GalliumOS
Keep your Chromebook running after the expiration date!

LINUX PRO MAGAZINE

ISSUE 251 – OCTOBER 2021

Linux from Scratch
Expand your Linux knowledge with this classic tool

Safe Eyes
Prevent eye strain with timely reminders

Wekan
Tune up your team with this helpful process management app

Regex Generators
Save time and simplify your searches

Build a Kitchen Kiosk
from an old e-reader

11 FOSS FINDS
COOL TOOLS FOR GEEKS!

LINUX NEW MEDIA
The Pulse of Open Source

Meet Me in St. Louis.

HPC is fueling breakthroughs across industries in science and beyond, from academia to commercial enterprises, by harnessing data to unlock insights faster and more accurately than ever before. HPC powers new capabilities in artificial intelligence and machine learning that, combined with modeling and simulation, accelerate discovery in solving our toughest challenges.

Join us in St. Louis to learn how HPC is empowering innovations that were never before possible to improve everyday life across the globe. Register early and save!

Program
14–19
NOVEMBER

Exhibits
15–18
NOVEMBER

The International Conference for High Performance Computing, Networking, Storage, and Analysis

SC21
St. Louis, MO | science & beyond.

Register Today!

sc21.supercomputing.org

Sponsored by: acm sighpc | IEEE COMPUTER SOCIETY TCHPC

COOKIE FIGHT

Dear Reader,

Back in the May 2021 issue, I reported on Google's effort to replace third-party tracking cookies with a new alternative they call Federated Learning of Cohorts (FLoC). The company's FLoC initiative replaces tracking cookies with an alternative technology that removes the power of third-party vendors while still sustaining, and perhaps enhancing, Google's own ability to target ads for ad buyers. New developments have added new folds of interest to this story. Time will tell whether these events will actually make a difference to the browser industry, but they are certainly worthy of attention.

First of all, Google announced a delay in their planned deployment of FLoC. Several major websites and ad vendors have denounced FLoC as a bad idea and a naked power grab by Google, and some commentators have speculated that the negative response is one of the reasons for the delay. It is unclear what they think one year would do to improve this negative response, unless Google is actually redesigning the system to appease its critics, which doesn't sound like Google. The other reason for the delay is that the EU has launched an antitrust investigation into Google's plan. The regulators' argument is that Google is an ad vendor, and Google the ad vendor should not be able to use the market power of the Google Chrome browser to enhance its position and eliminate rivals. (If you think this sounds a lot like the antitrust claims against Microsoft back in the 1990s, you are correct.)

Google is quite capable of pulling the plug on projects they don't think have a future – who remembers Google Latitude, Google Spaces, Google Wave? – but in this case, they just announced a delay, which means they still have an intention to proceed with it.

Meanwhile, as Google pauses and waits for the unfolding of its grand plan, Firefox is pressing ahead with its own, very different vision of cookie control. On August 10, Mozilla announced a new feature for Firefox 91 known as Enhanced Cookie Clearing. This new feature is part of a larger initiative called Total Cookie Protection, which was unveiled earlier this year. Total Cookie Protection confines all cookies from a single website into a separate *cookie jar* associated with the site. According to Mozilla, "In combination with the Supercookie Protections we announced [recently], Total Cookie Protection provides comprehensive partitioning of cookies and other site data between websites in Firefox. Together these features prevent websites from being able to 'tag' your browser, thereby eliminating the most pervasive cross-site tracking technique."

The new Enhanced Cookie Clearing option allows the user to zap everything in the cookie jar, clearing all cookies associated with a website all at once. It seems that trackers have started to store identifiers in increasingly obscure places to avoid cookie protections, including "Flash storage, Etags, and HSTS flags." Total Cookie Protection and Enhanced Cookie Clearing help to prevent these kinds of detection avoidance techniques that other cookie management systems might miss.

In the Firefox solution, third-party cookies still exist, but they can only be accessed from the site where they were placed, so they can't be used for tracking. On the other hand, the Google solution claims to eliminate third-party cookies, but it is far messier and more complicated than the Firefox method, and Google still gets to watch everything you do – they just store the data differently.

It will be interesting to see how a year's delay in Google's grand vision will play out against the rapid advance of Mozilla's grand vision. Keep in mind, also, that Apple has already blocked third-party cookies in the Safari browser and might be working on similar refinements to ensure a tighter seal. It is possible that, by the time the smoke clears from Google's antitrust review, the industry might have already moved on, and Google might have missed its opportunity to dictate the future of cookie protection. That's a big *if*, or course, because Google is still the biggest player in the eternal food fight for browser dominance, but we'll see how the cookie crumbles.

Joe

Joe Casad,
Editor in Chief

LINUX MAGAZINE

OCTOBER 2021

ON THE COVER

TWO TERRIFIC DISTROS
DOUBLE-SIDED DVD!
SEE PAGE 6 FOR DETAILS

Linux Mint 20.2 Cinnamon Edition and Garuda Linux KDE Dr460nized Edition

Two Terrific Distros on a Double-Sided DVD!

Linux Mint 20.2 Cinnamon Edition

64-bit

Linux Mint is one of the most popular Debian and Ubuntu derivatives. Cinnamon is Mint's own desktop and one of the most innovative desktops for Linux. The 20.2 release is a Long Term Support (LTS) release, which will receive updates and bug fixes until 2025.

As often happens with LTS releases, 20.2 contains numerous new or enhanced features. These include Bulky, a mass file renamer, plus an Android app that supplements Warpinator, enabling easy file transfer to phones and tablets. Another useful addition is the ability of the file manager Nemo to search for content as well as file names. Many new features are found in the Update Manager, which now supports Spices updates (i.e., updates for applets, desklets, themes, and extensions), as well as updates for universal Flatpak packages, which were previously accessed from Startup Applications. Just as usefully, the Update Manager also records how long each update has been available and suggests whether or not users might like a reminder to update. Still other options are to set conditions for when a notification should display the first time and to dismiss a notification for two days. The overall result is easier control of updates and their notifications

Linux Mint's Cinnamon edition is the perfect choice for users of all levels, with a constantly improving interface and a solid foundation in Debian. The 20.2 release should prove no exception.

Garuda Linux KDE Dr460nized Edition

64-bit

Based in India, Garuda Linux is a newer distribution based on Arch Linux by way of Manjaro. It's designed to achieve several well-defined goals that together add up to a unique distribution.

To start with, one of Garuda's goals is ease of use. To this end, it includes handpicked themes and its own line of graphical setting tools, including windows for GRUB options and network assistance. At the same time, another emphasis is on performance, with Btrfs as the default filesystem, `zstd` for file compression, and the zram kernel modules, which create a ramdisk for increased speed. The emphasis on performance also includes Garuda's own Timeshift, a utility that saves up to five system snapshots and is partly intended to reduce the problems that occasionally crop up with a rolling release. Still another goal is gaming, as shown by a gaming-optimized kernel and the distro's repository for games.

These goals add up to a distribution with something for everybody. Garuda does seem, however, best suited to mid-level users, similar to Manjaro without being a derivative. It is also a suitable starting point for anyone who wants a friendly introduction to the world of Arch Linux.

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

jax LONDON HYBRID

THE CONFERENCE FOR JAVA & SOFTWARE INNOVATION

OCTOBER 4 – 7, 2021
LONDON OR ONLINE

Use **LMcode15** to save an **extra 15%!**

REGISTER NOW!

CONFERENCE TRACKS

- Java Core & JVM Languages
- DevOps & Continuous Delivery
- Java & Blockchain
- Cloud, Kubernetes & Serverless
- Microservices
- Software Architecture & Design
- Agile & Culture

jaxlondon.com

NEWS

Updates on technologies, trends, and tools

Elementary OS 6 Odin Now Available

Elementary OS has always been a Linux distribution that takes the user experience seriously. And with the latest release (version 6, aka Odin), they take that to a new level. Users will now be able to more easily select between light and dark themes and even choose the accent colors to make the UI even more personal. In fact, version 6 is the most customizable release from the elementary designers and developers. This was made possible by a complete redesign and rewrite of the system stylesheet.

But this new release isn't all about style – there's plenty of substance. One of the many things the developers have done is leverage cutting-edge sandboxing technology to better enforce privacy and security. This release also makes use of Portals, which allows you to control how applications interact with one another (and your data). Within the Settings app, you'll find an Applications Permissions section, where you can control which apps have permissions for things such as the Home Folder, System Folders, Devices, Network, Bluetooth, Printing, Secure Shell Agent, and GPU Acceleration.

In addition, elementary OS 6 has gone all-in on Flatpak (a modern container format for application distribution). Many of the default elementary OS apps are now distributed as Flatpaks.

Finally, elementary OS enables multi-touch gestures out of the box, a feature that's long overdue and very welcome.

Find out more about elementary OS 6 from the official blog (*https://blog.elementary.io/elementary-os-6-odin-released/*) and download (*https://elementary.io/*) your copy of one of the most user-friendly Linux distributions available.

Kubuntu Focus Team Announces High-Performance Focus XE

From the makers of the Focus M2 (one of the hottest KDE-specific laptops on the market), comes the new Focus XE. And although it's focused on those looking for something small in form factor, it's certainly big in performance.

Sporting 11th generation CPUs and high-speed audio/data ports, this machine isn't an entry-level, budget-friendly laptop. The specs of the XE include a 4-core processor (Intel Core i7), up to 64GB of 3200 MHz RAM, up to 2TB NVMe storage, and Intel IRIS Xe Graphics. Other features include dual-band 5 GHz wireless, RJ-45 Gigabit Ethernet, Bluetooth dual version 5.2, and Thunderbolt v4. You'll also find two USB-A 3.2 GEN1 ports, and one USB-C 3.2 GEN2 port. An illuminated keyboard with 4mm of travel, a generous trackpad, a 1.0 MP HD webcam, and a

49 Whr Li-ION battery with FlexiCharge Battery Optimization (available in the BIOS), are all housed in a composite aluminum and plastic chassis.

The base model XE will run you USD $1,195, so this isn't a laptop that budget-minded users are going to seek out. But if you're looking for a sleek mobile workstation, that can be specced out to "beast mode" for $2,290 and enjoys the KDE desktop environment, the Focus XE might be just the machine for you.

Purchase your Focus XE from the Kfocus website (*https://kfocus.org/*).

Solus 4.3 Available for Download and Installation

Solus is the Linux distribution dedicated to the Budgie desktop. And this time around Budgie has received plenty of bug fixes and updates that add up to much-improved performance and reliability. These changes to the desktop environment also include new themes, window customizations, improved notifications, screen tracking, and more.

But the big additions come by way of the Linux 5.13 kernel. By shipping with this new kernel, Solus introduces support for Apple's M1 chipset, Intel's Alder Lake S Graphics, AMD's FreeSync/Adaptive-Sync, and a generic USB display driver. These additions mean Solus can run on even more hardware and will benefit from the performance gains offered by those chipsets and features.

If you opt to install Solus with the Gnome desktop, you'll be treated to version 40.2, which offers several new additions, such as tap-drag-release, horizontal workflow, improved gesture support for trackpads, and better keyboard shortcuts. Mutter (the window manager for Gnome) has also received numerous improvements, such as support for clipping (during background drawing) as well as scroll button locking. The Gnome developers have also resolved several X11 issues, such as unwanted position changes and window resizing during moving.

For those who opt to go the KDE route, you'll find the beautiful Plasma 5.22.2, which adds plenty of enhancements to the experience.

To get your copy of Solus 4.3, head over to the official download page (*https://getsol.us/download/*).

Steam Deck Linux-Powered Gaming System Set to Take Over the Handheld World

More than just a hand-held gaming system, the Steam Deck is a Linux-powered system with a KDE interface that can be docked and used as a regular PC. Steam Deck uses Proton as a compatibility layer to play Windows games on Linux, but users are free to replace it.

The device specs include an AMD 4-core Zen 2 CPU, an 8-core RDNA 2 graphics unit, 16 GB of memory, a 7-inch 1280x800-resolution touchscreen. As far as game control, Steam Deck includes several trackpads, thumbsticks, buttons, and triggers. A 40Wh battery is said to allow anywhere from two to eight hours of use. The device is charged via a single USB-C port that doubles as the means to connect the Steam Deck to external monitors and docks.

Steam Deck has three price points, based on different storage options: A base level of 64GB eMMC using PCIe Gen 2, a second-tier model with 256GB NVMe SSD using PCIe Gen 3, and a 512GB "high-speed" NVMe SDD that also uses PCIe Gen 3. The costs of the units (respectively) are $400, $530, and $650.

Valve plans on shipping the units in the US, Canada, EU, and the UK in December 2021. Reserve yours on the official Steam Deck site (*https://www.steamdeck.com/en/*).

MORE ONLINE

Linux Magazine
www.linux-magazine.com

Paw Prints • Jon "maddog" Hall
The Virus Rears its Ugly Head...
After months of machines being turned off, software licenses (with their expiration dates never "dormant") are up for renewal.

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

Prolog and Epilog Scripts
• Jeff Layton
HPC systems can benefit from administrator-defined prolog and epilog scripts.

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

Forensic Analysis with Autopsy and Sleuth Kit
• Matthias Wübbeling
Forensic admins can use the Autopsy digital forensics platform to perform an initial analysis of a failed system, looking for traces of an attempted attack.

IT Automation with SaltStack
• Robin Wittler and Tobias Jahnke
SaltStack is a fast and reliable modular toolbox written in Python that contains ready-made modules for many configuration management purposes.

HTTP/1.1 vs. HTTP/2 and HTTP/3
• Sandro Lucifora
HTTP/2 introduced multiplexing, resulting in superior bandwidth utilization over HTTP/1.1, and HTTP/3 solves the problem of transmission delays from packet loss by replacing TCP with QUIC.

Paragon NTFS Driver On Track For Upcoming Linux Kernel

Paragon submitted a read/write NTFS driver for the Linux kernel back in August 2020. At that time, the patch was refused because it was more than 27,000 lines of code. Since then, the company has submitted the patch in smaller chunks, which made it possible for the kernel maintainers to go through the code. This led to Linus responding (on *lore.kernel.org* – *https://lore.kernel.org/lkml/afd62ae457034c3fbc4f2d38408d359d@paragon-software.com/*) to say, "If the new NTFS code has acks from people - and it sounds like it did get them – and Paragon is expected to be the maintainer of it, then I think Paragon should just make a git pull request for it."

Paragon made it clear they would be maintaining the implementation, so it now looks as if the patch will make it into either 5.14 or 5.15.

This driver includes support for both normal and compressed files, supports journal replaying and full journaling support over JBD, and will be supported (via Paragon) once it's merged into the kernel. The patch does not, however, include all of the Paragon utilities. Regarding that, Paragon hints that there may still be a commercial version that will include everything.

