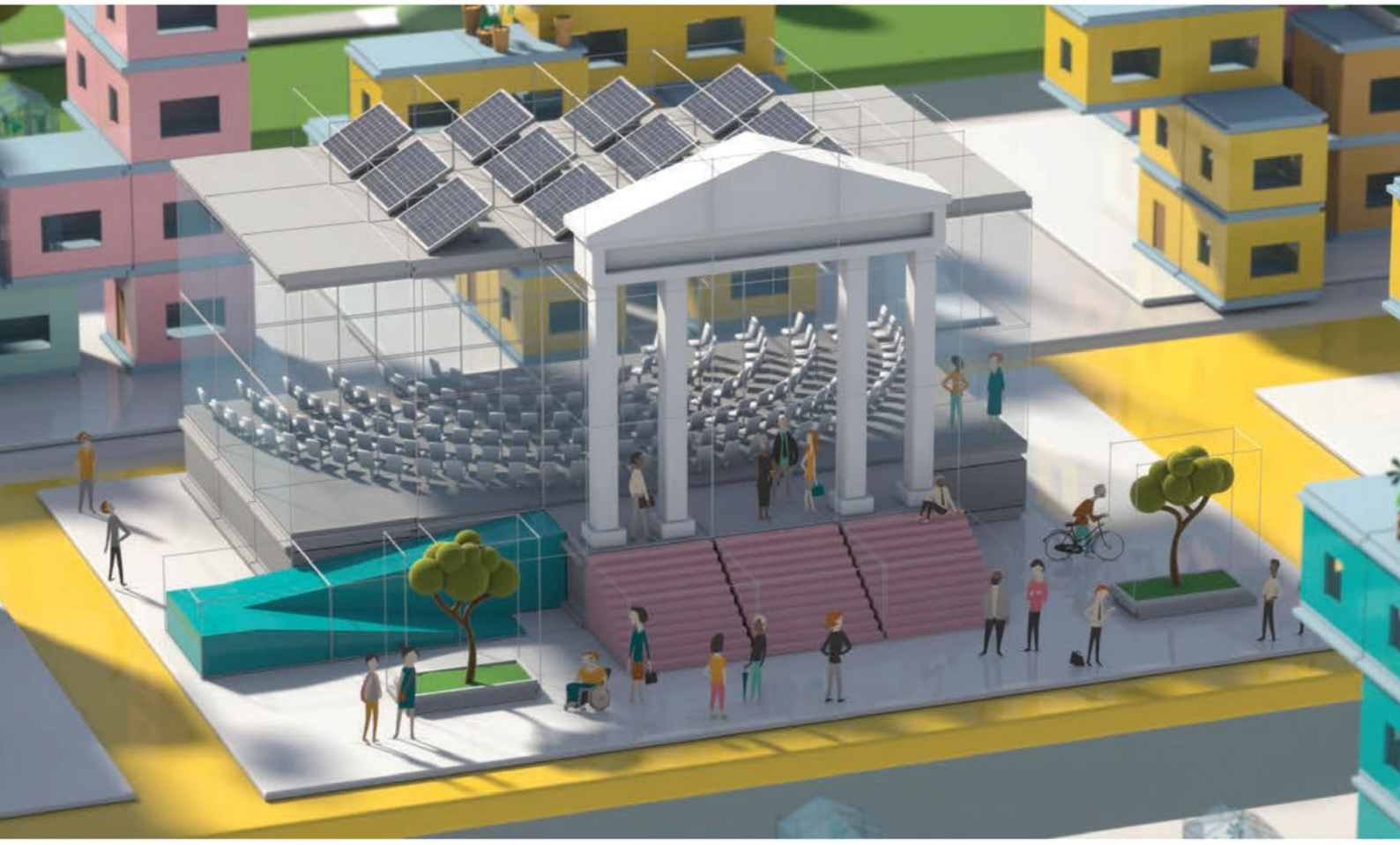
But Linux has oh so many tools for everything: this month we feature a utility that rotates wallpaper and background images – so you can benefit from the diverse designs stored on your system without having to sort through it and switch the pictures manually. And speaking of organizers, check out this month's tutorial on Ventoy, a cool tool that lets you boot multiple ISO system images from a single USB stick.

Image © Olexandr Moroz, 123RF.com

LINUXVOICE



Public Money
Public Code



Modernising Public Infrastructure with Free Software

fsfe Free Software Foundation Europe

Learn More: https://publiccode.eu/

MADDOG'S DOGHOUSE

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®.

A recent rocket launch has maddog thinking about high performance computing and accurate weather forecasts.

BY JON "MADDOG" HALL

Fluid Dynamics

Recently SpaceX, working with NASA, sent a capsule carrying four astronauts to the International Space Station. The launching and flight were flawless.

I have been watching space flights since 1961. I still remember all of the students of my elementary school packing into the school's auditorium, with the school wheeling in their large black-and-white TV on a cart (one of the few "portable TVs" in those days) to watch it launch.

That is assuming that the rocket did take off, because in those days it was very likely that a storm might move in and have the countdown delayed, or even canceled. The act of fueling the rocket and putting the astronaut (first one, later multiple astronauts) in the capsule had to be planned and executed far in advance of the takeoff, and Cape Canaveral (as it was called in those days) was very likely to have bad weather.

We have known how to predict weather 24 hours in advance for a long time with 100-percent accuracy. The problem was that it took 48 hours to gather and process the data, so we could tell you precisely what the weather would be 24 hours in the past.

The problem of weather forecasting is one called "fluid dynamics," and this is a common problem in many things we need to compute. Heat flow, turbine efficiency, virus spread, airplane and car design, you name it … they are all "fluid dynamics." Even in objects we consider "solid," there are applications of fluid dynamics technologies that can be computed.

Before 1994 these type of problems were tackled by "supercomputers," machines and operating systems designed by companies like Cray, ECL, CDC, IBM, and others. These companies would spend many, many millions of dollars on developing these supercomputers, using state-of-the-art technology, and then produce a limited number of machines that might gain the title of "world's fastest" for a couple of years until the next one came along. Often the development costs were not covered by the profits from the sales of machines and service. Some of these companies (or at least their supercomputer divisions) were going out of business.

Then two people at NASA, Dr. Thomas Sterling and Donald Becker, developed a concept that allowed the use of commodity computer components to solve these compute-intensive problems by dividing these problems into highly parallel tasks, which they called "Beowulf" systems. Roughly speaking, you could see about the same computing power from a system like this that you would see from a "supercomputer" that cost 40 times more. In addition, since these commodity based systems used an Open Source operating system (typically GNU/Linux-based) the programmers who worked on fluid dynamics problems knew how to program the system with the addition of a few libraries – Parallel Virtual Machines (PVM), Message Passing Interface (MPI), and Shared Memory Interface (OpenMP) – as well as standard methods of breaking apart large programs (decomposition, thread programming, and parallelism).

The Beowulf concept – now called High Performance Computing (HPC) and used on the world's 500 fastest computers – also allowed for people to experiment with better compilers and systems with relatively inexpensive hardware. Systems purchased for some large problem could be disassembled and repurposed for other tasks when the large problem was finished.

Early systems included Oakridge National Lab's "Stone Soupercomputer" (named after the "stone soup" fable), made of 133 cast-off desktop systems connected with simple Ethernet as the communication between boxes. Implemented by William Hargrove and Forrest Hoffman in response to not receiving the funding for a traditional supercomputer, the Stone Soupercomputer helped to solve several real-world problems, as well as act as a system to develop new HPC applications.

Over time various distributions (Rocks and OSCAR were two) specializing in this type of computing came out from the various National Laboratories that made it easier to set up your own high performance cluster, including the use of Raspberry Pis as the hardware.

Over the years, the time needed to gather and process the data to forecast weather 24 hours in advance, with 100-percent accuracy, dropped from 48 hours to 24 hours (stick your hand out the window) to 12 hours, and after that we never had to suspend a NASA launch because of weather. Over a broader scale, these calculations also helped predict weather for sporting events, weddings, agriculture, and other issues.

Today the same technology is being used to calculate the damage of a hurricane versus what would happen if the environment had been one or two degrees cooler, important in the age of climate change. Weather forecasters are showing that temperatures of a few degrees can mean rainfall differences of 10 or 15 percent. In a rainfall of 12 inches during a hurricane, this can mean one to two inches of additional water for flooding, enough to make a difference in whether your home or business will be flooded. ■■■

Add Variety to your desktop
Newly Wallpapered

Don't let your desktop look boring. With the Variety wallpaper manager, you can easily set up a rotating selection of background images and other customized options.

BY ERIK BÄRWALDT

Popular desktop environments under Linux usually come with a gallery of background images to give the desktop an individual touch. But over time, seeing the same old images while the computer boots up can become boring.

A few desktop environments, like KDE Plasma, have integrated a slideshow function to automatically rotate your desktop wallpaper at set intervals. However, most desktop interfaces under Linux do not offer this option. A simple solution to this problem is to install Variety [1], a tool that allows you to easily download and rotate different background images.

Installation

Many Linux distributions and some BSD derivatives come with the software already in their repositories. The project's GitHub page [2] also provides the source code from which the application can be built.

After downloading to your system, if you used the binary packages, you will find a starter in the desktop environment's menu tree. A click on this opens the software's Settings dialog where you first need to confirm the license. After doing so, the wizard creates an autostart entry on the system, which the application uses to load the background during startup. At the same time the routine drops an icon into the system tray of the desktop and opens the Preferences dialog. You can set up your individual configuration in this window (Figure 1).