The only caveat to this is that there are already more advanced filesystems available on the market. But with so many businesses still depending on NTFS, this patch should be a welcome addition to a lot of admins and companies.

LemonDuck Cryptomining Malware is Targeting Linux Systems

LemonDuck is a targeted attack that originally focused on vulnerabilities found in Microsoft's Exchange server to enable crypto mining on the compromised system. To make this attack even more vicious, LemonDuck removes other attackers from a compromised device to get rid of competing malware. This attack originally focused on China but has since begun targeting other countries (such as the US, Russia, Germany, the UK, India, Korea, Canada, France, and Vietnam).

© Nurul Fadiyah, 123RF.com

LemonDuck initially set its sights on Windows servers but has since expanded to Linux systems as well. On top of this, LemonDuck has expanded beyond crypto mining and can do things like send phishing emails, install backdoors, disable security controls, and steal credentials.

LemonDuck can spread via phishing emails, USB thumb drives, brute force attacks, and security exploits.

Microsoft's 365 Defender Threat Intelligence Team (*https://www.microsoft.com/security/blog/2021/07/22/when-coin-miners-evolve-part-1-exposing-lemon-duck-and-lemoncat-modern-mining-malware-infrastructure/*) had this to say about LemonDuck: "LemonDuck, an actively updated and robust malware that's primarily known for its botnet and cryptocurrency mining objectives, followed the same trajectory when it adopted more sophisticated behavior and escalated its operations."

Make sure you are following these CVEs to keep up on what's happening with this vulnerability: CVE-2017-0144 (EternalBlue), CVE-2017-8464 (LNK RCE), CVE-2019-0708 (BlueKeep), CVE-2020-0796 (SMBGhost), CVE-2021-26855 (ProxyLogon), CVE-2021-26857 (ProxyLogon), CVE-2021-26858 (ProxyLogon), CVE-2021-27065 (ProxyLogon).

Zack's Kernel News

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

The Patch Submission Process

Hans de Goede kicked off an interesting knowledge dump when he submitted a patch directly to Linus Torvalds for VBoxSF, the kernel's filesystem driver for Oracle's VirtualBox device.

Hans was frustrated by the lack of useful responses on the linux-fsdevel mailing list, especially because he was the VBoxSF maintainer and wanted to go through proper channels to keep his project up-to-date in the kernel. As a last resort, he sent the patch to Linus, hoping this might generate some movement.

However, Linus replied with the following useful explanation:

"The filesystem maintainer sending their patches to me as a pull request is actually the norm rather than the exception when it comes to filesystems.

"It's a bit different for drivers, but that's because while we have multiple filesystems, we have multiple _thousand_ drivers, so on the driver side I really don't want individual driver maintainers to all send me their individual pull requests – that just wouldn't scale.

"So for individual drivers, we have subsystem maintainers, but for individual filesystems we generally don't.

*"(When something then touches the *common* vfs code, that's a different thing – but [for] something like this vboxsf thing this pull request looks normal to me).*

"Even with a maintainer sending me pull requests I do obviously prefer to see indications that other people have acked/tested/reviewed the patches."

So Linus was happy to take this and future pull requests directly from Hans. But this was not the end of the story.

Al Viro, the Virtual Filesystem (VFS) maintainer, elaborated on Linus's statement that VFS-related patches were a different case. Al said, "there's one case when I want it to go through vfs.git, and that's when there's an interference between something going on in vfs.git and the work done in filesystem. Other than that, I'm perfectly fine with [the] maintainer sending pull request[s] directly to Linus (provided that I hadn't spotted something obviously wrong in the series, of course, but that's not 'I want it to go through vfs.git' – that's 'I don't want it in mainline until such and such bug is resolved')."

Al then clarified, adding, "Example: If there's a series changing calling conventions for some method brewing in vfs.git and changes to [the] filesystem's instance of that method in the filesystem tree. Then I'd rather it coordinated before either gets merged. It might be an invariant branch in either tree pulled by both, it might be a straight pull into vfs.git and sorting the things out there – depends upon the situation."

Randy Dunlap asked Al how developers should submit documentation changes for filesystems, and Al replied, "I'd been under [the] impression that kernel-doc stuff in general goes through akpm [Andrew Morton], TBH. I don't remember ever having a problem with your patches of that sort; I can grab that kind of stuff, but if there's an existing pipeline for that I'd just as well leave it there..."

The discussion ended up ranging over a wider terrain, with other filesystem maintainers asking about any special cases regarding their particular filesystems.

In general though, it seems that filesystems do have a pride-of-place, where maintainers can feel free to send their patches directly to Linus, as opposed to other drivers that should go through subsystem maintainers. The exception being filesystems that touch VFS code that affect other filesystems, in which case Al wants the code to go through his own tree first, to make sure there are no bad interactions before the maintainer sends in their final pull request.

An interesting thing for me is that this patch submission pipeline is not anything like a formal structure – most of the kernel development process is something that continues to evolve on its own over time, in response to the needs of the code itself. For example, Linus's rationale that, because there were relatively few filesystems, they could bypass the rigmarole that other drivers needed to go through. As these processes evolve, developers and even maintainers can find themselves slightly left behind, as Hans did. Then, as when Hans asked about it, the new current state of affairs can be clarified by people such as Linus and Al.

Even more interesting to me is that the clarification is itself a part of the evolution of the various processes. When someone such as Linus or Al needs to break out of their own ongoing work to answer a question like that posed by Hans, it could be the first time that they consider what would be best, relative to the current overall moment of development. And then their answer becomes the official position, where no such position existed for others or for themselves before the question was asked.

Status of NTFS

In the course of filesystem discussion, Rafal Milecki posted some marketing speak in favor of his company, Paragon Software, and then asked the best way to submit their filesystem, NTFS3, intended as an improvement over the Linux kernel's existing NTFS driver.

When Christoph Hellwig and Greg Kroah-Hartman asked why the code hadn't been posted for review in the recent past, Rafal blamed others, saying it was for "unknown reasons" and that there had been a lack of feedback from the community. Matthew Wilcox took extreme umbrage at that, pointing out that he had personally given feedback on Paragon's NTFS code, and adding that Paragon had done a very good job of responding to that feedback.

Not the best start for Rafal, but definitely not unheard of for people working on corporate contributions to the kernel, and absolutely recoverable. The underlying problem is that corporations tend to demand absolute tunnel vision and adulation from their employees, which can just look silly from the outside. But it's not silly! People need to make a living, even if it means acting pathologically. This aspect of corporate culture may also be one reason why the GNU General Public License is so needed in the world.

Anyway, Rafal corrected himself, saying that he'd only meant there had been a lack of feedback on the current patchset, and that he definitely appreciated the work people had done on the project.

Neal Gompa replied to Rafal, saying that he was highly in favor of Paragon's NTFS3 code getting into the kernel. Neal had tested the current patchset, as well as earlier iterations, and felt the code was definitely good enough to include in the kernel.

Neal added:

"I know that compared to all you awesome folks, I'm just a lowly user, but it's been frustrating to see nothing happen for months with something that has a seriously high impact for a lot of people.

"It's a shame, because the ntfs3 driver is miles better than the current ntfs one, and is a solid replacement for the unmaintained ntfs-3g FUSE implementation."

Leonidas P. Papadakos also said in the same vein, "I have to stress that this ntfs driver (fs/ntfs3, which would probably replace fs/ntfs, right?) is an important feature, from a user perspective. It would mean having good support for a cross-platform filesystem suitable for hard drives." He added, "Paragon has been very good about supporting this driver with 26 patchsets, and in my mind it would be suitable for staging. I've seen the discussion slow down since May, and I've been excited to see this merged. This driver is already in a much better feature state than the old ntfs driver from 2001."

At this point Linus Torvalds replied to Leonidas, saying:

"If the new ntfs code has acks from people – and it sounds like it did get them – and Paragon is expected to be the maintainer of it, then I think Paragon should just make a git pull request for it.

"That's assuming that it continues to be all in just fs/ntfs3/ (plus fs/Kconfig, fs/Makefile and MAINTAINERS entries and whatever documentation) and there are no other system-wide changes. Which I don't think it had.

"We simply don't have anybody to funnel new filesystems – the fsdevel mailing list is good for comments and get feedback, but at some point somebody just needs to actually submit it, and that's not what fsdevel ends up doing.

"The argument that "it's already in a much better state than the old ntfs driver" may not be a very strong technical argument (not because of any Paragon problems – just because the old ntfs driver is not great), but it _is_ a fairly strong argument for merging the new one from Paragon.

"And I don't think there has been any huge _complaints_ about the code, and I don't think there's been any sign that being outside the kernel helps."

Konstantin Komarov from Paragon replied to Linus confirming that Paragon would be the official maintainer of the NTFS3 code. As a roadmap, he also indicated that NTFS3 planned to take over the `fs/ntfs` directory in the source tree, once it had proven itself better than the existing driver. He said a pull request was imminent.

Theodore Ts'o, however, had his doubts – although he forgot something, which I'll come to in a moment. He replied to Linus's email, saying he couldn't agree that NTFS3 was better than the old driver and adding that Konstantin had not responded to questions about testing and quality assurance for the NTFS code.

He went on to say, "over the weekend, I decided to take efforts into my own hands, and made the relatively simple changes to fstests needed to add support for ntfs and ntfs3 file systems. The results show that the number [of] fstests failures in ntfs3 is 23% *more* than ntfs. This includes a potential deadlock bug, and generic/475 reliably livelocking. Ntfs3 is also currently not container compatible, because it's not properly handling user namespaces."

Theodore continued:

"Historically, the file system community at large have pushed for a fairly high bar before a file system is merged into the kernel, because there was a concern that once a file system got dumped into fs/ if the maintainers weren't going to commit to continuous improvement of their file system – the only leverage we might have is what effectively amounts to "hazing" to make sure that the prospective maintainers would actually be serious about continuing to work on the file system.

*"One argument for why this should be the case is that unlike a dodgy driver that "just" causes the kernel to crash, if data ends up getting corrupted, simply rebooting won't recover the user's data. And once a file system is added to mainline, it's a lot harder to remove it if it turns out to be buggy as all h*ck.*

"It's not clear this has been an effective strategy. And there are other ways we could handle an abandonware file system – we could liberally festoon its Kconfig with warnings and printk "DANGER WILL ROBINSON" messages when someone attempts to use a dodgy file system in mainline. But I think whatever rationale we give for accepting – or holding off – on ntfs3, we should also think about how we should be handling requests from other file systems such as bcachefs, reiserfs4, tux3, etc."

Matthew was dumbfounded by Theodore's test results – not the results for NTFS3, but the results for the existing in-kernel NTFS code. He said, "I don't understand how so many ntfs-classic xfstests pass." He asked, "Are the tests really passing, or just claiming to pass?"

And this is what Theodore had forgotten. He said that, to be honest, "I had forgotten that we had an in-kernel ntfs implementation. Whenever I've ever needed to access ntfs files, I've always used the ntfs-3g FUSE package."

So he had tested the new NTFS3 code against a user package rather than the existing NTFS implementation. To this, Linus replied, "Well, that's the one we are comparing to, so forgetting it is a bit of an oversight."

Linus added that the FUSE implementation "does indeed work reasonably well." But it was miles behind the NTFS3 code in terms of speed, "and that's kind of the point of ntfs3," Linus concluded.

Theodore agreed, but he was still very dubious about taking NTFS3 into the kernel in its current form. He said, "if you run fstress in parallel ntfs3 will lock up the system hard, and it has at least one lockdep deadlock complaint. It's not up to me, but personally, I'd feel better if *someone* at Paragon Software responded to Darrick and my queries about their quality assurance, and/or made commitments that they would at least *try* to fix the problems that about 5 minutes of testing using fstests turned up trivially."

Darrick J. Wong got behind Theodore on this point, saying:

"<cough> Yes, my aim was to gauge their interest in actively QAing the driver's current problems so that it doesn't become one of the shabby Linux filesystem drivers, like <cough> ntfs.

"Note I didn't even ask for a particular percentage of passing tests, because I already know that non-Unix filesystems fail the tests that look for the more Unix-specific behaviors.

"I really only wanted them to tell /us/ what the baseline is. IMHO the silence from them is a lot more telling. Both generic/013 and generic/475 are basic 'try to create files and read and write data to them' exercisers; failing those is a red flag."

Kari Argillander also remarked, "so many [have] asked and Konstantin has not responded recently. Hopefully he will soon. Of course is it little bit worrying that example generic/013 still fails after almost a year has passed and Konstantin said he is working on it. And it seems that [there are] more tests fails than [at the] beginning of review process." But Kari also pointed out that Konstantin had not been absolutely silent on the issue of testing and QA – Kari dug up an August 2020 mailing list quote from Konstantin, where Konstantin had said, "xfstests are being one of our standard test suites among others. Currently we have the 'generic/339' and 'generic/013' test cases failing, working on it now. Other tests either pass or being skipped (due to missing features e.g. reflink)."

Theodore thanked Kari for that mailing list reference and also added:

"Back in August 2020 Konstantin had promised that they would be publishing their own fsck and mkfs tools. Personally, I consider having a strong set of file system utilities to be as important, if not more important, than the kernel code. Perhaps there are licensing issues which is why he hasn't been able to make his code available?

*"One thing which I wonder about is whether there is anyone other than Konstantin which is working on ntfs3? I'm less concerned about specific problems about the *code* – I'll let folks like Christoph, Dave, and Al weigh in on that front.*

"I'm more concerned about the long term sustainability and maintainability of the effort. Programming is a team sport, and this is especially true in the file system. If you look at the successful file systems, there are multiple developers involved, and ideally, those developers work for a variety of different companies. This way, if a particular file system developer gets hit by a bus, laid low with COVD-19, or gets laid off by their company due to changing business strategies, or just decides to accept a higher paying job elsewhere, the file system can continue to be adequately supported upstream.

"If Konstantin really is the only developer working on ntfs3, that may very well explain why generic/013 failures have been unaddressed in over a year. Which is why I tend to be much more concerned about development community and development processes than just the quality and maturity of the code. If you have a good community and development processes, the code quality will follow. If you don't, that tends to be a recipe for eventual failure.

*"There are a large number of people on the cc line, include from folks like Red Hat, SuSE, etc. It would be *great* to hear that they are also working on ntfs3, and it's not just a one engineer show. (Also, given the deadlock problems, lack of container compatibility, etc., are the Linux distros actually planning on shipping ntfs3 to their customers? Are they going to help make ntfs3 suitable for customers with access to their help desks?)"*

Konstantin replied to the whole question of testing and QA, saying:

"The main thing to outline is that: we have the number of autotests executed for ntfs3 code. More specifically, we are using TeamCity as our CI tool, which is handling autotests. Those are being executed against each commit to the ntfs3 codebase.

"Autotests are divided into the "promotion" levels, which are quite standard: L0, L1, L2. Those levels have the division from the shortest "smoke" (L0) to the longest set (L2). This we need to cover the ntfs3 functionality with tests under [a] given amount of time (feedback loop for L0 is minutes, while for L2 is up to 24hrs).

"As for suites we are using – it is the mix of open/well known suites: xfstests, ltp, pjd suite, fsx, dirstress, fstorture – those are of known utilities/suites [--] [a]nd [a] number of internal autotests which were developed for covering various parts of fs specs, regression autotests which are introduced to the infrastructure after bugfixes and autotests written to test the driver operation on various data sets.

"This approach is settled in Paragon for years, and ntfs3, from the first line of code written, is being developed this way."

Darrick replied that this was very helpful and compared Paragon's internal testing with his own testing system, offering technical feedback to Konstantin. The discussion ended at this point, and it still seems unclear whether NTFS3 will go into the kernel or not. Theodore's maintainership issues have a solid history in kernel development, and it's possible Linus will take heed of that – or he may feel that NTFS3 is a clear improvement over NTFS regardless of future work and decide they might as well replace it right now. ■■■

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.**

SeaGL 2021 JOIN US!

Seattle GNU/Linux Conference returns for the ninth year!

We're an annual community-focused free/libre/open source software, hardware, and culture event in Seattle, and since last year, *virtually* all over the world! Join us for free, no registration required!

This year, alongside the keynotes and many presentations, will see the return of the **Career Expo** - providing resume reviews and career guidance, **TeaGL** - our virtual tea time and online tea swap, and many more socialization opportunities!

Interested in helping make SeaGL happen? Lend a wing to our *all volunteer effort*! Connected to a company who wants to sponsor? Contributions will have a *real and lasting impact* on the FLOSS community!

November 5th & 6th, 2021
Virtual Conference

Visit SeaGL.org for more information!

Building Linux from Scratch

Master Recipe

If you really want to learn about Linux, build it from scratch.

By Markus Frei

Linux from Scratch (LFS) [1] and Beyond Linux from Scratch (BLFS) [2] are online tutorials that provide step-by-step instructions for how to compile and assemble your own Linux system from freely available sources. The abbreviation LFS refers to both the documentation and the resulting Linux system.

If you've been around Linux long enough, you have probably come across the need to compile software – either from a homegrown application or possibly from source code available at an open source project website. You might think that compiling an operating system is similar, and in some ways it is, but building a complete operating system is vastly more complicated than compiling your average desktop application.

Keep in mind that the modern distro you are using to build that desktop app already has a complete toolchain that has been tested and approved for compatibility. Imagine that this build environment doesn't exist yet and you have to create it. That includes a compiler and also important libraries such as glibc and libstdc++. The Linux from Scratch instructions (sometimes call *the book*) contain 11 chapters that are divided into five major parts. The authors break the process of building a Linux system into three stages (Figure 1):

- configure a build host (Part II, Chapters 2 to 4)
- build a cross toolchain and temporarily required tools (Part III, Chapters 5 to 7)
- create the actual LFS system (Part IV, Chapters 8 to 11).

Gerard Beekmans, the author of LFS, started working with Linux in 1998. After using various distributions, he realized that he could not find a "one size fits all" solution, and he would need to build his own system. When he started compiling, he quickly discovered that he had unleashed an onslaught of compile-time errors and circular dependencies. The experience he gained, which he shared with the Linux community, met with such widespread interest that the LFS project was born.

LFS is one of several distribution kits that focus on letting the user build their own system. Many of these projects specialize in the embedded area, including BitBake from the Yocto project [3]. Other options include Linux Target Image Builder [4], OpenWrt [5], and Scratchbox [6].

LFS makes an effort to follow common standards, including POSIX.1-2008 [7], which defines an operating system interface, including an environment for portability purposes, the Filesystem Hierarchy Standard FHS 3.0 [8], and the Linux Standard Base LSB 5.0 [9]. The Linux Standard Base defines four standards for core, desktop, runtime languages and imaging, that are hotly debated. LFS can implement them only partially due to its minimal software equipment.

If you follow the instructions, you will end up with a minimal Linux system that includes an Ext4 filesystem, Grub 2.04, Kernel 5.10.17, Bash 5.1, GCC 10.2, Perl 5.32.1, Python 3.9.2, and Vim 8.2. (This article uses LFS v10.1 from March 2021.) You have the choice between LFS with SysVinit [10] and LFS with Systemd and D-Bus [11]. I'll cover the Systemd variant, which includes DHCP and NTP clients, as well as other bits and pieces that are not included in the SysVinit variant of LFS, although BLFS offers some additional SysVinit options.

The build takes a good deal of time: Using the test VM described in this article with KVM on a machine with Core i7 10th Gen and NVMe SSD, the build takes several hours even with the tests turned off.

Cross-Compilating

To minimize the influence of the build host's operating system, you need to make sure that the LFS system is independent of it. Chapters 5 and 6 help you create the software tools you'll need to build Linux from Scratch. You then deploy these tools in an isolated chroot environment only for the purpose of building LFS.

The build process is based on the principle of cross-compiling: A cross-compiler is typically used to compile code for other system architectures. On your build machine, a cross-comiler is theoretically not necessary because you are most likely building on x86_64 for x86_64. However, you will want LFS to be free of inlinked software from the build host, which a cross-compiler guarantees.

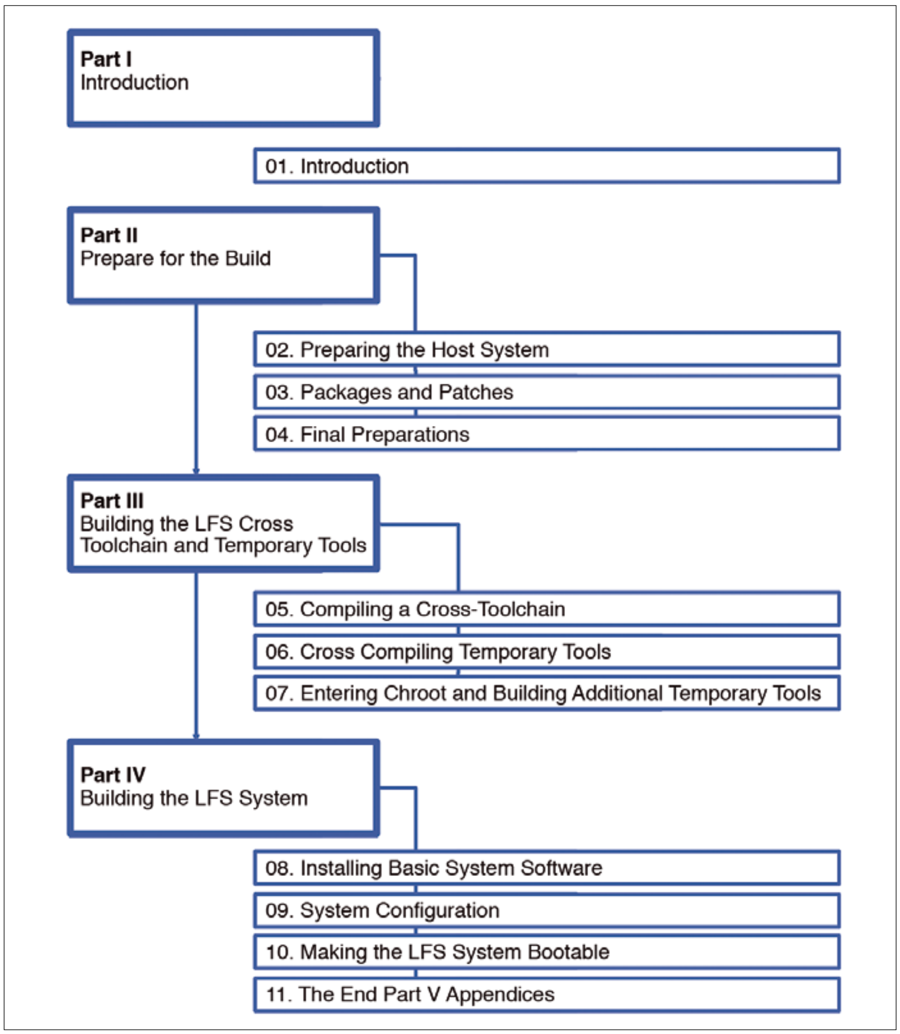
The LFS documentation assumes you have three hosts, each with a different architecture. Host A has a compiler for its own architecture. The host is slow and its capacity is limited. Host B does not have a compiler initially, but it is fast and well equipped. It will be producing the software for Host C. Host C, with yet another architecture, also has no compiler and is small and slow.

This scenario requires two cross compilers: Cross-Compiler A and Cross-Compiler B. On Host A, the compiler builds Cross-Compiler A, which in turn builds Cross-Compiler B. Cross-Compiler B is used on Host B to build Compiler C and all programs for Host C. However, they cannot be tested on Host B. Therefore, Compiler C rebuilds and tests itself on Host C.

For LFS to succeed with cross-compiling on the same host, you need to gently adjust the well-known vendor triplet `x86_64-pc-linux-gnu` (which always contains four components today) in `LFS_TGT` to fool the compiler into thinking it has different architectures. Then, on the build host, the compiler build host first builds the cross-compiler build host, which in turn builds the compiler for LFS. In the LFS chroot on the build host, the compiler LFS then rebuilds and tests itself and is then ready for deployment.

For LFS, you need to use the cross-compiler to compile not only the C compiler but also the glibc and libstdc++ libraries. The problem is that, to be built, the C compiler relies internally on libgcc, which in turn has to be linked against glibc – which doesn't exist yet.

To work around these circular dependencies, first build a minimal version of the cross-compiler build host that is just about good enough to compile a full-fledged glibc, as well as a minimally functional libstdc++. The libstdc++ library will be built again later in the chroot environment, this time completely. Further details are given in the LFS documentation.



Part I
Introduction
01. Introduction
Part II
Prepare for the Build
02. Preparing the Host System
03. Packages and Patches
04. Final Preparations
Part III
Building the LFS Cross Toolchain and Temporary Tools
05. Compiling a Cross-Toolchain
06. Cross Compiling Temporary Tools
07. Entering Chroot and Building Additional Temporary Tools
Part IV
Building the LFS System
08. Installing Basic System Software
09. System Configuration
10. Making the LFS System Bootable
11. The End Part V Appendices

Figure 1: The three parts of the LFS build process.

Hardware

The LFS build process will not work unless your build host is working perfectly. The tutorial in this article assumes an unused virtual machine based on a fully patched, current CentOS 8.3 minimal install with BIOS boot. If you're using a different Linux variant, you might need to modify some of the steps, but the process is similar.

The C compiler GCC benefits from a fast core. The documentation discusses a multi-thread build distributed over several cores, but this configuration is not recommended in practice because of occasional race conditions.

For the build server, two cores, 2GB RAM, an SSD-based hypervisor, and an Intel e1000 NIC are all you need. For mass storage, you need a 10GB disk (VirtIO, `/dev/vda/`) for the system and a 20GB disk (SATA or SCSI, `/dev/sda/`) for LFS.

Avoid VirtIO for the LFS disk and network card because the instructions in the LFS manual do not compile VirtIO support into the kernel. LFS is happy with a virtual SATA or SCSI disk that it identifies as `/dev/sda/` and a classic Intel e1000 network card with 1 Gbps. The disk name is important when it comes to making LFS bootable. The name of the network card (such as `enp1s0`) matters if you want to use Predictable Network Interface Names [12].

Getting Started

After CentOS is installed, you need to set up the packages required for the first build processes. You can retrieve the required `makeinfo` from the *texinfo* package in the PowerTools repository, starting with CentOS 8.3. The packages required for the build are quickly installed (Listing 1).

You then need to check the versions installed by CentOS with the help of a small script: LFS v10.0 requires a GCC version of 6.2 or higher and a GNU Make version of 4.0 or higher, which an earlier Linux distribution such as CentOS 7 will not give you.

If everything is fine so far, split the second disk of the build host into the partitions `/`, `swap` and `/home`, create the Ext4 filesystem, and define the `$LFS` environment variable, which is essential for all subsequent shell calls and resolves the root mount point. The variable must therefore work under all circumstances. You will want to complete the locale settings in CentOS 8 and mount the second disk.

Make the `/dev/sda/` disk bootable via BIOS/MBR and add various partitions. To give Grub's own `core.img` enough space, the first partition for `/` with a size of 10GB starts at 1MB. After that, create a `/swap` of 2GB, leaving the rest of the disk reserved for `/home`.

In this setup, `/boot` does not need its own partition but contains the kernel plus the boot loader configuration as an ordinary directory. You only need a separate partition if you want to support multiboot with other distributions, or if the boot loader can no longer access the root file system. Lack of access to the filesystem can happen as a result of missing drivers, disk encryption, software RAID, or LVM.

Loading Packages and Patches

Chapter 3 of the LFS documentation defines LFS with its later feature set. In addition to the build tools, you need to download the desired sources for a choice of software packages that define the system. The makers of LFS have taken the trouble to make a preselection, resolve their version dependencies, and list them in a text file that you can download and conveniently pass into Wget (Listing 2).

The LFS developers provide a number of patches that fix bugs or adapt the packages to the future target system. In addition, they explain all the options of the `configure` calls. An Md5sum checksum list, which is also provided, helps to check the checksums of the almost 90 packages that are downloaded.

Concluding Work

In LFS chapter 4, the *root* user creates the directory structures required for the build `$LFS/{bin,etc,lib,lib64,sbin,tools,usr,var}`, as well as the *lfs* user as owner of the build directories. Then it is a matter of becoming the *lfs* user and emptying and optimizing the shell environment for this account.

Since the new environment is a non-login shell, it needs a `.bashrc` file to extend and customize the shell environment – for example, to set environment variables such as `$LFS`, `LC_ALL`, or `PATH`. The `$LFS/tools/` directory will contain the cross-compiler later on.

The next step is to restart the build host to make sure there are no problems and that it works. At this point at the latest, it is advisable to create a VM snapshot.

Compiling the Cross Toolchain

In chapter 5 of the LFS manual, you remain logged in as the *lfs* user, for whom the real work now begins: building the cross-compiler and its tools (Listing 3). The cross-compiler and its tools are temporarily stored in `$LFS/tools/`; you can already install the libraries where they will need to be in the future.

The workflows for building the software packages almost always remain the same until the end. You change to the `sources/` directory, where you unpack the source files using `tar xf`. Then change to the new directory created in the process, where you create a `build/` directory, if necessary.