Preferences

In *General*, the first tab of the Preferences dialog, you define the time interval with which to change the wallpaper, whether the software is loaded at system start time, and the sources from which you want the program to load the background

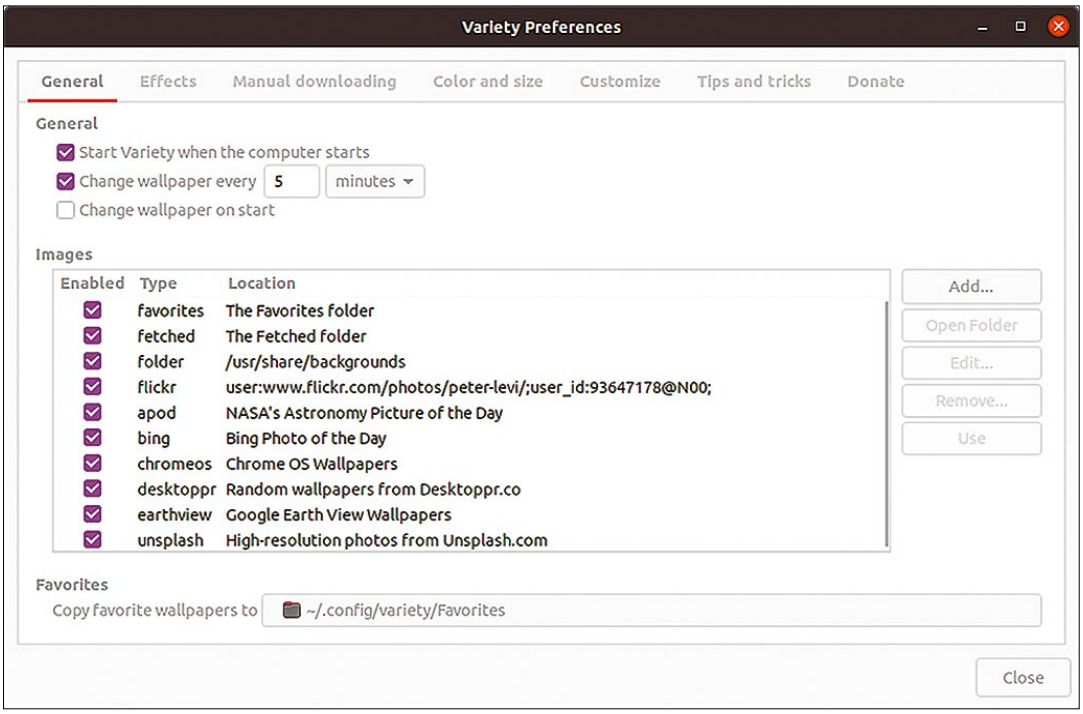


Figure 1: The Preferences dialog in Variety gives you many options for choosing and changing your desktop wallpaper.

images. To help you with this, Bulgarian developer Peter Levi added numerous directories with publicly accessible online image collections. You will find photos from Google Earth and NASA, among other sources. They are all enabled by default, but unchecking individual list entries disables unwanted sources.

By pressing the *Add* button to the right of the list of image locations on the web and then selecting a new source, you can integrate your own image collection. To make individual images on the system accessible to Variety, press the *Images* button in this dialog and then select the desired images in the file manager. The *Folder* option lets you transfer the complete contents of a folder to the Variety list in the file manager.

In the *Effects* tab, you can change the way the image is displayed. For example, by checking a box, you can blur or pixelate background images or display color images in black and white. ImageMagick takes care of these effects, so when Variety is installed, this software is also installed on the mass storage device if it is not already there.

Quotes

Under *Quotes* in the *Effects* tab, you activate a display of changing quotes on the desktop. Variety pulls the text from various sources on the Internet. Alternatively, you can create your own collection of quotations in a text file.

You can search the collections of quotes by keyword and author and may choose to include only those that match the given criteria. On many distributions, you have to explicitly select a font in the quotes settings dialog once; otherwise only special characters will appear on the screen.

Optionally, you can display a digital clock and the current date on the desktop by checking a box in the *Clock* group at the bottom (Figure 2).

For further manual configuration of time and date settings, Variety's developer provides detailed documentation online. You can reach it directly from the *Preferences* dialog under the *Effects* tab.

Color and Size

The images used by Variety as wallpaper do not necessarily have to have the same resolution as the monitor. The software lets you define different selection criteria for the images to be displayed in the *Color and size* tab of the *Preferences* dialog.

You will usually want images in landscape format, but Variety also supports the use of images in portrait format. For low-resolution images, which are often found on the Internet, specify a minimum size, which is based on the screen resolution. This prevents images with too low a resolution from becoming blurred when overmagnified.

Basically Variety adjusts the image to the current resolution of the screen by cropping. The program avoids unsightly bars at the edges or distorting of images. Unlike many other desktop background applications, Variety does not display content in a stretched or compressed form. You can also make the software display only those images that contain a high percentage of a certain color.

Tips

In the *Tips and Tricks* tab, you will find more detailed information about the application, for example, how to use it on the command line. In addition, developer Peter Levi provides a detailed

Figure 2: In addition to a choice of rotating background images, Variety gives you options for how to display the date and time.

changelog. In this tab you can also access the program's website directly, contact the developer, and find a link to a donation option.

The software automatically adds itself to the system tray at startup time. You can also use the program as an image viewer. In the corresponding context menu, which can be opened by right-clicking on the Variety icon, you can also open the Preferences dialog at any time and modify various options. The selection function is a special feature: After checking *Wallpaper Selection*, the program will show a reduced view of all image files in the current folder. If you have downloaded images off the Internet, the images stored on the clipboard will also appear here. Click on one of the reduced images to enable it as a new wallpaper (Figure 3).

There are two options *Previous* and *Next* with which you can influence the image sequence manually. To make modifications in the image file folder, click on the file path shown in the context menu. The routine now opens the file manager with the target path and closes Variety. You can add new images to the folder or delete old ones without causing problems with the background image display.

Conclusions

Variety currently offers what is probably the most flexible solution for customizing your desktop background. Regardless of the distribution and desktop environment you use, the software is very stable and avoids excessive resource consumption. By cropping the backgrounds to ensure they display correctly, Variety eliminates the need to adjust desktop backgrounds to the screen resolution. The software is therefore perfectly suited for creative users who expect more from their desktops. ■■■

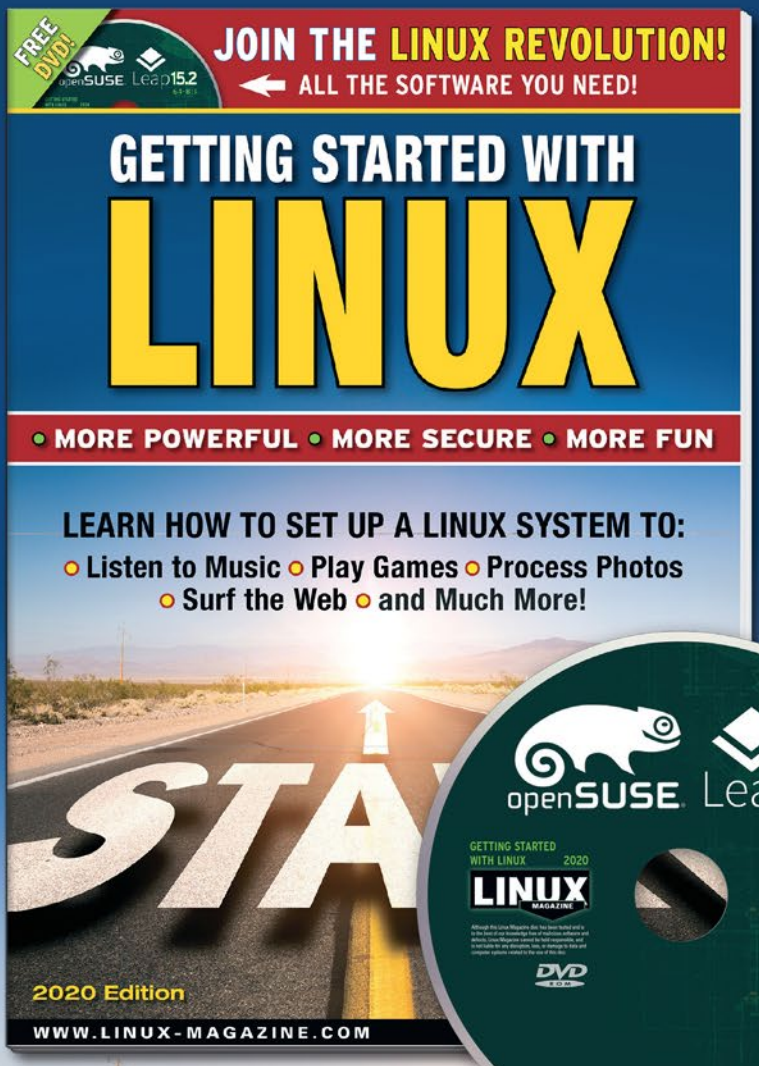
Info

[1] Variety homepage: *https://peterlevi.com/variety/*

[2] Variety GitHub project page: *https://github.com/varietywalls/variety*

Figure 3: The picture-in-picture option lets you select your wallpaper with a single mouse click.

Hit the ground running with Linux



Want your friends and colleagues to make the switch to Linux?

This single issue shows beginners how to:

- install Linux
- download and install free software for your Linux system
- play games
- create documents and spreadsheets
- process photos
- play music and videos
- and much more!

ORDER ONLINE: shop.linuxnewmedia.com/specials

System resource monitoring with bpytop

Fast and Functional

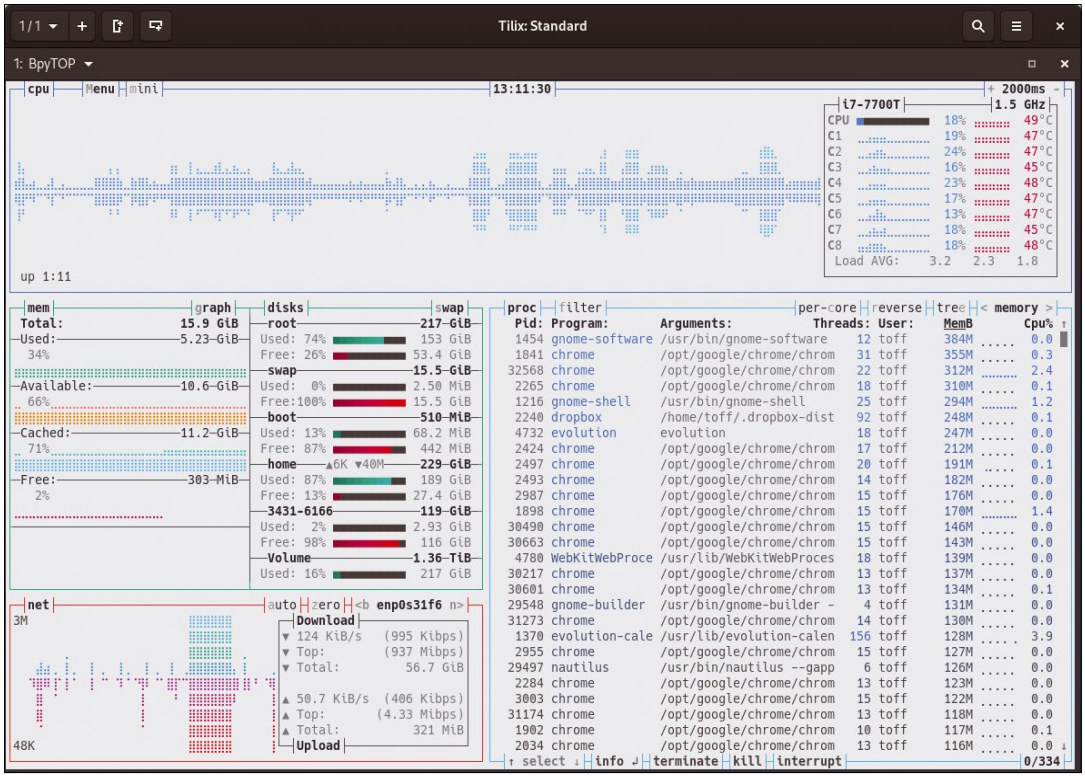
Linux users have many options for monitoring system resources, but bpytop, a new Python port of bashtop, more than stands out from the crowd. BY CHRISTOPH LANGNER

With a system like Linux, which is available to multiple users at the same time, it is important to keep an eye on the available resources. Is a graphical application running haywire and blocking all of the CPU's cores? Is a user running a computationally intensive script and thus making work agonizingly slow for the server's other users? Has a process gone wild and started to write increasing amounts of corrupted data to the hard disk?