Now the package can be compiled and installed. Then delete the source code directory again. You can do this in a first run for the linker and assembler, the cross-compiler, the Linux API headers, glibc, and libstdc++.

Listing 1: Installing the Required Packages

```
$ dnf -y install yum-utils
$ yum config-manager --set-enabled powertools
 dnf -y install bison byacc bzip2 gcc-c++ patch perl python3 tar texinfo wget
```

Listing 2: Downloading a Software Set

```
$ wget http://lfs.linux-sysadmin.com/lfs/downloads/stable-systemd/wget-list
$ wget --input-file=wget-list --continue --directory-prefix=$LFS/sources
```

Listing 3: Testing the Cross-Compiled GCC

```
$ echo 'int main(){}' > dummy.c
$ $LFS_TGT-gcc dummy.c
### Shows the information from the segment headers
### of the file, if any
$ readelf -l a.out | grep '/ld-linux'
### Should return something like:
### [Requesting program interpreter: /lib64/ld-linux-x86-64.so.2]
```

Building Temporarily Required Tools

Still working as *lfs*, proceed to compile Bash, Grep, and the native LFS-GCC, among others, with the cross toolchain that you created in LFS chapter 6. You can install the applications on the rudimentary LFS filesystem, but they cannot yet be used; you will still need the host system for certain tasks. At least the libraries installed in LFS are already used for linking. However, the applications will only really be used in the chroot environment in the next section.

For temporary tasks, you will need to compile a macro processor, a linker and an assembler, the native GCC compiler, a TUI (Text-based User Interface) library, as well as Bash, in the second run. In addition, a number of tools are created, including Awk, Chmod, Cp, Dd, Diff, File, Grep, Gzip, Make, Patch, Sed, Tar, and Xz.

Additional Temporary Tools

Up to this point, the user *lfs* has been the owner of the LFS file system; in chapter 7 of the LFS documentation, the owner changes to *root*. You then create the `console` and `null` devices on the disk, as the kernel expects these devices at boot time.

Resolving the circular dependencies means that, working as the *root* user for the first time, you can now switch to a chroot environment that almost completely locks out the build host's operating system for further tasks. The exception is the running kernel: For the chroot environment to work, communication with the kernel is configured via the Virtual Kernel File System (VFS).

VFSs are file systems (such as `tmpfs/`) that do not take up any disk space and are located entirely in main memory. Normally, the system mounts devices below `/dev` as virtual file systems, using Udev to create them when detected or on first accesses during the boot process. At the moment, Udev is still missing, so you will need to create the required devices manually and mount them via bind-mount. The pitfall is that, when you restart the LFS build host later, you need to re-run the mount commands for the virtual devices before continuing.

The first time you change to the chroot environment, an unusual Bash warning appears – `/etc/passwd` is missing. Now for the final directory structure, including some correctly set permissions plus some log files. The directory structure follows the Filesystem Hierarchy Standard (FHS) and creates, among other things, the `/boot`, `/home`, `/mnt`, `/opt`,and `/srv` directories, while leaving out things you don't need, such as `/usr/local/games`.

The standard users and groups are still missing. Because there are no standardized specifications, LFS follows Udev and other distributions. You then compile and install software to test other programs (using a temporarily created *tester* account). In addition, you are still missing some building blocks of the toolchain. Start by creating Libstdc++ (second run), a package for internationalization and locale support, a parser generator, and the Perl and Python programming languages. This is followed by tools for info pages and smaller utilities, such as Blkid, Dmesg, and others.

The toolchain you just completed now runs independently of the build host. At the end of the chapter, the LFS documentation deals with striping, backups, and restores, but I will not go into these topics in detail.

Installing Basic Software

The next chapter of the LFS documentation first gives an introduction to the topic of package management. It turns out, however, that the project does not use a package manager; the whole point of LFS is to learn to build things, so a package manager would simply not sit well with the orientation of LFS.

Some hard work follows. In chapter 8, for example, you use `configure`, `make`, and `make install` to add a whole bunch of applications to the disk, as well as some important libraries. In addition to Bash, the tools include text processors such as Awk, Grep, and Sed, as well as packers and archive programs such as Bzip, Gzip, Tar, and Xz. In addition, you will add GCC, the boot loader Grub, Man and Mandb, OpenSSL, Perl and Python, Vim, and Udev.

Compiling the GCC is by far the most time-consuming task in the entire project. This step takes almost half of the total compilation time by running the test suites. You will see masses of *FAILED* messages, but you don't need to worry about them as long as a `grep` on the log files generated by the test suite gives you the expected results. You can also safely ignore warnings about C syntax that appear during the compile process.

The unit tests carry out numerous variations on the `make check` or `make test` command, partly as the *tester* user (via `su tester -c`). If you skip these unit tests, you save a lot of time, but you also risk missing one or two problems in the build process.

After all this work, it's time to clean up. Remove libraries and tools that are no longer needed, as well as the temporary user, *tester*. If the LFS you created will not be used for programming and you do not need debugging functionality, you can save 2GB space by cleaning up the debugging symbols (which the code section takes into account).

Finally, leave the chroot environment and re-enter it immediately afterwards with bash path hashing [13] switched on.

Configuring the System

Chapter 9 is about configuring the network stack. If so desired, you can disable predictable network interface device names, which I will do (Listing 4). This means that LFS detects the first NIC in the system as the classic `eth0`. DHCP gives you an IP address; then you move on to configure name resolution and assign a hostname.

Minor configuration work on the system clock, the Linux console, and the settings for the system locales complete the system. If you are in doubt about the appropriate locales, check the files `/usr/share/keymaps` and `/usr/share/consolefonts`.

Making the System Bootable

It is now time to make LFS bootable. Chapter 10 starts by creating an `/etc/fstab` file that points to the LFS partitions, `sda1` (system) and `sda2` (swap). Now build and install the Linux kernel. Because later extensions, such as in BLFS, always require reconfiguration of the kernel, do not delete the source code after the build. Grub-install puts the boot loader on the disc; you can then configure it with Grub-mkconfig.

Listing 4: Using Classic Interface Names

```
# ln -s /dev/null /etc/systemd/network/99-default.link
```

Using `make defconfig` in the code section, LFS configures the kernel without interaction and with meaningful default values. If necessary, you can change these settings and use the text-based kernel configuration variant via `make menuconfig` to manually set the myriad parameters of the kernel. Important: The makers of Systemd strongly recommend the use of IPv6. You will want to switch the kernel features from Listing 4 on or off to match the network settings.

Conclusion and Reboot

You have now completely installed LFS and can immortalize your name in two files, `/etc/lsb-release` and `/etc/os-release`. Logout from the chroot, set a root password, and unmount `$LFS` – nothing else stands in the way of a reboot. (If you run into any problems, see Table 1 for some troubleshooting tips.) If you selected the LFS disk as the primary boot medium in the build VM, your LFS will be up and running after a short time.

As you wade through the details of the process described in this article, you will quickly see that building Linux from Scratch takes a lot of time, energy, and attention. The team around the LFS project leader, Gerard Beekmans, has several answers to the question of whether this effort is worthwhile.

The LFS project will provide you with hands-on practice with building your own distribution. You'll learn about all the necessary components and how they interact with each other. You'll also learn about the build process itself. If you want to, you can use LFS as a starting point for your own development contributions.

At the end of the process, LFS provides you with an up-to-date, working, and very compact Linux system that includes not much more than the kernel and some tools. Building a distribution yourself helps you stay extremely flexible. LFS gives you the shell of a house that you can customize to suit your own taste, from one-room apartment to luxury villa. LFS can be completely audited if required, and, perhaps even more importantly, you have complete control over all security patches. Last but not least: LFS is simply cool.

To dampen expectations a bit: LFS is not especially convenient for a production environment. An SSH daemon is missing, as are Sudo, Wget, and Parted. LVM is not available to manage the file system,

Listing 5: Important Kernel Features

```
General setup -->
  [ ] Auditing Support [CONFIG_AUDIT]
  [*] Control Group support [CONFIG_CGROUPS]
  [ ] Enable deprecated sysfs features to support old userspace tools [CONFIG_SYSFS_DEPRECATED]
  [*] Configure standard kernel features (expert users) [CONFIG_EXPERT] --->
    [*] open by fhandle syscalls [CONFIG_FHANDLE]
  Processor type and features  --->
    [*] Enable seccomp to safely compute untrusted bytecode [CONFIG_SECCOMP]
  Firmware Drivers  --->
    [*] Export DMI identification via sysfs to userspace [CONFIG_DMIID]
  Networking support  --->
    Networking options  --->
      <*> The IPv6 protocol [CONFIG_IPV6]
  Device Drivers  --->
    Generic Driver Options  --->
      [ ] Support for uevent helper [CONFIG_UEVENT_HELPER]
      [*] Maintain a devtmpfs filesystem to mount at /dev [CONFIG_DEVTMPFS]
  Firmware Loader --->
    [ ] Enable the firmware sysfs fallback mechanism [CONFIG_FW_LOADER_USER_HELPER]
  File systems  --->
    [*] Inotify support for userspace [CONFIG_INOTIFY_USER]
    Pseudo filesystems  --->
      [*] Tmpfs POSIX Access Control Lists [CONFIG_TMPFS_POSIX_ACL]
```

Table 1: Tips for Troubleshooting

Symptom	Cause	Remedy
Build action reports `libtool: warning: remember to run 'libtool --finish /usr/lib' (File-5.39)`	Caused by prefix parameters.	Don't run anything and ignore the message.
`The system has no more ptys. Ask your system administrator to create more.`	You may have rebooted the build host after step 7.3 and forgotten to remount the device nodes before the Chroot.	Complete code sections 7.3 and 7.4 again.
LFS simply freezes after the boot and outputs a `Grub` message.	The LFS disk was set up with BIOS/GPT.	Swap `/boot` out onto a separate partition.
When booting in LFS, the following message occurs in Grub `error: hd1 cannot get C/H/S values.`	This often happens if you create the `/boot/grub/grub.cfg` manually.	If you specify the LFS disk, the second disk in the system, as the first boot medium, the file must have a `set root=(hd0,1)` entry. As the second boot medium this is `set root=(hd1,1)`.
`Kernel Panic - not syncing: VFS: Unable to mount root fs on unknown-block(0,0)`	It looks like there are some drivers missing for the medium you are using.	If you are using VirtIO disks, add the drivers or move to SATA disks.

and the network stack is anything but complete. BLFS helps you implement those features as advanced topics.

Be aware of the fact that you will need to keep your LFS system up-to-date by downloading and compiling security updates on a regular basis. For this, the Linux from Scratch documentation discusses a process for simplifying the package management process and defining a package user [15]. A package manager is not used in the entire LFS project, although it is possible to extend LFS by adding a manager such as Pacman or GNU Stow [14]. Your self-built Linux distribution also lacks features such as an active user community, QA-tested packages including errata, an extensive mirror network, contactable developers, and conveniences such as a wiki, IRC chat, mailing lists, forums, bug trackers, or an FAQ.

With LFS, Linux knowledge is not taught but assumed, so you'll need some background in Linux to get started. But if you're ready to roll up your sleeves and dive in, Linux from Scratch will help you build a deeper understanding of the Linux kernel and open source software. ■■■

Author

Markus Frei has been involved with server applications and the interaction of infrastructure and software development for over 25 years. He is the co-owner of Linuxfabrik in Zurich (*https://www.linuxfabrik.ch/*), which offers service and support for Linux and open source applications, as well as managed hosting.

Info

[1] Linux From Scratch: *http://www.linuxfromscratch.org/lfs*

[2] Beyond Linux From Scratch: *http://www.linuxfromscratch.org/blfs*

[3] Yocto Project: *https://www.yoctoproject.org*

[4] Linux Target Image Builder: *http://ltib.org*

[5] OpenWrt: *https://openwrt.org*

[6] Scratchbox: *http://www.scratchbox.org*

[7] Posix: *http://pubs.opengroup.org/onlinepubs/9699919799/*

[8] File System Hierarchy Standard: *http://refspecs.linuxfoundation.org/FHS_3.0/fhs/index.html*

[9] Linux Standard Base: *https://refspecs.linuxfoundation.org/lsb.shtml*

[10] LFS with SysVinit: *http://www.linuxfromscratch.org/lfs/view/stable*

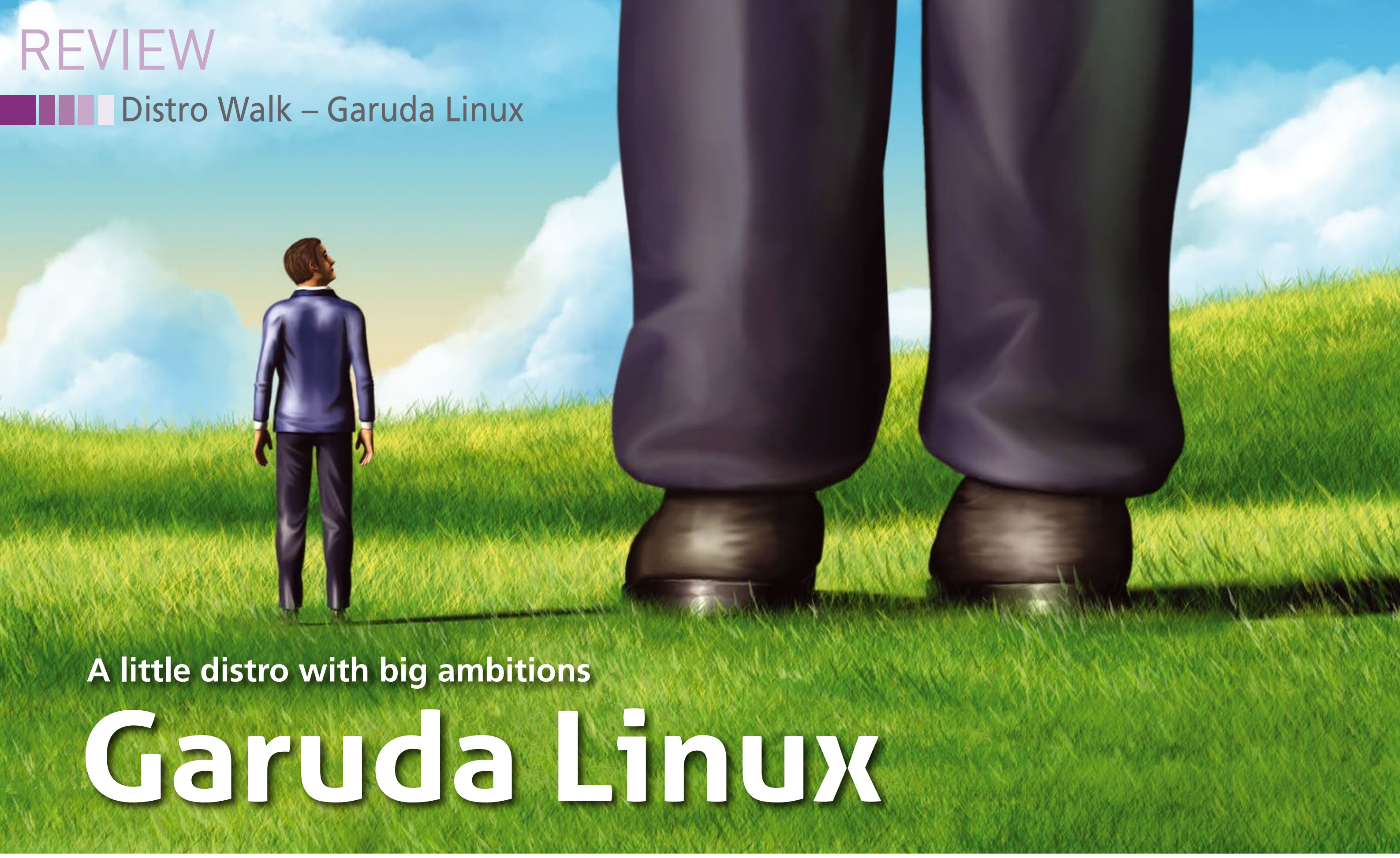
[11] LFS with Systemd: *http://www.linuxfromscratch.org/lfs/view/stable-systemd*