Such questions can be answered by various monitoring tools. The popular distributions typically preinstall Top. The program provides an overview of the active processes and measures both the CPU load and the RAM usage. Htop offers similar functions in a visually appealing format. Other programs display the network load (iftop) or determine which applications or processes cause massive writes to the hard disk (iotop).

Bashtop [1] combines many of these functions in a text-based interface. The program displays the CPU load both numerically and as a pseudographic. It lists the active processes along with their memory and CPU consumption, visualizes the throughput rates of the individual network interfaces, and much more. The program has a downside, however. As the name already suggests, it was programmed completely as a Bash script. This makes further maintenance and development complicated; it also slows down the

Figure 1: Bpytop shows many details for the system. The standard design is a little darker, but this brighter theme improves visibility.



Listing 1: Install bpytop

```
01 $ sudo apt install git build-essential python3-distutils python3-psutil
02 $ git clone https://github.com/aristocratos/bpytop.git
03 $ cd bpytop
04 $ sudo make install
05 $ sudo make uninstall
```

program. But now there's another option that addresses these issues, because bashtop's developer has released an official Python port of the utility called bpytop [2].

Installation

Bpytop was first introduced in August 2020. Accordingly, most of the major distributions have not yet added it to their package sources. Only Arch Linux and its derivatives, such as Manjaro, offer the program, and only in the Arch User Repository (AUR). Arch users can call an AUR helper such as Yay for the install. The `yay -S bpytop` command imports bpytop with all dependencies. For Ubuntu, there is a Snap package [3], but it makes very little sense to set up a lean command-line tool with a monster like Snap. In this case, a manual install is the preferred approach.

The commands from the first four lines of Listing 1 help you load the build dependencies onto your system and build bpytop from the source code previously downloaded off GitHub. If you keep the `bpytop/` build folder, you can cleanly remove the program from the system with the command from line 5. After the install, call the program with the `bpytop` command in a terminal window. There is no entry in the start menu of the desktop environment.

Operations

The bpytop screen is divided into several areas (Figure 1). At the top, the program displays the load and temperature of the individual cores (see the "Sensors" box), as well as a plot of the utilization as a graph. Below this on the left, you will find the main memory and data carrier usage. In the box below, bpytop displays the current transfer rates for a selected network interface card and the corresponding history. The box on the right contains a list of all active processes together with their memory usage, resulting CPU load, process ID, user, number of threads, and other data.

Within the application you can now use the mouse or the keyboard. Gray letters mark dialogs and switches – you have to look carefully here: The letters are small, and options are case sensitive. For example, pressing *m* or clicking on *mini* will switch to a reduced mode that only displays the taskbar and CPU history (Figure 2). Pressing *M* or clicking on *Menu* opens a menu where you can access the settings and help features or quit the program (Figure 3). Use *b* or *n* or click on *<b* or *n>* to change the network interface card displayed in the *Net* area.

In the settings, you can choose between 10 different color variants in *Color theme*. You can switch between the options with the left and right arrow keys. There is also the possibility to permanently activate mini mode or to disable the color gradient in the process list (*Proc gradient*). If required, additional drives can be included or excluded from the display in *Disk filter*. The program in the right-hand column provides information on the individual options.

Processes

Bpytop uses most of its space to display the process list. By default, it sorts the list by CPU usage. You can use the left and right arrow keys or click on the square brackets next to the current mode (for example *cpu lazy*) to change the sort order.

Use *reverse* to reverse the sort order; *tree* activates a tree view. With *f* and a search term, you can filter out certain processes from the mass of data. Press Enter to complete the filter so that

Sensors

For bpytop to be able to display the temperatures of the individual CPU cores, as well as the processor load, you need *lm-sensors*. The package reads the temperature sensors of numerous CPUs and mainboards. To do so, install *lm-sensors* on the system (on Debian-based distributions type `sudo apt install lm-sensors`) and then start sensor detection by typing `sudo sensors-detect`. You can usually press Enter to accept the defaults. At the end, the script identifies the kernel modules needed for the system and offers to add them to `/etc/modules` or `/etc/conf.d/lm_sensors` (depending on the distribution) so that the system loads them automatically at boot time. After a reboot, the `sensors` command will show you the CPU and motherboard temperatures, and bpytop will now also provide the temperatures.

Figure 2: Mini mode removes the displays for hard disk usage and network interface card utilization. This leaves more space for the active processes running on the system.

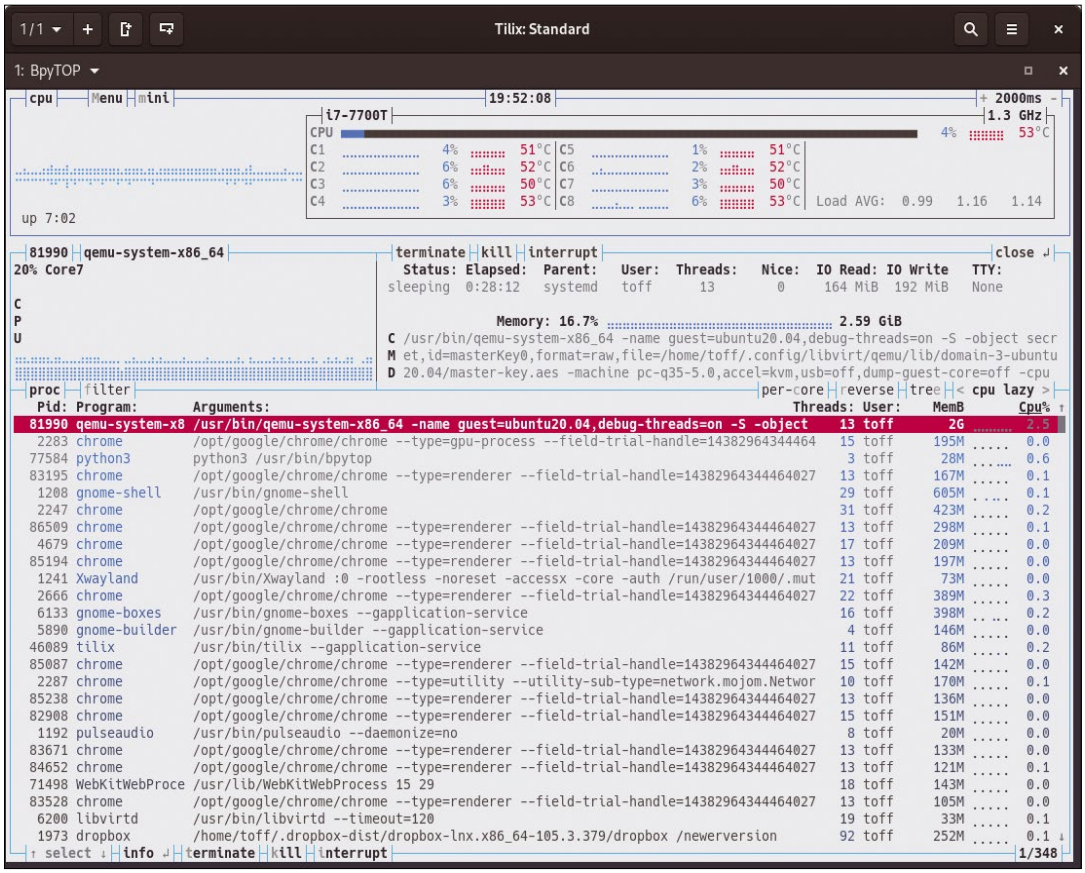
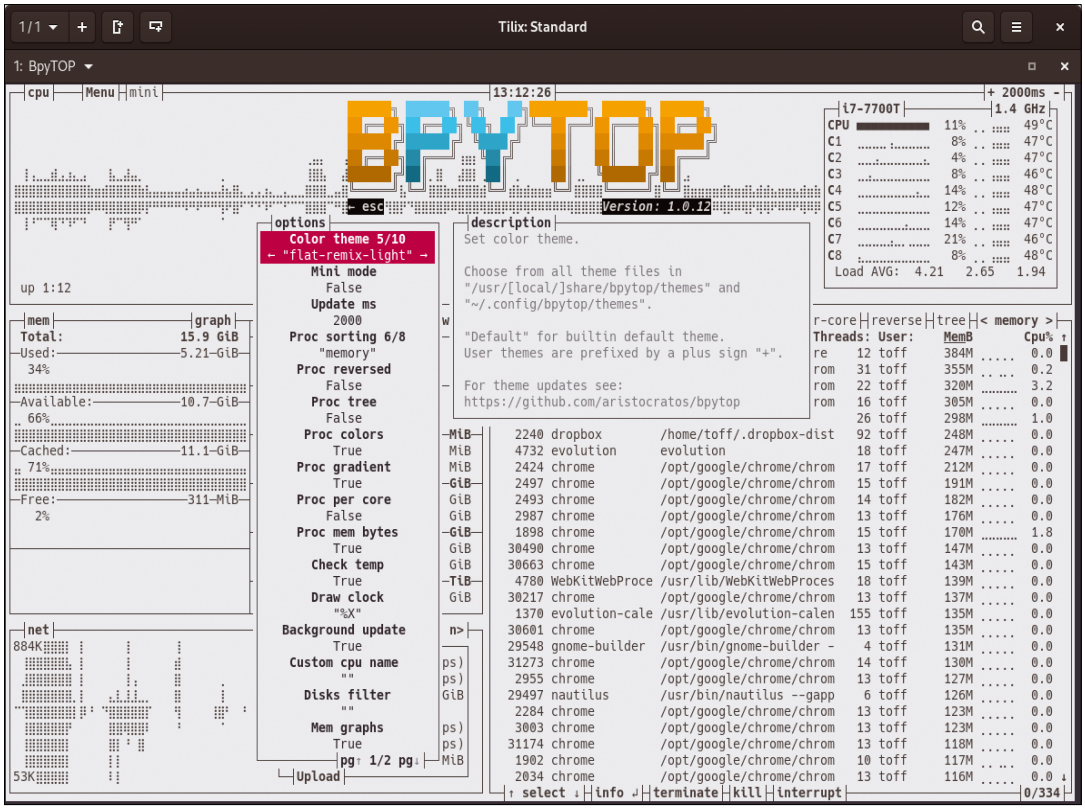


Figure 3: In the detailed settings of bpytop, you can choose between 10 different themes and numerous other configuration options.



you can browse the list of matches using the up and down arrow keys. If necessary, you can press Del to empty the filter again.

Once you have selected a process in the list, a menubar appears below the box with *info*, *terminate* (sends signal `SIGTERM`), *kill* (`SIGKILL`), and *interrupt* (`SIGINT`). You can trigger an action either by clicking on the corresponding entry or by using the highlighted shortcut keys. However, be careful when terminating processes; there are no prompts. If you shoot down the wrong process, unsaved data may be lost.

Conclusions

Although similar functions have been offered by tools like Top, htop, or Glances [4] for quite some time, having another arrow in your quiver can certainly be an advantage. Bpytop works smoothly on current Linux systems and makes a flawless impression both in a virtual terminal and via SSH. The developer shows great commitment and responds quickly to bug reports and suggestions for improvement. You can also quickly contact the developer on social media, such as Reddit [5]. ■■■

Info

[1] bashtop: *https://github.com/aristocratos/bashtop*

[2] bpytop: *https://github.com/aristocratos/bpytop*

[3] Snap: *https://snapcraft.io/install/bpytop/ubuntu*

[4] Glances: *https://github.com/nicolargo/glances*

[5] gnmAristocrat: *https://www.reddit.com/user/gnmAristocrat*

FOSSPicks

Sparkling gems and new releases from the world of Free and Open Source Software

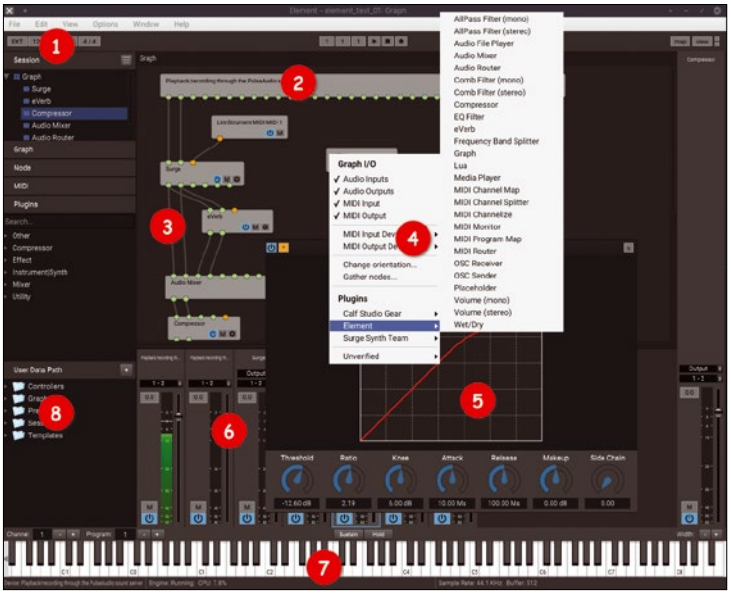
Graham has been so enamored with the Bevy game development framework he discovered in this issue that he's embarked on creating the isometric RPG he always wanted to write as a teenager. BY GRAHAM MORRISON

Audio host

Kushview Element

Linux has become an incredible platform for audio experimentation. It may not have the professional-grade applications seen on macOS and Windows (nor the ARM cross-compilation woes). It may not have the plethora of proprietary plugins that make those systems professionally viable, but it does have the frameworks, the developers, and an expanding niche of experimental music producers that are willing to forgo the convenience of mainstream applications in a quest for creativity-driven innovation. Norns, Zynthian, and ORAC are all amazing open source Linux-based examples of this. Each is open to anyone willing to step out of their comfort zone and try something new, and they all have huge communities trying to improve and create in equal measure.

Another piece of this puzzle is the plugin framework. This allows you to use the same instruments or effects on whichever platform you prefer: Windows, macOS, REAPER, Ardour, Renoise, Bitwig Studio, and many other applications. Natively, Linux has the LV2 plugin format, but the industry standard is Steinberg's venerable VST. VST instruments and effects on both macOS and Windows are the cross-platform standard, but they seldom make their way to Linux due to VST SDK legacy licensing issues. Fortunately, Steinberg released its latest VST 3 SDK under the GPL, opening the door to Linux-native open source VST plugins and leaving us with the next problem. There are too few open source host applications with which to play with these potential plugins. Consequently, there's nothing to encourage their development. It's the classic chicken and egg problem.



1. JACK or PulseAudio/ALSA: Element will work with both popular audio back ends and a huge number of plugins. **2. Audio, MIDI, and OSC:** Like JACK on steroids, Element lets you create and save a "graph" of audio, MIDI, and OSC mapping. **3. Multiple connections:** Use mixers and multiple connections to create effect sends and returns. **4. Internal nodes:** Element includes its own effects, processors, and MIDI device nodes. **5. Internal effects:** Internal effects, such as the compressor and audio router, have their own GUI. **6. Mixer:** Control the volume from synths and effects with a mixer view. **7. Virtual keyboard:** Even without a MIDI device, you can generate MIDI data. **8. Preset management:** Save your own presets, templates, and configurations.

This is where Kushview Element saves the day. Not only is it an open source host designed to launch VST 3 instruments and effects, alongside LADSPA and LV2 formats, it's also a comprehensive audio and MIDI processing environment that works seamlessly with ALSA, PulseAudio, and JACK. It's the perfect hub for all Linux audio experimentation. The central interface is based on cable connections you draw between input and output modules that represent either PulseAudio's inputs and output or that of your audio hardware. It also uses modules that you create from plugins. This is called the "grid" view, and if you have the amazing open source Surge VST 3 synth installed, for example, this can be added from the right-click menu and its outputs connected to your speaker or headphone inputs. You can then connect either a MIDI input or the virtual piano keyboard to trigger notes on Surge's MIDI input. Add more effects and other instruments, route audio and MIDI data freely between them, and even remote control all values, including the built-in audio mixer. It's the perfect virtual studio to host your favorite sound and processing plugins. You can then save your setup for easy recall, whether you're experimenting with sound, playing live, or listening to music.

While Element is open source and can easily be built from its GitHub repository (we did this), the latest binaries are available via a low-cost, one-off fee or recurring subscription. Ardour uses a similar "paywall" strategy that has undoubtedly helped ensure its continued development, paying the salary of its lead developers and getting Ardour used in all kinds of professional, educational, and amateur setups. Hopefully, paying for binary downloads helps Element to succeed in the same way, because the application itself is definitely worth it.

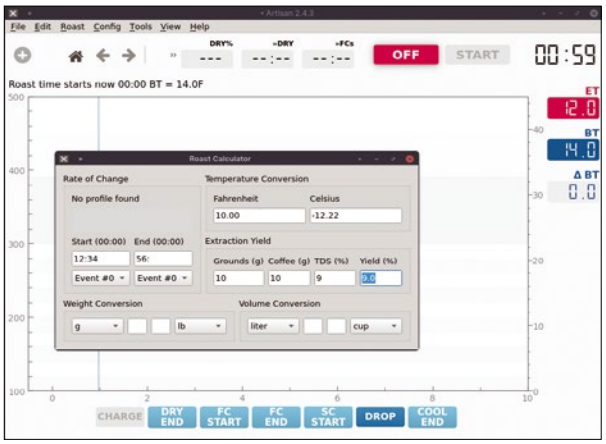
Project Website
https://kushview.net/element

Coffee roaster

Artisan

If there's one trait that seems extraordinarily common to people in the Linux and open source communities, it's the desire to deconstruct a process so it can be rebuilt better. You see this in software development, but it also reaches into many other geek-friendly niches, from retrocomputing and circuit design through to craft beer brewing and creating the perfect coffee. It could be because these subjects require a lot of data and a lot of parameter tweaking. In beer brewing, for example, even a simple brew requires a complex recipe, varying temperatures and pressures, and particular carbon dioxide levels. It's also why software like BrewPi and Fermentrack, both of which we've previously looked at, produce such good results. They manage the data processing for us and allow us to track and create the perfect reproducible brew.

The realm of coffee brewing is the same. Whether it's at the roasting stage, the grinding stage, the tamping stage, the brewing stage, or the frothing stage, coffee presents limitless options for tuning, adapting, and perfecting. Artisan can at least help with the first of these. It's an open source application that does for the coffee roasting stage what BrewPi does for the beer brewing stage. It connects to many popular heating mechanisms designed for coffee bean roasting and allows you to monitor and control everything about the roast. This collection of metrics then becomes a profile that can be used to perfect your next roast. The main functions include a profile analyzer, roast comparator, profile transposer, and roast simulator. The analyzer is the heart of the application, as it maps incoming data to curves and sets profiles. A second profile can be loaded into the background for comparison, and the transposer helps you map profiles from other hardware. You can then run the simulator to engage the hardware without wasting the beans. It's complex, but much easier than writing this stuff down or controlling it manually.



The analyzer is the heart of Artisan, as it allows you to track many different values across roasts and easily view and compare the most important values at any time.

Project Website

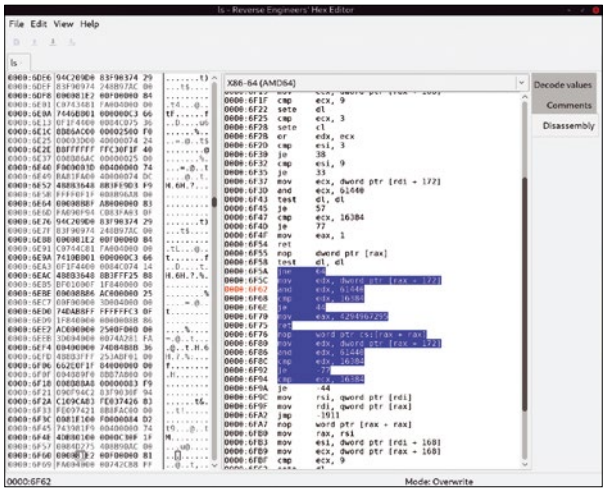
https://github.com/artisan-roaster-scope/artisan

Hexadecimal editor

rehex

Back in the olden days of home computing, after your pocket money had run out and you'd completed your last computer game, the only thing left to do was revisit the games you'd long since finished. But this didn't necessarily mean replaying them. With the help of a cartridge like the Action Replay on the Commodore Amiga, or a dismantled joystick button located between two serial port pins on a Commodore 64, you could reset your machine whilst keeping what was previously running in memory. You could then carefully peek and poke your way around the RAM – easily with the Action Replay – and even sideload a memory editor, allowing you to explore a disassembly of the original game code alongside the raw image, sound, and level data from which the game was built.