[12] Predictable Network Interface Names: *https://www.freedesktop.org/wiki/Software/systemd/PredictableNetworkInterfaceNames/*

[13] Bash Path Hashing: *https://www.computerhope.com/unix/bash/hash.htm*

[14] GNU Stow: *https://www.gnu.org/software/stow*

[15] Package-User approach: *http://www.linuxfromscratch.org/hints/downloads/files/more_control_and_pkg_man.txt*



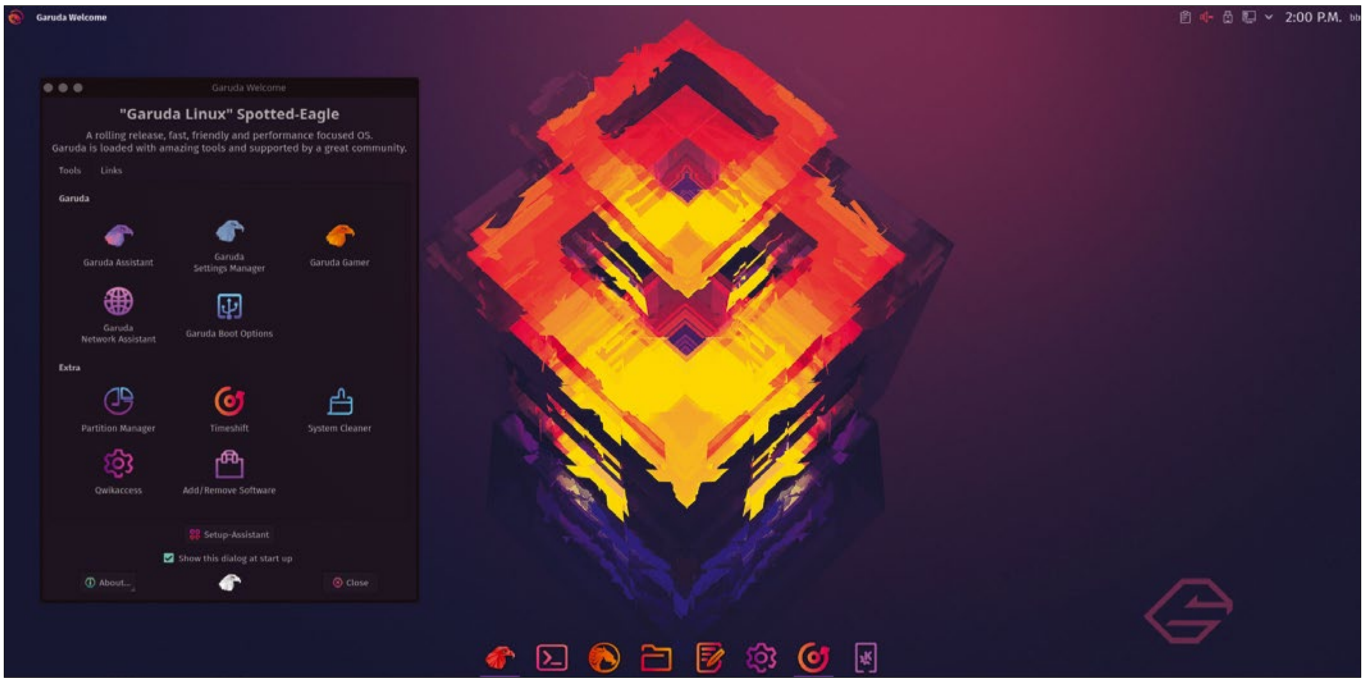
A little distro with big ambitions

Garuda Linux

This friendly Arch Linux distro focuses on usability and modern hardware, making it particularly appealing to keen gamers. *By Bruce Byfield*

Traditionally, Garuda is a giant bird or bird-like being who appears in several Asian mythologies and is a cultural and national symbol in several countries. By contrast, Garuda Linux [1], from India, is a relatively new Linux distribution with an emphasis on efficiency, ease of use, aesthetics, and gaming, as well as a growing name for originality among distributions. Recently, Shrinivas Kumbhar, the lead founder, agreed to talk about the distribution and where it is going.

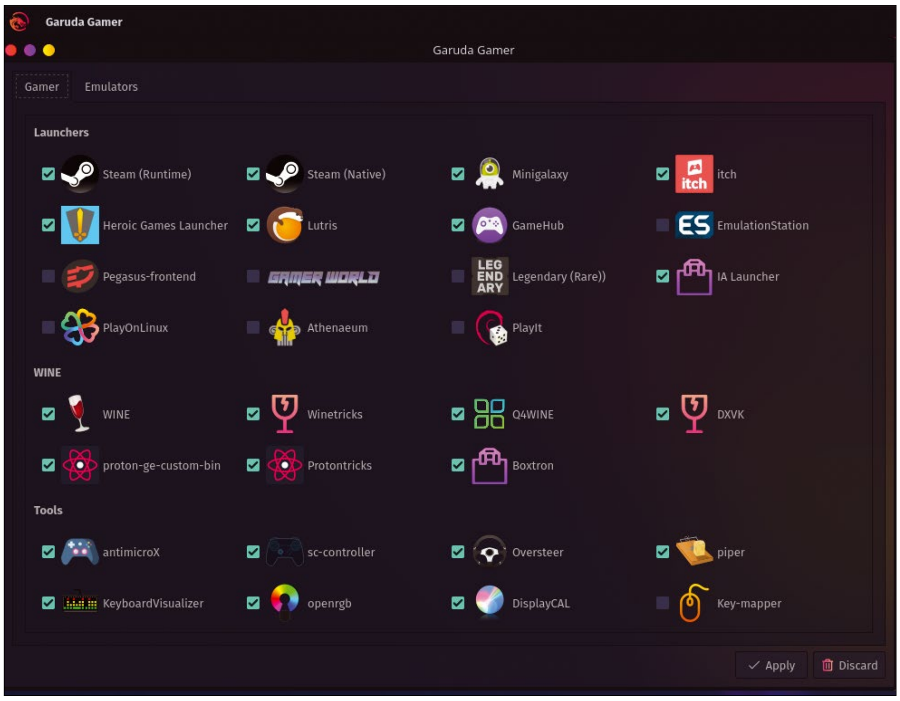
Kumbhar was introduced to free software in junior college when he attended a seminar on ethical hacking. A few years later, he experimented briefly with Remix OS and tried to install Kali Linux. However, when he managed to install Kali, he says, "I couldn't figure out anything. So my hunt for a friendly beginning distro began and I installed Linux Mint. But due to its outdated look and outdated software, I got rid of it. I



"Garuda Linux" Spotted-Eagle

A rolling release, fast, friendly and performance focused OS.
Garuda is loaded with amazing tools and supported by a great community.

Figure 1: The look of Garuda's Dr460nized spin has been described as "cyberpunky."


Garuda Gamer

Gamer | Emulators

Launchers

- ✓ Steam (Runtime)
- ✓ Steam (Native)
- ✓ Minigalaxy
- ✓ itch
- ✓ Heroic Games Launcher
- ✓ Lutris
- ✓ GameHub
- ☐ EmulationStation
- ☐ Pegasus-frontend
- ☐ GAMER WORLD
- ☐ Legendary (Rare))
- ✓ IA Launcher
- ☐ PlayOnLinux
- ☐ Athenaeum
- ☐ PlayIt

WINE

- ✓ WINE
- ✓ Winetricks
- ✓ Q4WINE
- ✓ DXVK
- ✓ proton-ge-custom-bin
- ✓ Protontricks
- ✓ Boxtron

Tools

- ✓ antimicroX
- ✓ sc-controller
- ✓ Oversteer
- ✓ piper
- ✓ KeyboardVisualizer
- ✓ openrgb
- ✓ DisplayCAL
- ☐ Key-mapper

Figure 2: Garuda has extensive support for gaming, although it is equally suited for use as a workstation.

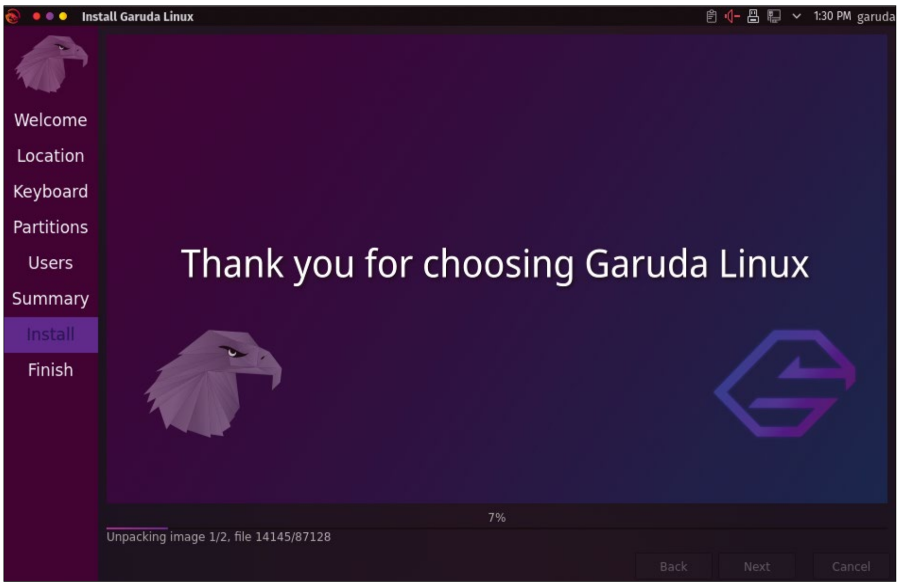
installed Ubuntu and the same thing happened, and so my distro hopping journey began." Interested in gaming, he tried SparkyLinux but was disappointed in its reliance on Debian and Openbox. In the end, Kumbhar settled on Manjaro and discovered its Arch User Repository, as well as KDE Plasma. His first effort at a distro was a Manjaro spin he named manjarowish. When he decided to use Arch Linux's repositories directly, he also renamed the project Garuda, "and ported some awesome tools from Manjaro and various other distros like MX Linux."

Today, Garuda is a loose organization of enthusiasts who share a common core of code while developing their own spins and own aesthetics. For example, the Dr460nized spin features a blurred, dark desktop wallpaper very different from the staid default look in most distributions (Figure 1). Currently, nine spins are listed on the website, varying from traditional desktops such as Gnome, Xfce, and KDE Plasma to lightweight interfaces such as IceWM and tiled desktops such as bspwm and Qtile. The spins also run the range from Garuda Sway, a Wayland version designed for beginners, to Garuda Linux Barebones, which is designed for advanced users. Kumbhar explains, "While every maintainer is free to implement ideas as he wishes, our shared code is discussed within the team (and sometimes in the community as well to gauge interest) and implemented if an overall good solution has been found. The mixture of younger people with partly crazy ideas combined with an overall very experienced selection of long-time Linux users is what shapes Garuda's appearance and codebase. That being said, we are still in the process of getting everyone involved with the team. We trust each other with decisions, but there is always something new to learn, which is really great."

For the most part, Garuda's target audience is specific: people with some Linux experience who want an easy introduction to Arch Linux. In addition, Kumbhar adds, "the system is also tailored to people who want to game a lot. We have a wide range of emulators and games already available in our repo, which makes it really easy to get started quickly (Figure 2). However, Garuda is also perfectly fine for use as a workstation. Also, people who want to switch from Windows might find Garuda a good choice if they are interested in learning how the system works. Most of the basic system maintenance tasks are present in our GUI applications, which takes away some of the fears of having to do everything yourself." According to Kumbhar, the distro's site averages 100,000 visitors per month and has had 40,000 downloads as of July 2021.

Design Philosophy

Asked about Garuda's design philosophy, Kumbhar replies, "we want a beautiful and fast system which focuses on performance and responsiveness. Garuda is mainly focused on modern hardware, the reason being is that we want to break the conception that Linux is only used to repurpose old hardware. That is not the approach we take. We want bleeding-edge features, so we make changes constantly and try many things. Due to our rolling releases, we make changes very often, and months old is very ancient in our thinking."


Install Garuda Linux

Thank you for choosing Garuda Linux

7%

Unpacking image 1/2, file 14145/87128

Figure 3: Garuda features the popular Calamares installer.

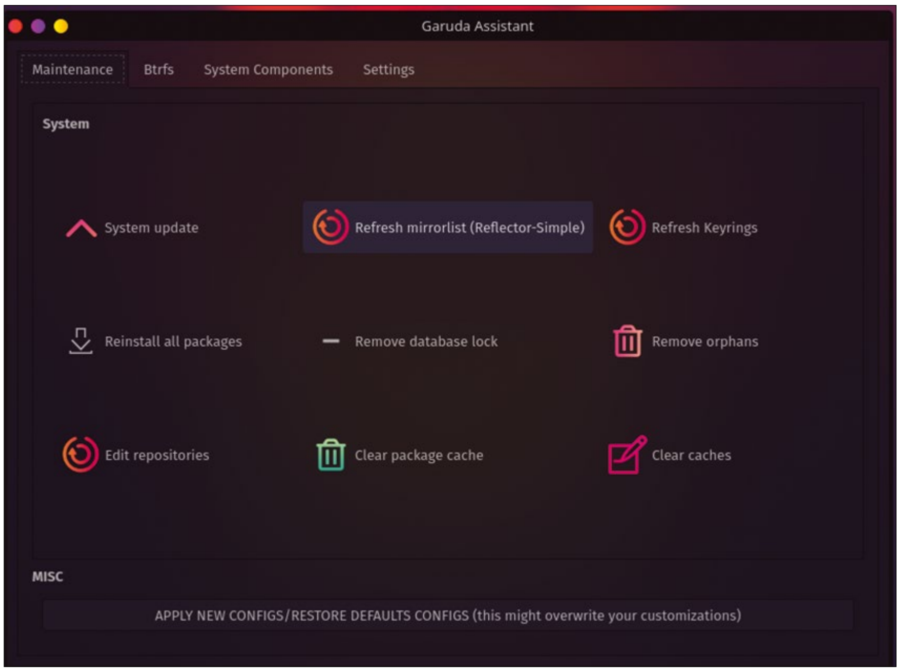


Figure 4: Garuda features an unusually thorough collection of administrative tools.

As an example of how Garuda has evolved, Kumbhar notes that the original plan was to have a system with everything included. However, feedback encouraged the creation of lite and ultimate editions. After another round of feedback, Garuda removed the ultimate editions and created a Setup Assistant tool instead. "Our current goal," Kumbhar adds, "is to reduce the system maintenance that our user has to do in Garuda due to it being a rolling release."

Notable Features

Garuda's selection of features illustrates the distribution's priorities. Like many Arch derivatives like Manjaro, Garuda uses the independently developed Calamares installer [2] to take the pain out of installation (Figure 3). In addition, though, Garuda has an initial setup tool that shows a list of available applications in a dozen different categories – something I have never seen in any other distro. This initial setup tool is accompanied by an unusually thorough set of system management tools, including ones for boot options, network management assistance, and GRUB boot options, which taken together make system management from the desktop a practical possibility (Figure 4).

Less obvious to the casual eye are Garuda's technical tweaks. For example, it installs by default Arch's *linux-zen* [3], a custom kernel that enhances performance with a custom I/O scheduler and a general focus on low-latency performance. The distro also uses the zram [4] kernel module, which boosts performance by creating a compressed swap space in RAM, and `nohang` [5], a daemon that kills lagging or unresponsive applications when memory is low. Even more importantly, Garuda uses the Btrfs filesystem, taking full advantage of its ability to take system snapshots by displaying them at the GRUB boot menu via an application dubbed Timeshift. Timeshift would be a desirable feature in any distribution, but as Kumbhar explains, Timeshift "is integral in making Garuda safe to use with its rolling release model. As Btrfs is a CoW (Copy-on-Write) filesystem, we get instant system restore points which are created automatically before every system update. If there is any issue which can't be resolved, the system can then be easily and instantly rolled back to the state before the update occurred. This feature can often save people in case they play around with the system and perform some system-breaking modifications, making Garuda a perfect fit for people who want to tinker with things in order to learn the ins and outs of Linux."

Looking Ahead

According to Kumbhar, Garuda's main emphasis "is to make the whole distribution very user-friendly. That's why we will continue to implement ideas as we notice what people have issues with." Recently, for example, Garuda added to its system maintenance tools the ability to update keyrings in the background while applying hotfixes before an update. In the near future, the distribution also plans to provide the x86_64_v3 microarchitecture [6] to provide support for some classes of Intel/AMD CPU processor support – just as soon as Arch releases its support. "Last but not least," Kumbhar adds, "there are plans to combine our different Garuda applications into one to have a central place for conveniently administering the system."

At first glance, Garuda might seem to be moving in too many directions at once. Yet, for the most part, its efforts seem to work. By combining ease of use with technical innovation, Garuda provides the best of both. With all its ambition, Garuda is a model of what a derivative should be, keeping in sync with developments within Arch while striking off in specialist directions of its own. ■■■

Info

[1] Garuda Linux: *https://garudalinux.org/*

[2] Calamares installer: *https://calamares.io/*

[3] linux-zen: *https://security.archlinux.org/package/linux-zen*

[4] zram: *https://en.wikipedia.org/wiki/Zram*

[5] nohang: *https://github.com/hakavlad/nohang*

[6] x86_64_v3 microarchitecture: *https://www.phoronix.com/scan.php?page=news_item&px=GCC-11-x86-64-Feature-Levels*

DrupalCon
EUROPE2021

JOIN US!

Just a few weeks to go before
DRUPALCON EUROPE 2021
4 - 7 OCTOBER

Visit our website https://events.drupal.org/europe2021 for more information

Don't forget to register before it's too late!

Take a look at the Program

I want to host my local DrupalCamp during DrupalCon Europe 2021

I want to become a Sponsor

I have a question
drupal@kuoni-congress.com

Where the art of possible comes to life. Network learn and be inspired.

A new slim-line laptop

The Next Generation

We test a new entrant in the world of slim and lightweight Linux-powered laptops. *By Mike Saunders*

Back in November 2020 [1], *Linux Magazine* tested the InfinityBook S 14 v5 from Tuxedo, a German laptop vendor that has been in the Linux business for more than 15 years. Despite a couple of minor gripes with the trackpad and power management, we were largely happy with the machine: It was light, slim, and had excellent battery life (14 hours on medium brightness, playing videos).

Now Tuxedo has a new laptop on offer, the InfinityBook Pro 14 Gen6 [2]. Tuxedo advertises the InfinityBook Pro as an "ultra portable business notebook," with particular marketing efforts focused on its display, magnesium-alloy chassis, and glass touchpad. The laptop is available in various configurations starting at EUR1160 (~$1,360), although the base price does not include a WiFi card. Our review unit checked in at EUR1349 (~$1,582), featuring the screen, chip, RAM, and storage specifications shown in Table 1.

Inside the box, you'll find the laptop along with a USB-C to RJ45 (Ethernet) adaptor, a WebFAI USB stick to restore the operating system (see "The Software Side" box), the power adaptor, and a tiny packet of Intel stickers (thankfully not glued onto the laptop by default). There's also a 40-page user guide and even a Tuxedo-branded paper notepad and pen as well.

Tuxedo laptops come standard with an EU two-year warranty, which can be extended at extra cost to three years for EUR149 (~$175), four years for EUR249 (~$292), and five years for EUR349 (~$409).

Design, Dimensions, and Ports

We were impressed by the InfinityBook S 14's weight and size, but the InfinityBook Pro goes even further: It's slightly reduced in every dimension (1.5cm deep at its thinnest point) and weighs just a smidgen over 1kg (2.3 lb). This reduction in weight comes with a compromise, though, with the keyboard and screen exhibiting slightly more flex when pressed hard compared to the earlier model. We didn't see this as a big problem, though; the overall build quality is solid and way better than the cheap plastic laptops you find at PC stores.

One interesting customization option is the image on the back of the display. By default, the image is a low-profile dark Tuxedo logo, but you can customize it to your liking for an extra EUR59 (~$69) when placing the order (Figure 1).

On the left-hand side of the laptop, you will find a Kensington lock, USB-C port, USB-A port, SD card reader, and combined headphone/microphone jack, and the right-hand side houses the power jack, HDMI, another USB-A port, and a Thunderbolt 3 port. We're happy to see a decent selection of ports on a slim-line model, obviating the need for a bag of dongles in most use cases.

Seven cross-head screws keep the underside cover in place – removing these screws provides access to the replaceable RAM, storage (SSD), and WiFi card (Figure 2).

Screen, Keyboard, and Trackpad

You have two display choices available when configuring the machine: a 2K IPS non-glare 60Hz screen at 1920x1200p with 350 cd/m2 brightness and 1500:1 contrast, and a 3K Omnia Display non-glare 90Hz at 2880x1800p with 400 cd/m2 (and the same contrast) for an extra EUR100 (~$117). We went with the latter for our review unit, and it's a superb screen with excellent viewing angles

Table 1: Specs at a Glance

CPU	Intel Core i5-1135G7 (quad-core @ 2.4 GHz)
Graphics	Intel Iris Xe Graphics G7 (80EUs)
RAM	16GB 3200MHz CL22 Samsung
Screen	14" Omnia Display (3K, 16:10, 90Hz)
Webcam	1.0 megapixel
Storage	500GB Samsung 980 SSD (NVMe PCIe 3.0)
Networking	Intel WiFi 6 AX200 and Bluetooth 5.1
Ports	2x USB-A 3.2 Gen1, 1x USB-C 3.2 Gen2, 1x Thunderbolt 4 (PCIe Gen3 4 Lanes)/USB-C (DisplayPort 1.4a), 1x HDMI 2.0, 1x SD card reader, 1x combined headphone + microphone jack, 1x Kensington lock
Size	31x21.2x1.5-1.9 cm
Weight	1.05kg (2.31 lbs)
Price	EUR1349 (~$1,582)

and only a very thin bezel wrapped around it (Figure 3). Side-by-side with a MacBook Air from late 2020, the InfinityBook Pro is slightly dimmer at the max setting but more than good enough for working outside.

Above the screen is a basic webcam that the `lsusb` command identifies as a Chicony Electronics Ltd HD. We found the webcam to be quite poor with lots of graininess even under adequate lighting conditions. While acceptable for short video chats, you would not want to use the webcam for giving or recording presentations.

The InfinityBook Pro's chiclet keyboard (Figure 4) is responsive and quiet, with two levels of backlighting, decent travel, and a dedicated Tux key rather than a sticker over the usual Windows key. Our only grumble is the arrangement of the cursor keys: The keys are in a full block with the left and right keys at the full height of the combined up and down keys. Some people prefer this, but we miss the ability to immediately recognize the left/right keys with our fingertips when placing them in that area of the keyboard – without having to actually look.

A very spacious glass trackpad sits beneath the keyboard, reaching from the left Alt key to the right Ctrl key. Due to the trackpad's "diving board" style, it's much easier to click at the bottom than the top. The click sound is slightly louder than we'd like but not enough to be annoying when there's other background noise. One interesting addition is a white dot in the top-left corner (Figure 4): Double-tap this dot to disable the trackpad while typing. This is a nice touch, especially for coders, although we found that the trackpad's palm detection was good enough to ignore unintended inputs when tapping away at the keys.

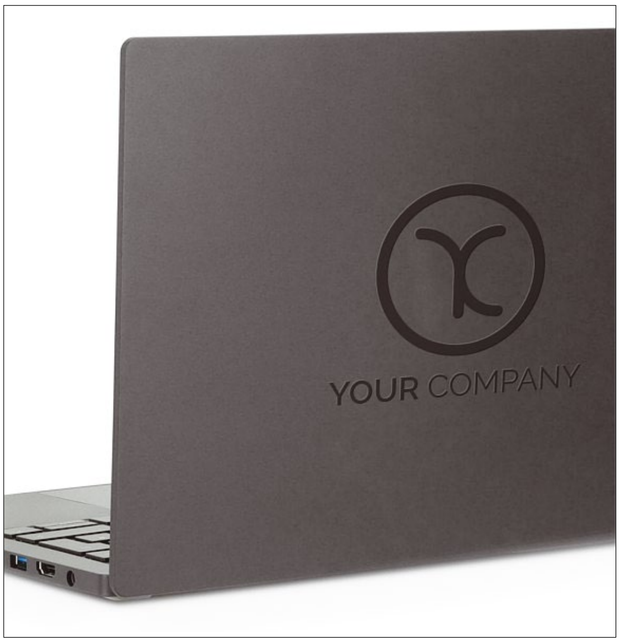


Figure 1: For an extra fee, you can place your own logo on the back of the display.

The Software Side

Tuxedo offers a few operating system options, available when ordering and configuring the machine. The default choice is TUXEDO_OS (which is essentially Ubuntu with the Budgie desktop and some other customizations), but Ubuntu and Kubuntu are also options. You can also order the laptop with no Linux distribution preinstalled. After setting up the distro of your choice, you then need to download and run `tuxedo.sh` to get some laptop-specific extras and settings.

Another option during the ordering process is complete drive encryption, as well as the possibility to have Windows preinstalled inside a VirtualBox virtual machine (VM) for EUR99 (~$116, Home Edition) or EUR159 (~$186, Pro Edition). In addition, you can disable various features (or hinderances, as some might see them) at the BIOS level, such as the Intel Management Engine and webcam.

Our review unit shipped with Kubuntu 20.04, which largely ran very smoothly without the annoying power management glitches we experienced when testing the InfinityBook S (such as a flashing screen on suspend and resume). The keyboard backlight setting still isn't stored between closing and opening the lid, though – a minor annoyance.

Performance and Battery

Our review unit shipped with a quad-core Intel i5 chip running at 2.4GHz and 16GB of RAM, but the machine can be configured with an i7 (1165G7) processor and up to 64GB of RAM – the latter adds EUR305 (~$357) to the base price. In testing, we found the i5 to be plenty capable of daily tasks, ramping up to 4.2GHz where necessary.

The InfinityBook Pro's fan is silent when the laptop is idling, but as soon as CPU activity ramps up it kicks in quietly. It's barely noticeable when watching 60fps 1080p YouTube videos, but as we kept adding more CPU-

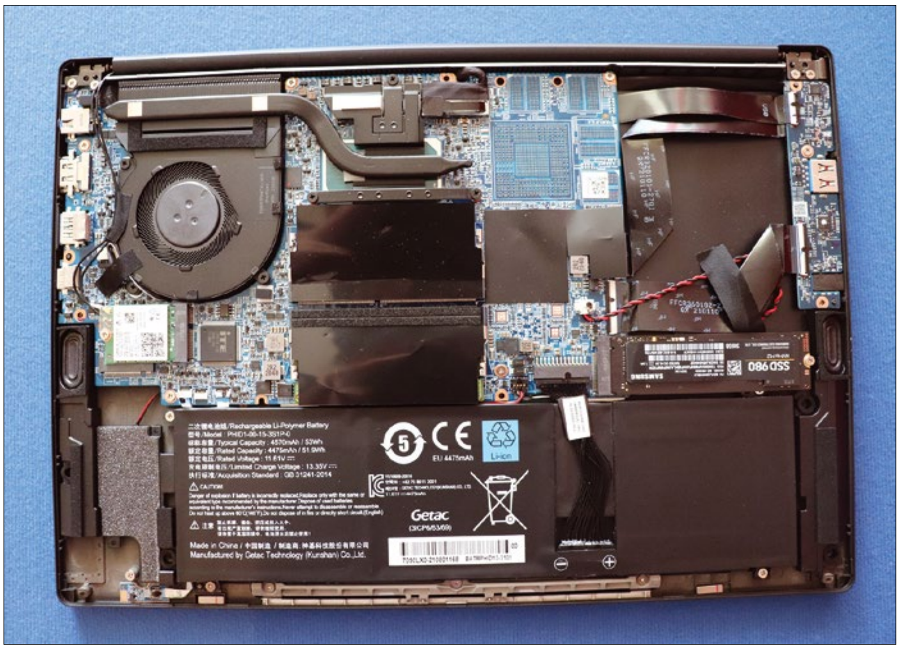


Figure 2: Under the hood: Pop the back off the laptop to upgrade the RAM, SSD storage, and WiFi card.