In the 1980s and early 1990s, hardware limitations meant the code and data you could view was typically free of the hardware or operating system abstraction you generally have to deal with today. Executables on Linux, for example, are compiled against many different libraries running at many different levels, and the running binaries themselves are often confined from accessing anything outside of user space. But that doesn't mean there isn't fun to be had and insights to gain by looking inside those binaries. Reverse Engineers' Hex Editor (`rehex`) can do this because it has been developed to help you get the same kind of insight into how binaries run that you might have gotten by pressing the red button on an Action Replay. It will load a binary, or any other file, and show its contents on a hexadecimal grid, even files greater than 1TB in size. You can edit these values and conveniently view them as various decoded value types, which is useful when you're looking for specific patterns. But what makes `rehex` really special is the integrated disassembly of an executable's machine code, similar to those old '80s hacks, and equally as liberating – even in the modern era.



The `rehex` hex editor also disassembles a binary, allowing you to explore how an application actually works.

Project Website

https://github.com/solemnwarning/rehex/

Web browser

NetSurf

While it sadly seems that Mozilla's Firefox, and its Gecko web engine, are experiencing challenging market conditions, we're still a long way from the dark days of many websites only working in Internet Explorer. This is mainly because the Blink web engine, used in Google Chrome, Microsoft Edge, and Opera, may have become dominant, and it is still open source, with a long open source origin story. Several projects, including favorites like Nyxt (formally Next) and qutebrowser, take advantage of this via QtWebEngine. While those big corporations undoubtedly wield a great deal of influence over Blink's future, we're equally free to fork it and change it however we wish.

However, monoculture is seldom a good thing, especially when it comes to the World Wide Web, and that's why other web engines – both big and small – are so important. One such nascent and diminutive engine is called Kosmonaut. It doesn't yet render pages fully, but its rapid Rust via OpenGL bindings development looks very promising. But Kosmonaut isn't alone in this space either. At the other end of the scale is NetSurf, an old browser that's still being updated and still sports its own entirely unique web engine. It's an application that started life on RISC OS in 2002, and its cross-platform credentials are still a huge part of its appeal. You can run NetSurf on an Amiga, an Atari ST, BeOS, and even Samsung TVs. As you'd expect of something that still runs on a Motorola 68000 processor, it's fast. Launching the latest release on a modern Linux machine is lightning fast, as is the layout and web rendering. There's

NetSurf on Linux uses a GTK front end, which has been overhauled for the latest release.

only one problem: Modern, complex websites look wrong. This is mostly due to NetSurf's "in-development" JavaScript handling, which still has a long way to go. As the text is still perfectly legible, however, this is often worth the trade-off.

Project Website

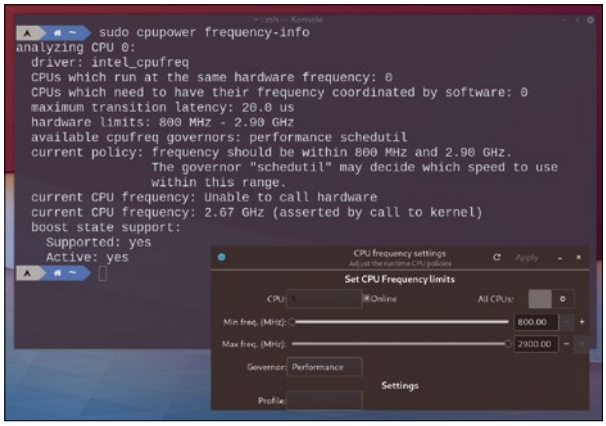
https://www.netsurf-browser.org/

CPU scaling

cpupower-gui

When CPUs were single core and either expensive or directly soldered onto the main board, many of us actually cared about what our CPUs were doing. This was because they couldn't do much, and every cycle counted. The humble Commodore 64 CPU was running under a single megahertz in PAL-inflicted regions, for example, while the Amiga 500 only just broke the 7MHz barrier – while its rival, the Atari ST, gloated from the realm of 8Mhz. However, the Amiga did have an important advantage. You could swap out the scheduler for something more aggressive or more multitasking friendly, depending on whether you were accessing a BBS or number crunching a scene in LightWave 3D. Of course, now speed isn't everything. Whether your CPU is clocked at 2, 3, 4, or 5Ghz, it's likely you have more than one core to parallelize tasks, and a big part of your CPU management is now about keeping the temperature down and the battery life maximized rather than getting things done faster.

But sometimes you still need to get things done faster, and you still need that hands-on control over how things are prioritized. But rather than swapping out the process scheduler, the best approach in Linux is to control how aggressive its CPU management is. This is known as CPU scaling, and it allows you to set a minimum and maximum CPU frequency, either for each CPU, or for all of them at the same time. You may want maximum battery life,



Scaling governors operate like presets for CPU management, varying between *Powersave* and *Performance* with *Userspace* for your own frequencies.

for instance, which means limiting them all to the lowest frequency. Or you may want maximum CPU speed all the time, or you may just want to tweak your distribution's default values. A common tool for setting these values is `cpupower`, but it can be tricky to use. Try this, `cpupower-gui`, instead. It's an easy to use portal to the same functionality where you can easily access the dark arts of CPU management while seeing the changes you're making.

Project Website

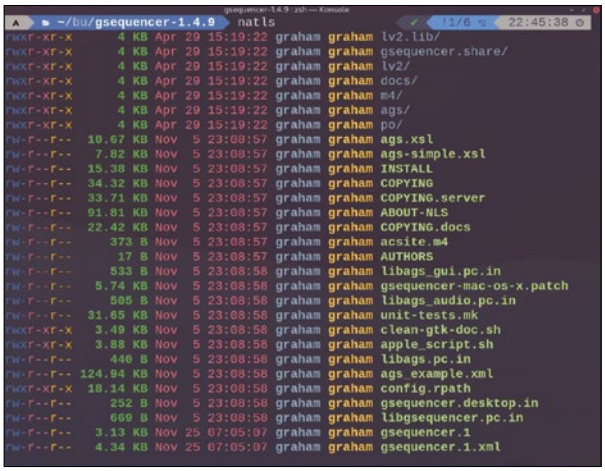
https://github.com/vagnum08/cpupower-gui

ls replacement

nat

It's a brave project that challenges the status quo, and few commands are as much a part of the status quo as the humble `ls`. For most of us, `ls` and its permutations have long been committed to command-line muscle memory. You type `ls` to see what's going on in the current directory, type `ls -l` to see file permissions and a little more detail, and type `ls -l --time-style=full-iso` to show off. And yet, this isn't enough for the nat project, which boldly proclaims itself "the better ls" before you even get to the documentation. Nat is indeed an `ls` alternative, written in the system language du jour, Rust. Typing `natls` is intended to be a more fulfilling experience than typing `ls`. Your terminal bursts with color (if your terminal supports it). There isn't just a different hue for the size, date, ownership, and file columns, there are also different colors for the user, group, and system permissions. It's like a psychedelic version of `ls -l`.

Alongside the colorful output, `natls` provides all the same functionality you'd expect from `ls`. This includes output showing file permissions, file sizes, modification dates, and group and user ownership. These options are also well organized, with flags to disable them individually in the output, turn off sorting, and fit everything into as wide a space as your terminal can spare. It also uses a sensible time format, which can be easily changed with another argument. But the best thing about



Make `ls` a thing of the past by making your own modifications to the excellent `natls`.

natls is that most of its Rust source code can be found in a single, manageable, file, which makes it incredibly easy to understand and modify. This makes the project much easier to modify than the original `ls`, and it could be an ideal starting point to create your own perfect `ls` replacement while also learning about a strong contender for the next standard system programming language.

Project Website

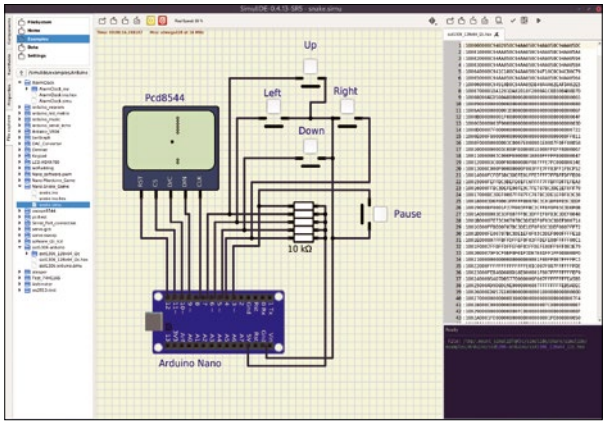
https://github.com/willdoescode/nat

Circuit simulator

SimulIDE

It seems we can't go a month without stumbling on another new or rebooted circuit-building tool. We recently looked at KTechLab, for instance, and several more in the months before that. This time it's the turn of SimulIDE, a modest and relatively easy-to-use application that can help you build and test your own circuits. Despite being built with a modern Qt version, SimulIDE has a distinctly mid-'80s Macintosh feel to its user interface. There are two reasons for this. The first is that the huge set of monochrome icons on the left shares a similar aesthetic with those early applications. The icons themselves represent the various components, generators, and annotations you can use in your own circuits, all of which are fortunately more modern than Aldus PageMaker. The second reason for this retro vibe is the grid-type background to the right. This is where you drag and connect your components into a circuit.

One of the best things about SimulIDE is that there's a set of integrated tutorials to help you get started, not just with the application, but with designing your own circuits. These tutorials guide you through the GUI and which components are available, as well as connecting them up and testing whether they work. As with KTechLab, this is the best part, because SimulIDE lets you simulate the circuit, complete with currents, meter readings, and working components. This even extends to microcontrollers like the Arduino and components like



SimulIDE helps you build your own circuits and emulates what happens when they have power, including AVR and PIC simulation courtesy of simavr and gpsim respectively.

screens that require firmware. Common open source versions of the firmware are included, and you can view the hexadecimal data for these files in the viewer on the right. One of the best examples puts this all together into a working Arduino circuit, complete with buttons, resistors, and a screen, as well as the Arduino code to play the famous snake game, which itself runs on the simulator. It's brilliant and a lot less messy than doing it yourself.

Project Website

https://www.simulide.com

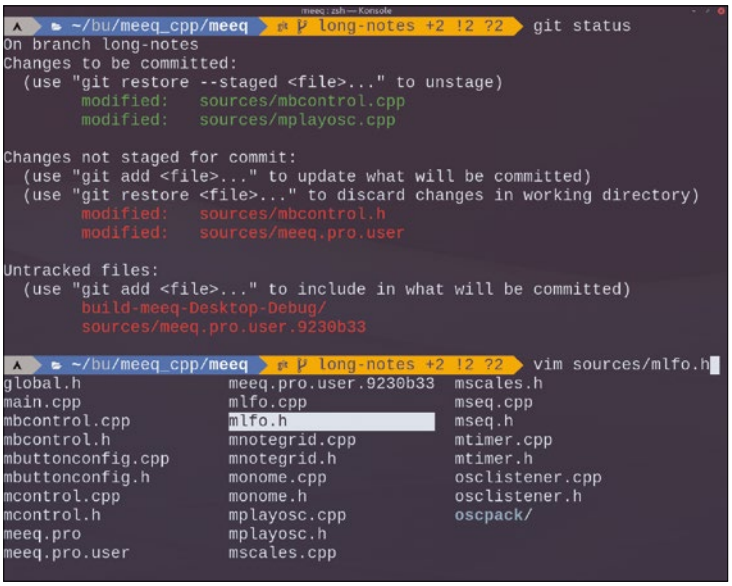
Shell upgrades

Oh My Zsh

To anyone from a desktop, phone, or tablet background, the humble command line can appear intimidating, best left to sci-fi nerds in fallacious hacking movies. But as many of us know, there's no reason to be intimidated at all. The command line is often the best and simplest way to get certain tasks done, from editing documents to accessing remote servers. However, like KDE Plasma, its default configuration on many distributions can be a little austere, lacking modern shell features and a modern look. All of this can be solved by switching your terminal first to Zsh and then to its wonderful community-driven Oh My Zsh framework.