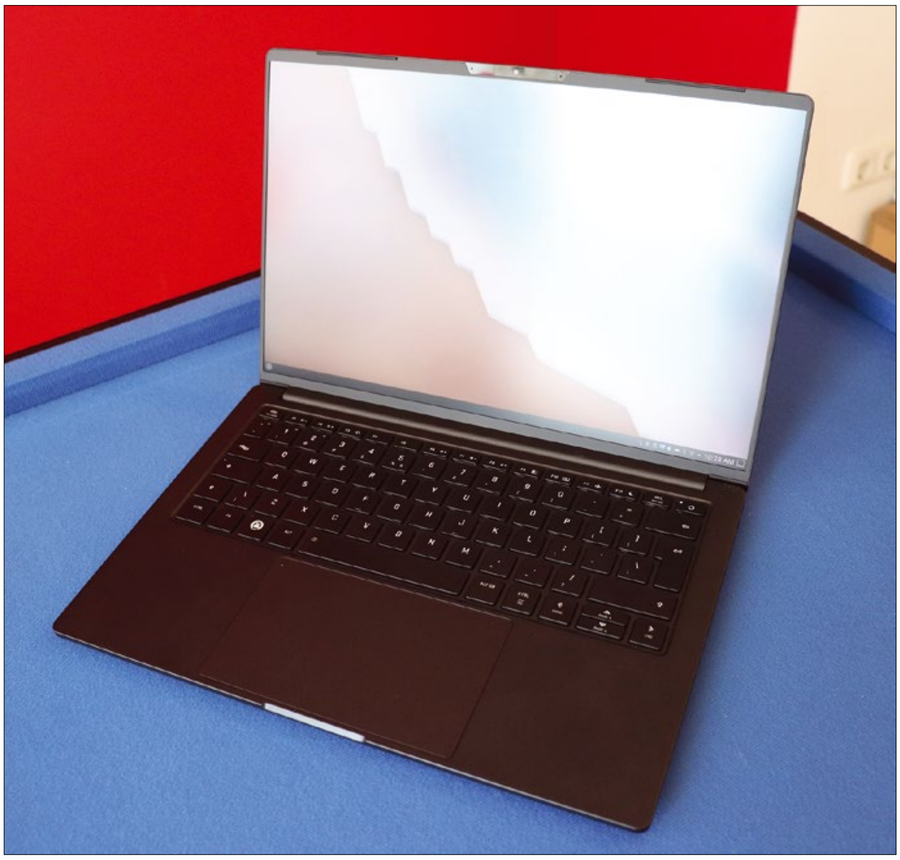


Figure 3: Hate large bezels on laptops? The InfinityBook Pro's screen reaches almost to the edge of the case.

bound tasks it turned into a louder, higher-pitched noise. In most use cases, the fan is not especially distracting, but having spent time with various other slim laptops – such as the fanless M1 MacBook Air – we'd like to see more of a push towards reducing fan noise. However, this is limited by the available CPUs, of course.

Although the laptop has vents under the hinge on both sides, only the left one is used to expel hot air from the fan. The air intake area is the grill on the underside of the laptop, so there's a potential risk of overheating if you use it on a bed or other non-flat surface. Having the air intake and output areas located entirely inside the hinge area would be a better solution.

In terms of battery life, Tuxedo claims that this model's 53Wh Li-Ion battery can achieve "max 12 hours when idle" and "max 8 hours for office use." In testing, we found this to be somewhat optimistic: We left the laptop idling with medium screen brightness and the battery lasted seven hours. In a real-world setting with more demanding tasks (watching videos, installing software, doing some research on the web, and writing this very review), we got 4 hours and 50 minutes out of one charge.

To get the battery back up to 100 percent, the supplied charger needs just under two hours (with the laptop turned off). Because the charging light is built in to the power button on the keyboard, you can't see when the battery has been charged with the lid closed-- you need to open the lid.

Conclusion

On the plus side, the InfinityBook Pro is light, slim, upgradable (RAM and storage), well built, and has a superb display and large, responsive trackpad. Its Linux support is very good, and battery life is okay, if not stellar. Negatives are the bad webcam, fan noise under high CPU loads, and cursor key layout. However if those things won't affect you or how you work, this is a great all-round workhorse that crams an impressive amount into its 1kg body. ■■■

Info

[1] "Tuxedo InfinityBook S 14 v5" by Mike Saunders, *Linux Magazine*, issue 240, November 2020, pp. 20-21

[2] Tuxedo InfinityBook Pro 14 Gen6: *https://www.tuxedocomputers.com/en/Linux-Hardware/Linux-Notebooks/10-14-inch/TUXEDO-InfinityBook-Pro-14-Gen6.tuxedo*

Author

Mike Saunders has been using, advocating, and writing about Linux since 1999. He was one of the founders of *Linux Voice* and gets nostalgic thinking about it.

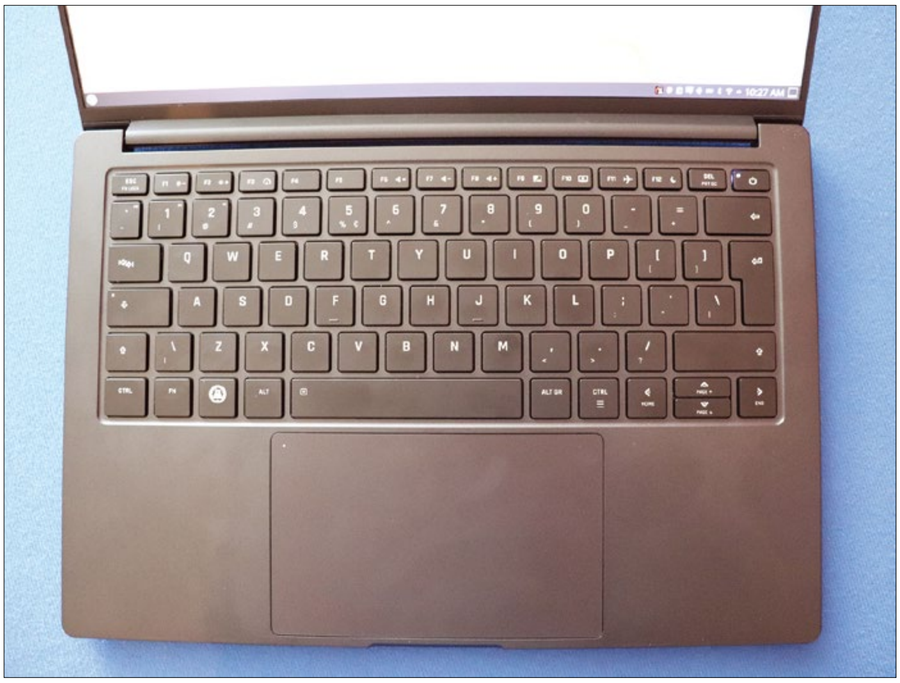


Figure 4: You can configure the machine with many different keyboard layouts. The white dot on the trackpad (top-left) lets you disable the trackpad when typing.

Tools for generating regular expressions

Pattern Seekers

As regular expressions grow in complexity, regex generators can make the job easier by computing the patterns for you. *By Frank Hofmann*

A regular expression (regex) [1] is a sequence of characters that describes a search pattern. You can use a regex to save time searching and replacing in texts, such as strings in programming languages, database query results, or normal documents. Regexes also can help you effectively use utilities, such as `grep` [2], `xmlgrep` [3], and `ugrep` [4].

Ultimately, a regex's usefulness depends on the pattern you select. Formulating a regex that delivers precise results is no easy task (see the "DIY Regular Expressions" box). To save time, a regex generator can automate this step for you by taking a text/character string and deriving a suitable regular expression.

Regex generators work with varying degrees of precision. Some regex generators offer maximum precision, where the regular expression finds exactly one pattern. Others offer minimum precision, where the regex finds a set of patterns with a similar structure. In this article, I test several regex generators to determine how well they work (see also the "Nonfunctional" box).

Olaf Neumann's Regex Generator

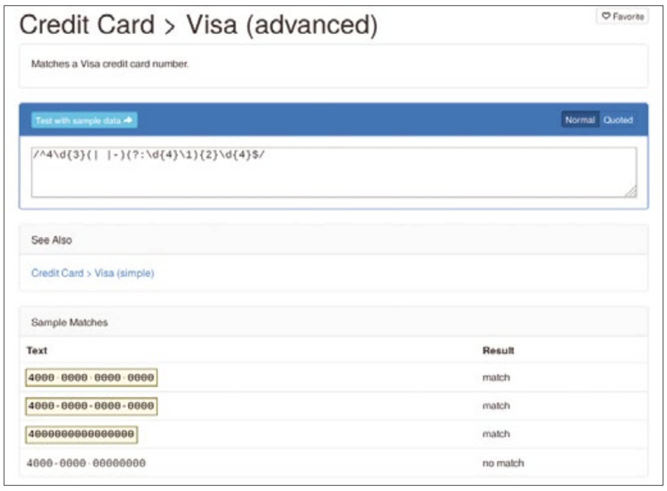
Regex Generator [12], which is implemented in Kotlin and JavaScript, lets you create regular expressions via Olaf Neumann's website. The website offers several input boxes. First, you enter the text

DIY Regular Expressions

While formulating a regular expression can take some time, especially for newcomers, it is worth the effort if you want faster results. Regular-Expressions.info [5], Regex DB [6], and *Mastering Regular Expressions* [7] all clearly explain strategies for creating regexes and provide a huge number of examples to show how useful regexes can be.

For instance, you can use a regular expression to test whether a credit card number entered on a website form is in the correct format. Figure 1 shows a regular expression for numbers on Visa credit cards: the number *4* followed by any three digits, then a separator (nothing, a space, or a hyphen), followed by up to three blocks of any four digits, which can again be followed by a separator. This regular expression does not validate the card, but it reliably filters out incomplete numbers and identifies typing errors.

The RegEx101 wiki [8] also has a good reputation for testing your regular expressions. Not only does RegEx101 support different regex dialects, but it also shows you which patterns match and provides an explanation. Figure 2 shows a RegEx101 test for dates, where the individual components (day, month, and year) are given as decimal values separated by hyphens.


Credit Card > Visa (advanced)

Matches a Visa credit card number.

```
/^4\d{3}(| |-)(?:\d{4}\1){2}\d{4}$/
```

See Also

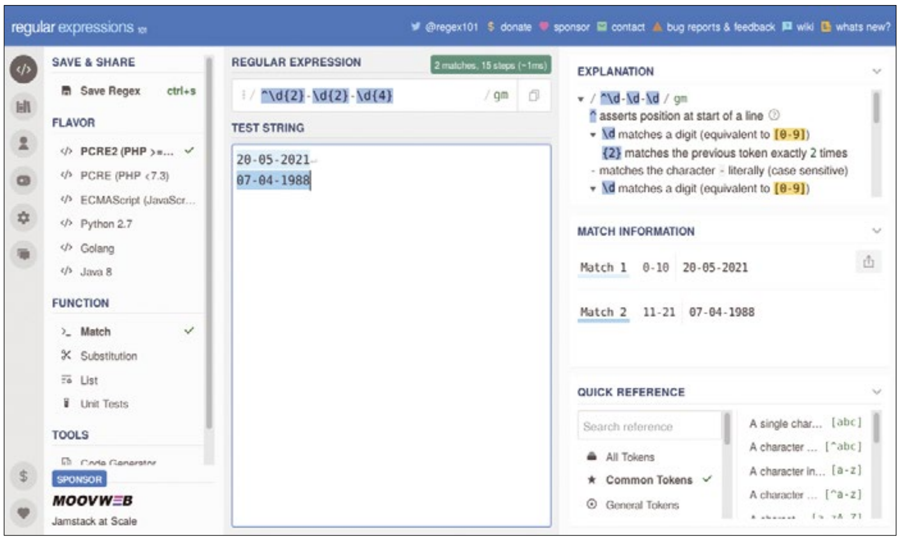
Credit Card > Visa (simple)

Sample Matches

Text	Result
4000 0000 0000 0000	match
4000-0000-0000-0000	match
4000000000000000	match
4000-0000 00000000	no match

Figure 1: Regular expressions offer a relatively easy approach to validating the structure of a Visa card number.

Photo by JJ Ying on Unsplash



REGULAR EXPRESSION 2 matches, 15 steps (~1ms)

```
/ ^\d{2}-\d{2}-\d{4} / gm
```

TEST STRING

```
20-05-2021
07-04-1988
```

EXPLANATION

```
/ ^\d-\d-\d / gm
^ asserts position at start of a line
\d matches a digit (equivalent to [0-9])
{2} matches the previous token exactly 2 times
- matches the character - literally (case sensitive)
\d matches a digit (equivalent to [0-9])
```

MATCH INFORMATION

```
Match 1   0-10    20-05-2021
Match 2   11-21   07-04-1988
```

Figure 2: RegEx101 checking date input.

string for which you are searching. As an example, I used a typical log entry consisting of a date and a message, both separated by a space.

Nonfunctional

During testing, I came across two projects, txt2re [9] and grex [10], which each offered a good approach in theory but ultimately did not generate usable results.

The private website txt2re, developed by Mark James Ennis, works similarly to Regex Generator. Ennis's dislike of regular expressions led him to try to automate the process as much as possible. Unfortunately, txt2re currently only works with the default text string `20:Apr:2016 This is an Example!`, making the tool only useful for demonstration purposes. It remains unclear whether the limited functionality is due to the web browser or the JavaScript used.

Grex v1.2 is based on Mozilla's Rust programming language and the JavaScript-based `regexgen` [11] tool. While grex is currently available for download from its GitHub project page, my attempt to compile grex from the sources on Debian 11 failed.

From your entered text in box 1, the software derives the individual components, which it highlights in different colors (shown in box 2). You then select the relevant component by clicking on the respective color or component. The component is then colored gray. Regex Generator then generates the appropriate regular expression from your selection and displays it in box 3 (Figure 3).

You can then select the programming language where you will use the regular expression. Currently, you can choose between several languages, including PHP, Ruby, and JavaScript (Figure 4).