If you've not already switched, Z shell (Zsh) is best described as an extended version of Bash, the terminal most distributions use by default. When using Zsh, you won't have to relearn anything, because it works in exactly the same way as Bash, while providing easily learned additional features. There's no need to use `cd`, for example, because you can switch directories by typing their names. Zsh will also correct many typos automatically and more intelligently complete local and remote commands with a press of the Tab key. Paths and variables can be expanded from single letters and symbols, and history navigation is much cleverer. It's also more readily themable, which is something Bash has always struggled with.

There are many projects that have attempted to coax some creative design ingenuity out of Bash. Perhaps the best of these is Powerline, a set of Python scripts that augment the humble Bash prompt with a colorful array of symbols, contextual information, and modifiable layouts. It works well but always feels patched atop Bash rather than part of it. Even worse is when it breaks after a Python upgrade, or when you try to migrate your Powerline configuration to a new machine. A big part of Zsh is its modularity and plugin mechanism, and this is exactly what the terribly named Oh My Zsh has been built to take advantage of. It's a framework that can make Zsh look and feel much like Bash with Powerline, but it does so far more elegantly while also offering dozens more themes and options.



Oh My Zsh and Zsh work together to make harnessing the power of Zsh easy and enjoyable.

The Powerlevel10k theme even includes its own binary to help with command-line configuration.

Oh My Zsh is easily installed and configured via the same `.zsh` configuration file you use for Zsh. The installer will even make all the necessary additions to get started quickly. All you often need to do is download and install an accompanying theme. Installation for everything with Oh My Zsh is usually via a simple `git` command and a good theme to start with is called Powerlevel10k. Not only does it look fantastic, but it also includes its own configuration executable. This asks you all kinds of questions about what you'd like to appear and where, alongside previews of how the changes will affect your command line. It allows you to automatically install whatever Zsh plugins are required to get exactly the result you want and far surpasses the capabilities of Powerline. It works best when combined with a few Oh My Zsh plugins, of which there are hundreds, including various version control status updates, many command aliases and autocompletion profiles, Vim modes, and a Kubernetes prompt. This is what makes Zsh so powerful – it's still easy to use, but it's also easy to expand and fit your exact requirements.

Project Website

https://ohmyz.sh/

Strategy FPS

Unvanquished

If you've always wanted to play more of a support role in an FPS game, Unvanquished is the game for you.

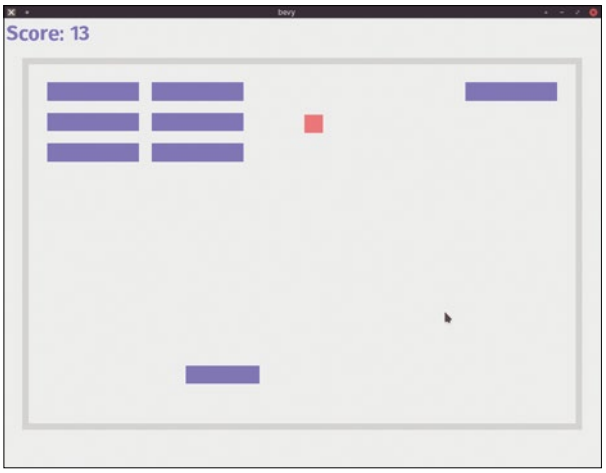
There are quite a few decent open source online multiplayer first person shooter (FPS) games for Linux, such as Red Eclipse and Warsow. One particularly interesting example is an older game called Tremulous, which combines twitchy and instinctive combat with real-time strategy elements. Players are either humans or aliens, and they can either attack the opposition directly or choose to stay closer to home and build up the local base. This base building is much like you'd find in a real-time strategy game, fueled by the number of "frags" your whole team can score. As your team progresses, more options are unlocked, and theoretically your shooters have a stronger advantage. Tremulous hasn't been updated for many years, but its unique gameplay, and even map compatibility, has been transplanted into another game engine and reincarnated as a new game called Unvanquished.

Unvanquished is built from scratch using the Daemon game engine, an engine itself forked from the OpenWolf Engine. Despite the significant challenge of re-engineering the original gameplay, Unvanquished offers a bona fide Tremulous experience encased within beautiful, shaded, high frame rate graphics. The real achievement is that this conversion includes proper open-sourced assets, which is itself the result of a process that started in 2015 with the development of asset management tools. This means the engine, the game code, the graphics, the textures, the sounds, and the music are all free with a capital F, which is both highly unusual and highly commendable. It means game developers have a huge pool of first-class content that they can use when building their own games or when creating any kind of media. And of course, we also have the game, which is itself a wonderful multiplayer creation that combines some of the best elements of both FPSs and real-time strategy.

Project Website
https://unvanquished.net/

Games engine

Bevy



Game building in Bevy uses the Entity Component System (ECS), which is sadly nothing to do with the ECS chipset on the Commodore Amiga.

There are many open source game engines. GamingOnLinux.com lists 21, from the amazingly comprehensive Godot to the humble Pygame. Bevy is a new addition, and if your ambitions are modest, it already offers a wonderful environment from which to build a cross-platform game. Unlike with Godot, though, Bevy development requires you to dive into the programming aspect from the beginning. This is because Bevy describes itself as a "data-driven" game engine using the Entity Component System (ECS) paradigm. This is the hot new thing in game development, and it is even being deeply explored by game engine behemoth Unity. Exploring Bevy's excellent documentation reveals ECS to be a programming methodology based on composing your game from entities, components, and systems. As the documentation puts it, "Entities are unique things that are assigned groups of Components, which are then processed using Systems."

In practice, this is considerably different from working with something like Pygame, especially when Bevy itself is written in, and dependent upon, the Rust programming language and framework. Rust is similar to C++, but also thoroughly modern in its design, making it safer and easier to work with. It also helps to make the ECS implementation in Bevy relatively easy to understand and use, thanks to it using normal Rust data types rather than new complex ECS-only types. This is all covered in the excellent beginner's tutorial, and it does a great job of introducing these concepts and showing how advantageous they can be in writing performant code. If the included examples are anything to go by, the results can be seriously impressive, with both 2D and 3D projects. When this is tied to Rust's super fast compile times, hot reloading of changed assets without restarting your game, and Rust's excellent integration into tools like Visual Studio Code, Bevy feels like the beginning of something revolutionary in game design.

Project Website
https://bevyengine.org/

A tool for distro hoppers

Saving Steps

With Ventoy you can conveniently put multiple bootable ISO images on a single USB drive. BY CHRISTOPH LANGNER

The Linux community teems with "distro hoppers," users who are always trying out new distributions. There is nothing at all wrong with that; we are always happy to discover new treasures in the Linux universe. Nevertheless, distro hopping can be hard work. Just downloading and transferring the ISO images to a USB memory stick takes time and involves risks. A simple typo in a command might cause you to move the image to the wrong place by mistake and accidentally delete important data.

Wouldn't it be far simpler if you could simply copy an ISO image off the web to a USB stick with a file manager and boot directly from the drive? Ideally, such a Linux USB memory stick would not only carry a distribution but even offer a selection of systems. For example, you could quickly demonstrate the differences between Ubuntu, Fedora, and Manjaro to a friend without having to prepare and carry around several USB drives. The Ventoy boot manager offers precisely this solution [1].

Installing Ventoy

Ventoy is available for both Linux and Windows. On Linux, however, the program is currently only available in Arch Linux and its Manjaro derivative. Ventoy is found in the Arch User Repository (AUR). To install, you will therefore need to use an AUR helper like Yay, where you load the program into your system by typing `yay -S ventoy`.

For other distributions, use the release packages provided by the developers [2]. To install, download the `ventoy-<version>-linux.tar.gz` archive from the project's GitHub page, unpack the tarball, and then run the `./Ventoy2Disk.sh` command. Ventoy shows the structure of its syntax and an explanation of the individual parameters. Listing 1 shows the process for the 1.0.18 version, current at the time of writing.

Preparing a USB Memory Stick

Even with Ventoy, you can't avoid formatting a USB memory stick, hard disk, or SSD. However, you only need to perform the operation once. Plug the Ventoy USB drive into the computer and use `lsblk` or graphical tools such as the Gnome desktop disk manager (Figure 1) to discover the device ID of the Ventoy USB memory stick. Listing 2 shows the `lsblk` output in a terminal window. The correct device ID can be found using the information on the size of the disk in the fourth column.

Armed with this information, then change back in the terminal to the folder in which you unpacked Ventoy. When you get there, call the program with root privileges (Listing 2, last line). Pass in the `-i` (install) option and the device ID of the USB drive. To be on the safe side, Ventoy displays the

Listing 1: Ventoy2Disk

```
$ tar xzf ventoy-1.0.18-linux.tar.gz
$ cd ventoy-1.0.18
$ ./Ventoy2Disk.sh
**********************************************************
*                Ventoy2Disk Script                      *
*             longpanda  admin@ventoy.net                *
**********************************************************

Usage:  Ventoy2Disk.sh CMD [ OPTION ] /dev/sdX
  CMD:
   -i  install ventoy to sdX (fail if disk already installed with ventoy)
   -I  force install ventoy to sdX (no matter installed or not)
   -u  update ventoy in sdX

  OPTION: (optional)
   -r SIZE_MB  preserve some space at the bottom of the disk (only for install)
   -s          enable secure boot support (default is disabled)
   -g          use GPT partition style, default is MBR (only for install)
```

most important information again, such as the device name and the size of the selected data carrier (Figure 2). In this way you can quickly see whether you accidentally specified a hard disk that is normally used for backups.

The Ventoy installation routine now automatically partitions the disk. A small partition is used by the boot manager to start the system; you cannot and never should change anything here. The program searches for ISO images on the partition labeled `ventoy` and formatted with exFAT. It does not matter how you organize the images there. You can move the ISO files directly into the root folder or organize them in subfolders. The only conditions are that the file names can only contain ASCII characters and spaces can't be used.

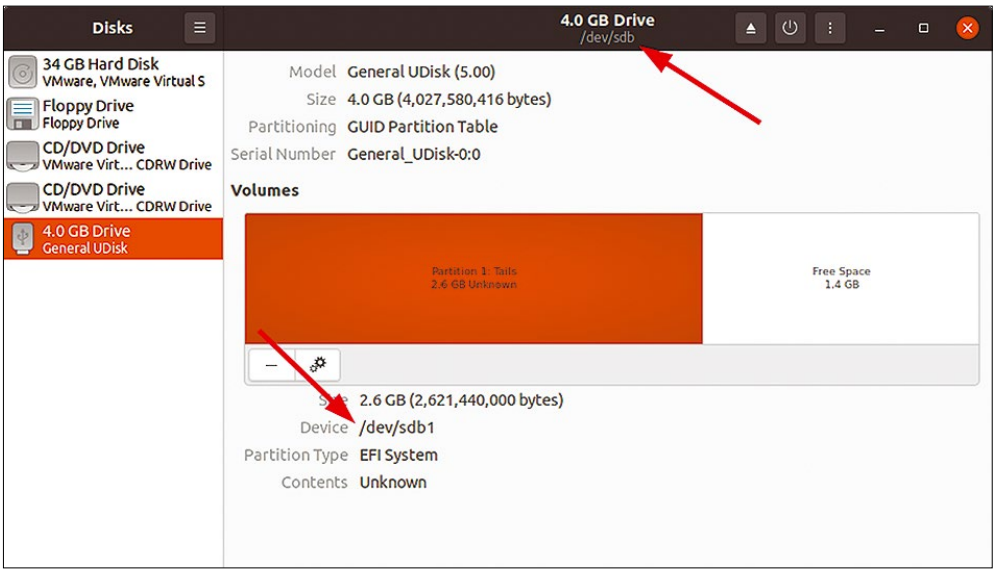


Figure 1: You can discover the USB memory stick's device ID to be written to at the command line with `lsblk` or, as shown here, with the Gnome desktop disk manager.

Both the developers and the Ventoy community are constantly testing current distributions for compatibility [3]. A wide range of older and recent distros has already been tested, from Debian, Ubuntu, and Fedora through to more exotic candidates such as Zorin, ArcoLinux, and Untangle. The project also supports booting from Windows images and FreeBSD-based systems, including pfSense, GhostBSD, and FreeNAS (see also the box "Secure Boot").

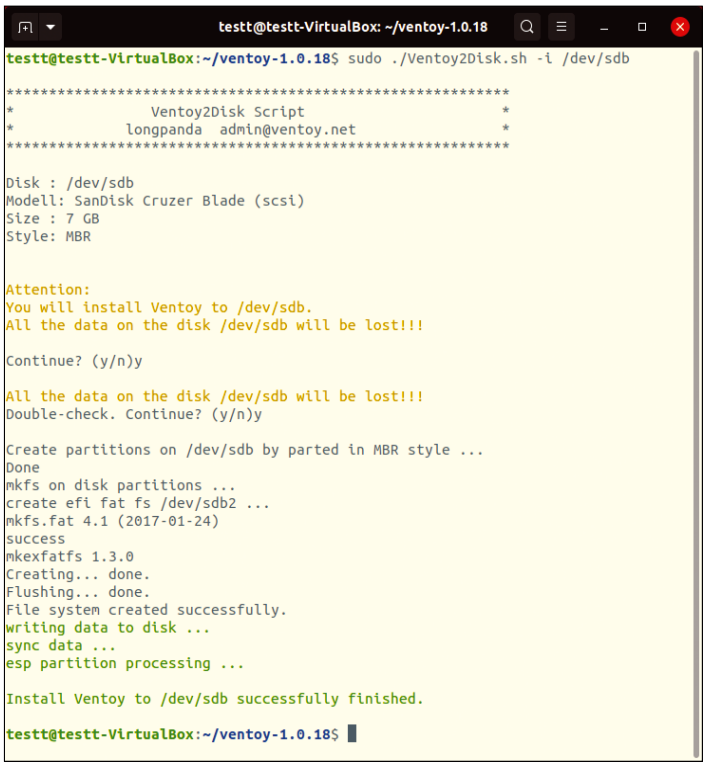


Figure 2: The command-line tool `Ventoy2Disk.sh` lets you format a USB memory stick with Ventoy. Afterwards you only need to copy the ISO images to be booted onto the stick.

Listing 2: lsblk Output

```
$ lsblk
NAME     MAJ:MIN RM   SIZE RO TYPE MOUNTPOINT
sda        8:0    0 232,9G  0 disk
|-sda1     8:1    0 232,9G  0 part /home
[...]
sdc        8:32   1   7,5G  0 disk
|-sdc1     8:33   1   256M  0 part /run/media/toff/boot
|-sdc2     8:34   1   7,2G  0 part /run/media/toff/rootfs
[...]
$ sudo ./Ventoy2Disk.sh -i /dev/sdc
```

Secure Boot

As of version 1.0.07, Ventoy also supports booting on computers where the Secure Boot mechanism is activated. When formatting the Ventoy USB memory stick, you need to use the `-s` option to do this:

```
$ sudo ./Ventoy2Disk.sh -s -i /dev/sd<X>
```

You also need to import the shim UEFI key when you first boot from the Ventoy stick. In the documentation, the developers explain the process, but also recommend disabling Secure Boot in UEFI in case of problems [4].

Booting with Ventoy

To boot a Linux distribution with Ventoy, download its ISO image off the web and copy it to the Ventoy partition. This works on Linux as well as on macOS or Windows, because Ventoy uses exFAT for the data partition. You do not need special rights or special programs; a file manager is fine. After the copy process is complete, remove the USB memory stick from the prep computer and boot the target computer with it. Like with any normal Linux image, you need to be sure to boot from the USB drive. Usually you change the boot medium in the UEFI/BIOS or via a boot menu, which can often be enabled by pressing Esc or F12.

Figure 3: Ventoy in use: We simply used the file manager to copy the Linux images to the Ventoy partition on the USB memory stick.

Once Ventoy has booted, you now have the option to boot the images copied to the data partition yourself (Figure 3). Use the up and down arrow keys to navigate in the menu, and press Enter to launch the selected image. Ventoy displays all images found on the data partition on one level. If necessary, you can press F3 to enable a tree view. Use Enter to select directories or Esc to move up one level. The menu is updated automatically during the next boot process if you copy more ISO images to the partition or delete existing ones.

Updates and Persistence

If you repeat the installation routine, the Ventoy2Disk script will abort with the message that Ventoy is already installed on the USB stick. However, by using the switches `-u` and `-I` (i.e., say, `sudo ./Ventoy2Disk.sh -u /dev/sd<X>`), Ventoy gives you the option of updating Ventoy on the USB stick (`-u`, preserving the images) or completely reformatting the stick with Ventoy and deleting all data (`-I`).

Ventoy can also be extended to include additional functions thanks to a plugin system. For example, the program supports the installation of your own themes to change the look or the contents of the

boot manager [5]. Since Ventoy itself is based on GRUB 2, there is a wide selection on the web [6]. Further plugins allow the integration of logos, titles instead of file names in the menus, and the ability to add descriptions. You can also automate the installation process for operating systems with your own scripts (Kickstart for RHEL or Fedora, Preseed for Debian or Ubuntu, AutoYaST for SUSE).

The Persistence plugin is also interesting; it lets users define a separate image file as nonvolatile memory [7]. This means that you can work with a distribution you booted started as a Live system as if it were permanently installed on the system. All data and settings are kept on restarting. The distributions tested for this operating mode include Ubuntu, Linux Mint, elementary OS, and Zorin. Persistence images are either created with the `CreatePersistentImg.sh` script, or you can download a prepared image from the project website.

To configure the plugins, you first need to create a `ventoy/` subfolder on the Ventoy stick's data partition or create a file there named `ventoy.json`. Then enter the desired settings in line with the specifications in the documentation. The example in Listing 3 activates the Tela theme [8], which I copied to `ventoy/theme/tela/`, defines aliases for the three Linux ISOs stored on the stick, and defines nonvolatile memory for the Ubuntu image (Figure 4). The root folder of the USB memory stick acts as the root for all paths. If a configuration does not work, the error can be analyzed using Ventoy's debug mode, which you enable by pressing F5.

Conclusions

Ventoy proves to be an extremely practical tool for users who frequently try out new distributions. A USB memory stick, once equipped with a boot manager, removes the need to "burn" images in the future. To boot a new distribution, you simply move the ISO file there using your choice of file manager. This saves time, can also be done on Windows, and reduces the risk of accidentally overwriting a disk. This makes Ventoy ideal for sharing your favorite new distributions. For example, you could present a colorful bouquet of distributions to the participants at an installation party. ■■■

Figure 4: Ventoy with the Tela GRUB theme courtesy of *Gnome-look.org* and aliases for the Linux images on the USB memory stick.

Listing 3: Tela Theme

```
{
  "theme": {
    "file": "/ventoy/theme/tela/theme.txt",
    "gfxmode": "1920x1080"
  },

  "menu_alias": [
    {
      "image": "/images/ubuntu-20.04-desktop-amd64.iso",
      "alias": "Ubuntu 20.04 LTS"
    },
    {
      "image": "/images/tahr64-6.0.5.iso",
      "alias": "Puppy Linux: Tahrpup64 6.0.5"
    },
    {
      "image": "/images/manjaro-gnome-20.0.3-minimal-200606-linux56.iso",
      "alias": "Manjaro Gnome 20.0.3 Minimal"
    }
  ],

  "persistence": [
     {
       "image": "/images/ubuntu-20.04-desktop-amd64.iso",
       "backend": "/persistence/ubuntu-20.04-desktop-amd64.img"
     }
  ]
}
```

Info

[1] Ventoy: *https://www.ventoy.net*

[2] Releases: *https://github.com/ventoy/Ventoy/releases*

[3] Tested distributions: *https://www.ventoy.net/en/isolist.html*

[4] Secure Boot: *https://www.ventoy.net/en/doc_secure.html*

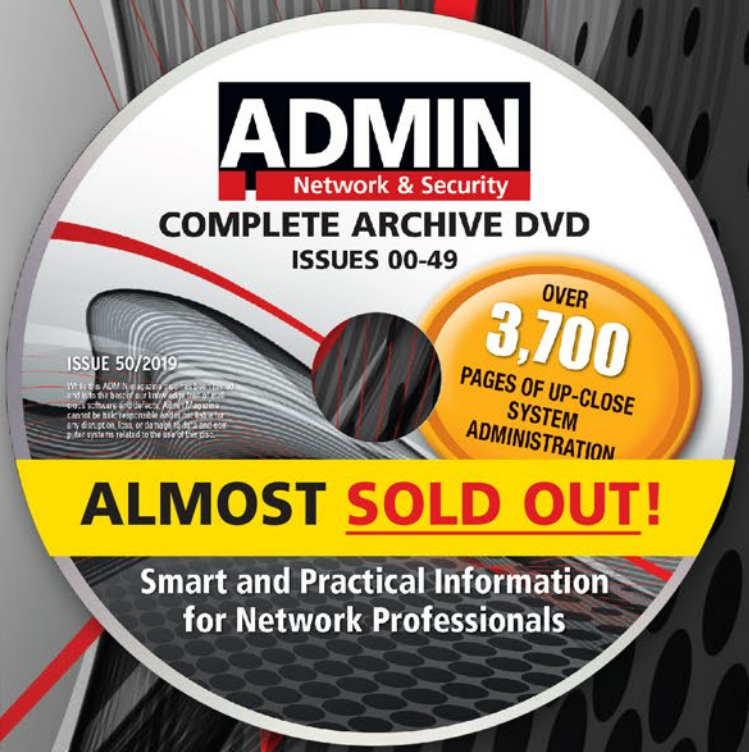
[5] Ventoy theme plugin: *https://www.ventoy.net/en/plugin_theme.html*

[6] Themes for GRUB 2: *https://www.gnome-look.org/browse/cat/109/order/latest/*

[7] Ventoy Persistence plugin: *https://www.ventoy.net/en/plugin_persistence.html*

[8] Tela theme for GRUB 2: *https://www.gnome-look.org/p/1307852/*

9 Years of ADMIN on One DVD



ALMOST SOLD OUT!

Smart and Practical Information for Network Professionals

This searchable DVD gives you 50 issues of ADMIN, the #1 source for:

- network security
- system management
- troubleshooting
- performance tuning
- virtualization
- cloud computing

Clear off your bookshelf and complete your ADMIN library with this powerful DVD!

ORDER NOW!

shop.linuxnewmedia.com

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.

#242/January 2021

3D Printing

The weird, wonderful, futuristic world of 3D printing is waiting for you right now if you're willing to invest a little time and energy. This month we help you get started with practical 3D printing in Linux.

On the DVD: Ubuntu 20.10 "Groovy Gorilla" and Fedora 33 Workstation

#241/December 2020

Secure Your System

Security often means sophisticated tools like firewalls and intrusion detection systems, but you can also do a lot with some common-sense configuration. This month we study some simple steps for securing your Linux.

On the DVD: KDE neon 5.20.0 and elementary OS 5.2

#240/November 2020

It's Alive

Build a whole operating system? Well maybe not a Linux, but if you're interested, you can implement the basic features of an experimental OS using resources available online. We can't show you the whole process, but we'll help you get organized and take your first steps.

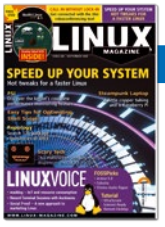
On the DVD: Linux Magazine 20th Anniversary Archive DVD

#239/October 2020

Build an IRC Bot

IRC bots do the essential work of coordinating and forwarding chat messages on the Internet. This month we show you how to build your own custom bot – and we give you an inside look at how to work directly with IRC.

On the DVD: Debian 10.5 and Devuan 3.0



#238/September 2020

Speed Up Your System

Your Linux experience goes much more smoothly if your system is running at peak performance. This month we focus on some timely tuning techniques, including the kernel's new Pressure Stall Information (PSI) feature.

On the DVD: Bodhi Linux 5.1 and openSUSE Leap 15.2

#237/August 2020

Webcams and Linux

This month we explore webcams, screencasting, and a cool teleprompter tool. We also look at PHP Building Blocks, Guacamole the clientless remote access tool based on HTML5, and a study of the handy MystiQ Audio/Video conversion tool.

On the DVD: Ubuntu Studio 20.04 and Kubuntu 20.04

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 *events@linux-magazine.com*.

NOTICE

Be sure to check the event website before booking any travel, as many events are being canceled or converted to virtual events due to the effects of COVID-19.

FOSDEM '21

Date: February 6-7, 2021

Location: Virtual Event

Website: *https://fosdem.org/2021/*

FOSDEM is a free event for software developers to meet, share ideas, and collaborate. Every year, thousands of developers of free and open source software from all over the world gather at the event in Brussels. In 2021, they will gather online.

CloudFest 2021

Date: March 23-25, 2021

Location: Virtual Event

Website: *https://www.cloudfest.com/*

CloudFest returns on an all-digital platform amid a global pandemic as we take stock of the strengths, weaknesses, risks, and opportunities that are revealed by a globe-spanning threat like COVID-19. Join us at CloudFest 2021, and let's build something great together.

Events

FOSDEM 2021	February 6-7	Virtual Event	https://fosdem.org/2021/
CloudFest 2021	March 23-25	Virtual Event	https://www.cloudfest.com/
Kubernetes Community Days	April 8-9	Amsterdam, Netherlands	https://sessionize.com/kcdams2021/
DevOpsCon London Hybrid Edition	April 20-23	London, UK and Online	https://devopscon.io/london/
KubeCon + CloudNativeCon	May 5-7	Virtual Event	https://bit.ly/kube-cloudnativecon
Linux Storage Filesystem & MM Summit	May 12-14	Palm Springs, California	https://events.linuxfoundation.org/lsfmm/
LISA21	June 1-3	Anaheim, California	https://www.usenix.org/conference/lisa21
SYSTOR 2021 Hybrid	June 14-18	Haifa, Israel	https://www.systor.org/2021/venue.html
stackconf online 2021	June 15-16	Virtual Event	https://stackconf.eu/
ISC High Performance 2021 Digital	June 24-July 2	Virtual Event	https://www.isc-hpc.com/
Cloud Expo Europe	July 7-8	London, United Kingdom	https://www.cloudexpoeurope.com/
USENIX ATC '21	July 14-16	Santa Clara, California	https://www.usenix.org/conference/atc21
KVM Forum	August 2-4	Vancouver, British Columbia	https://events.linuxfoundation.org/
Embedded Linux Conference North America	August 4-6	Vancouver, British Columbia	https://events.linuxfoundation.org/
Open Source Summit North America	August 4-6	Vancouver, British Columbia	https://events.linuxfoundation.org/
Embedded Linux Conference Europe	September 29-Oct 1	Dublin, Ireland	https://events.linuxfoundation.org/
Open Source Summit Europe	September 29-Oct 1	Dublin, Ireland	https://events.linuxfoundation.org/

CALL FOR PAPERS

We are always looking for good articles on Linux and the tools of the Linux environment. Although we will consider any topic, the following themes are of special interest:

- System administration
- Useful tips and tools
- Security, both news and techniques
- Product reviews, especially from real-world experience
- Community news and projects

If you have an idea, send a proposal with an outline, an estimate of the length, a description of your background, and contact information to *edit@linux-magazine.com*.

The technical level of the article should be consistent with what you normally read in *Linux Magazine*. Remember that *Linux Magazine* is read in many countries, and your article may be translated into one of our sister publications. Therefore, it is best to avoid using slang and idioms that might not be understood by all readers.

Be careful when referring to dates or events in the future. Many weeks could pass between your manuscript submission and the final copy reaching the reader's hands. When submitting proposals or manuscripts, please use a subject line in your email message that helps us identify your message as an article proposal. Screenshots and other supporting materials are always welcome.

Additional information is available at:
http://www.linux-magazine.com/contact/write_for_us.

NOW PRINTED ON recycled paper from 100% post-consumer waste; no chlorine bleach is used in the production process.

Authors

Contact Info

Editor in Chief
Joe Casad, jcasad@linux-magazine.com

Copy Editors
Amy Pettle, Megan Phelps

News Editor
Jack Wallen

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
© Elnur Amikishiyev, 123RF.com

Advertising
Brian Osborn, bosborn@linuxnewmedia.com
phone +49 89 3090 5128

Marketing Communications
Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
2721 W 6th St, Ste D
Lawrence, KS 66049 USA

Publisher
Brian Osborn

Customer Service / Subscription
For USA and Canada:
Email: cs@linuxpromagazine.com
Phone: 1-866-247-2802
(Toll Free from the US and Canada)

For all other countries:
Email: subs@linux-magazine.com

www.linuxpromagazine.com – North America
www.linux-magazine.com – Worldwide

Printed in Nuremberg, Germany by hofmann infocom GmbH on recycled paper from 100% post-consumer waste; no chlorine bleach is used in the production process.

Distributed by Seymour Distribution Ltd, United Kingdom

LINUX PRO MAGAZINE (ISSN 1752-9050) is published monthly by Linux New Media USA, LLC, 2721 W 6th St, Ste D, Lawrence, KS, 66049, USA. Periodicals Postage paid at Lawrence, KS and additional mailing offices. Ride-Along Enclosed. POSTMASTER: Please send address changes to Linux Pro Magazine, 2721 W 6th St, Ste D, Lawrence, KS 66049, USA.

Published monthly in Europe as Linux Magazine (ISSN 1471-5678) by: Sparkhaus Media GmbH, Zieblandstr. 1, 80799 Munich, Germany.

Issue 244 / March 2021

Stream Processing

Next month we explore a new approach to data for a new world of constant motion.

Approximate On Sale Date

UK / Europe	Feb 06
USA / Canada	Mar 05
Australia	Apr 05

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***

Secure and Private

All systems by TUXEDO Computers come ready to start with Linux. In addition to impressive performance, they offer comprehensive protection of your personal data and strong protection of your privacy.

Privacy+
Intel ME, webcam, audio and WiFi deactivatable

Fully encrypted
System and data completely protected against unauthorized access

Zero Spyware
Verifiable security thanks to Open Source software

Automatic Installation
System reset thanks to web-based, fully automatic installation

100% Linux

5 Year Warranty

Lifetime Support

Built in Germany

German Privacy

Local Support

TUXEDO COMPUTERS

tuxedocomputers.com

HETZNER

DEDICATED ROOT SERVER
DESIGNED FOR PROFESSIONALS

Dedicated Root Server AX41-NVMe

- ✔ AMD Ryzen 5 3600 Simultaneous Multithreading
- ✔ 64 GB DDR4 RAM
- ✔ 2 x 512 GB NVMe SSD
- ✔ 100 GB Backup Space
- ✔ Traffic unlimited
- ✔ Location Finland and Germany
- ✔ No minimum contract
- ✔ Setup Fee £35.50

monthly from £31

Dedicated Root Server AX51-NVMe

- ✔ AMD Ryzen 7 3700X Simultaneous Multithreading
- ✔ 64 GB DDR4 ECC RAM
- ✔ 2 x 1 TB NVMe SSD
- ✔ 100 GB Backup Space
- ✔ Traffic unlimited
- ✔ Location Finland and Germany
- ✔ No minimum contract
- ✔ Setup Fee £54.00

monthly from £49

www.hetzner.com