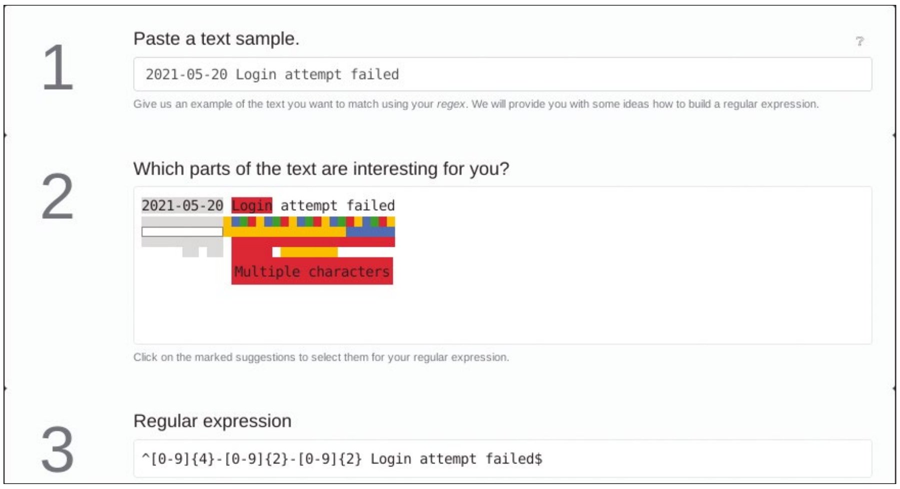
In terms of precision, Regex Generator provides moderate accuracy. The date matches any data, but the text only matches the error message `Login attempt failed` in upper and lowercase thanks to the `/i` parameter at the end of the regular expression. Consequently, Regex Generator is adequate if you just want to search for failed logins for arbitrary dates, for example.

rgxg

ReGular eXpression Generator (`rgxg`) [13] generates regexes from the command line (Figure 5). You can find `rgxg` in the Debian, Ubuntu, and Linux Mint repositories and install it using familiar on-board tools. For better output readability, you should use `rgxg` in a terminal window with a dark background.

In practice, `rgxg` expects a call with different subcommands (see Table 1).

In addition, `rgxg` offers a whole range of parameters to further refine the generated expressions (see Table 2).



Paste a text sample.

```
2021-05-20 Login attempt failed
```

Give us an example of the text you want to match using your *regex*. We will provide you with some ideas how to build a regular expression.

Which parts of the text are interesting for you?

```
2021-05-20 Login attempt failed
```

Click on the marked suggestions to select them for your regular expression.

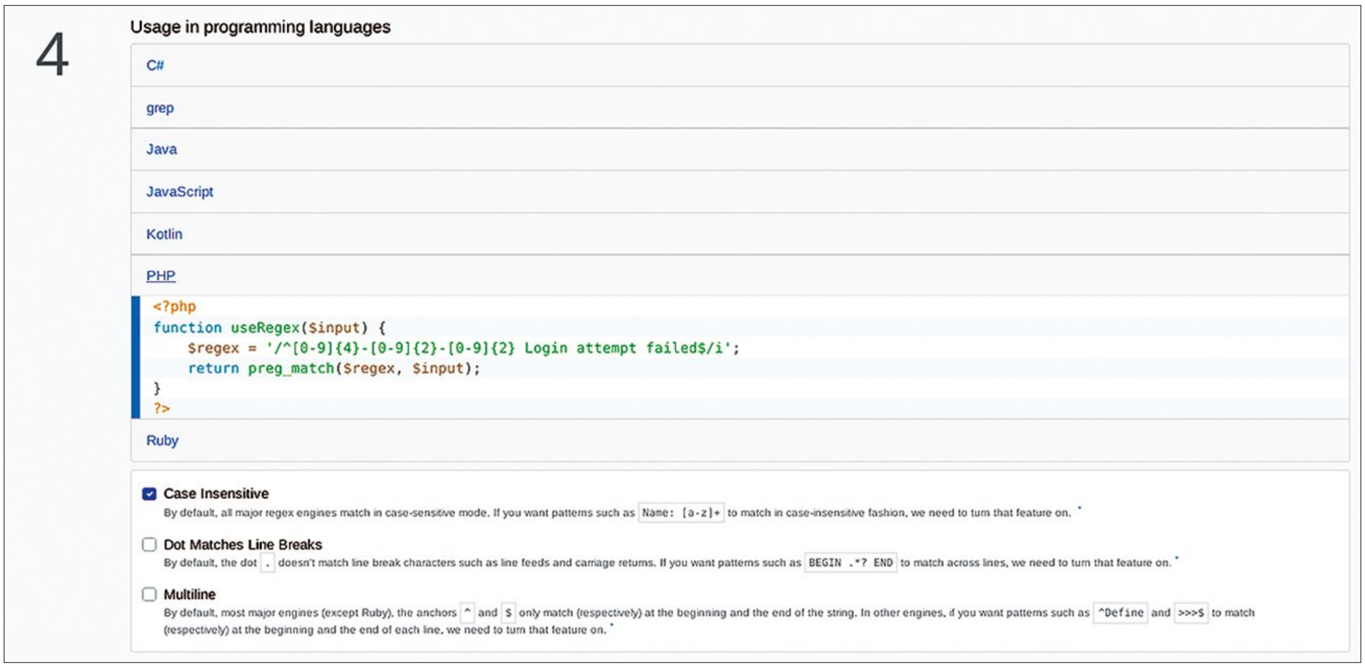
Regular expression

```
^[0-9]{4}-[0-9]{2}-[0-9]{2} Login attempt failed$
```

Figure 3: After entering the desired text string and selecting the components in Neumann's online Regex Generator, the appropriate regular expression is returned.

Table 1: rgxg Subcommands

Subcommand	Description	Example	Generated Expression
`alternation`	Generates an expression to match each of the stated patterns	`rgxg alternation January February`	`(January|February)`
`cidr`	Generates an expression to match all addresses in a CIDR block (IPv4)	`rgxg cidr 192.168.1.0/24`	`192\.168\.1\.(25[0-5]|2[0-4][0-9]|1[0-9]{2}|[1-9]?[0-9])`
`escape`	Escapes a string	`rgxg escape '(1+2)*3'`	`\(1\+2\)\*3`
`range`	Generates an expression to match the stated range	`rgxg range 5 10`	`(10|[5-9])`



4

Usage in programming languages

C#

grep

Java

JavaScript

Kotlin

PHP

```
<?php
function useRegex($input) {
    $regex = '/^[0-9]{4}-[0-9]{2}-[0-9]{2} Login attempt failed$/i';
    return preg_match($regex, $input);
}
?>
```

Ruby

[✓ Case Insensitive]
By default, all major regex engines match in case-sensitive mode. If you want patterns such as `Name: [a-z]+` to match in case-insensitive fashion, we need to turn that feature on.

☐ Dot Matches Line Breaks
By default, the dot `.` doesn't match line break characters such as line feeds and carriage returns. If you want patterns such as `BEGIN .*? END` to match across lines, we need to turn that feature on.

☐ Multiline
By default, most major engines (except Ruby), the anchors `^` and `$` only match (respectively) at the beginning and the end of the string. In other engines, if you want patterns such as `^Define` and `>>>$` to match (respectively) at the beginning and the end of each line, we need to turn that feature on.

Figure 4: After selecting the desired programming language, Regex Generator displays a regular expression (shown here as a PHP function).

Table 2: rgxg Parameters

Parameter	Description	Example	Generated Expression
`range -N`	No outer brackets	`rgxg range -N 5 10`	`10|[5-9]`
`range -z`	Only numbers with leading zeros	`rgxg range -z 0 31`	`(3[01]|[0-2][0-9])`
`range -z -m`	Only figures with a certain number of leading zeros	`rgxg range -z -m 4 0 31`	`(003[01]|00[0-2][0-9])`

In terms of accuracy, `rgxg` offers maximum precision thanks to the parameters that let you generate a regular expression that exactly matches the given pattern.

txt2regex

Another command-line option, txt2regex [14] is a shell script with keyboard-based menu control. Like `rgxg`, you will find txt2regex in the repositories of most recent distributions, such as Debian or Fedora. On macOS, you can use Fink [15] for the install.

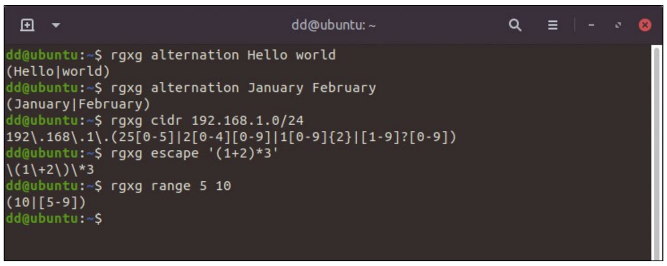
Due to the user interface's color scheme, you will want to use txt2regex in a terminal with a dark background. Alternatively, use the `--nocolor` parameter to switch off the color or `--whitebg` to switch to a suitable display for a light background. When txt2regex is running, typing an asterisk will toggle the display.

Txt2regex offers a helping hand when generating regular expressions by asking questions about the characters, including how many characters follow each other. Based on the input, txt2regex assembles the expression for various tools, such as `awk`, `find`, `grep`, PHP, and PostgreSQL.

You can access a complete list of dialects by typing a forward slash. You then select the corresponding letter from the list (Figure 6) to generate the regular expressions in the desired dialect (Figure 7).

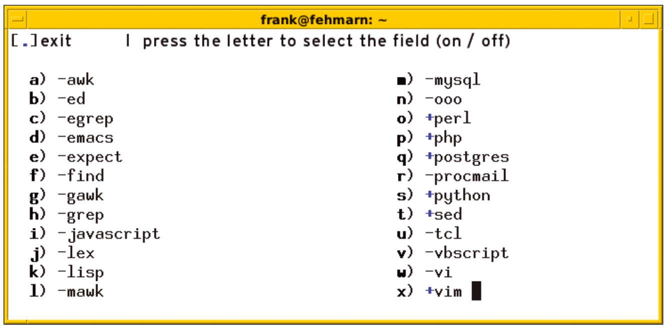
Txt2regex requires precision. As you use txt2regex, you will learn to specify the pattern sequence precisely. Txt2regex provides an overview of how you need to specify the regular expression in the respective tool for it to recognize the pattern. This is a big help because various implementations and dialects are found in everyday Linux: regular expressions, extended regular expressions, and Perl-Compatible Regular Expressions (PCRE).

Like `rgxg`, txt2regex formulates regular expressions with maximum precision.



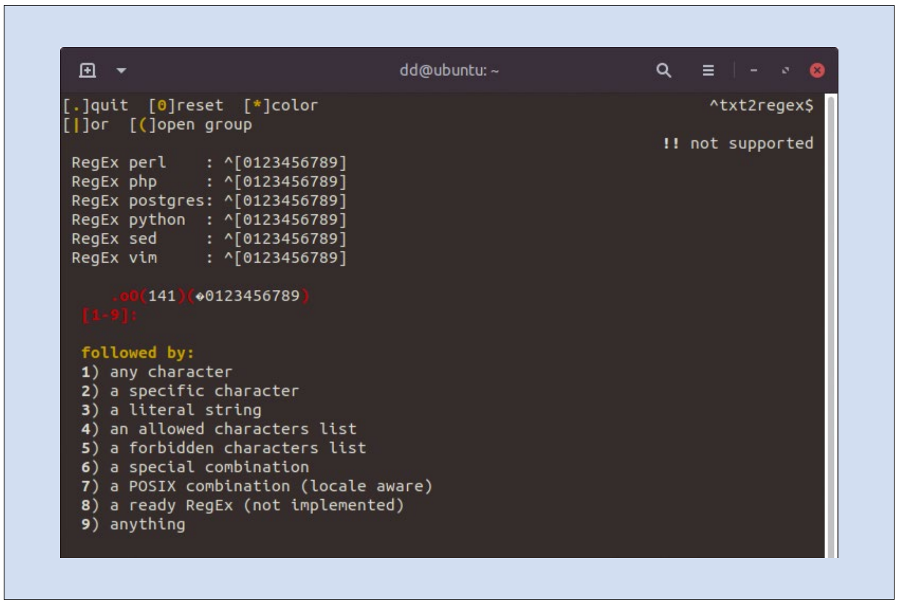
```
dd@ubuntu:~$ rgxg alternation Hello world
(Hello|world)
dd@ubuntu:~$ rgxg alternation January February
(January|February)
dd@ubuntu:~$ rgxg cidr 192.168.1.0/24
192\.168\.1\.(25[0-5]|2[0-4][0-9]|1[0-9]{2}|[1-9]?[0-9])
dd@ubuntu:~$ rgxg escape '(1+2)*3'
\(1\+2\)\*3
dd@ubuntu:~$ rgxg range 5 10
(10|[5-9])
dd@ubuntu:~$
```

Figure 5: Regular expressions generated by the `rgxg` command-line program.



```
[.]exit       | press the letter to select the field (on / off)

  a) -awk                           m) -mysql
  b) -ed                            n) -ooo
  c) -egrep                         o) +perl
  d) -emacs                         p) +php
  e) -expect                        q) +postgres
  f) -find                          r) -procmail
  g) -gawk                          s) +python
  h) -grep                          t) +sed
  i) -javascript                    u) -tcl
  j) -lex                           v) -vbscript
  k) -lisp                          w) -vi
  l) -mawk                          x) +vim
```

Figure 6: Txt2regex supports a large number of dialects for regular expressions.



```
dd@ubuntu: ~
[.]quit  [0]reset  [*]color                              ^txt2regex$
[|]or  [(]open group
                                                      !! not supported
 RegEx perl    : ^[0123456789]
 RegEx php     : ^[0123456789]
 RegEx postgres: ^[0123456789]
 RegEx python  : ^[0123456789]
 RegEx sed     : ^[0123456789]
 RegEx vim     : ^[0123456789]

     .oO(141)(♦0123456789)
  [1-9]:

  followed by:
  1) any character
  2) a specific character
  3) a literal string
  4) an allowed characters list
  5) a forbidden characters list
  6) a special combination
  7) a POSIX combination (locale aware)
  8) a ready RegEx (not implemented)
  9) anything
```

Figure 7: Txt2regex generates patterns for the selected dialects. This is an intermediate step in querying for the desired string.

Listing 1: RegexGenerator Script

```
from RegexGenerator import RegexGenerator

myRegexGenerator = RegexGenerator("415-5553-7676")

print(myRegexGenerator.get_regex())
```

Listing 2: rg.py

```
$ python3 rg.py

\d{3}[-]\d{4}[-]\d{4}
```

sion by using parameters to adapt the expression to your specified pattern.

RegexGenerator

In looking for libraries that work similarly to the other tools discussed in this article, my research turned up limited results. For Python, I found RegexGenerator [16] and its associated *regex-generator-lib* [17].

Listing 1 shows a short script for RegexGenerator, which determines a regular expression for the string `415-5553-7676`. You can then save and run the regular expression as `rg.py` (Listing 2). The generated regex shown in Listing 2 does the trick: a pattern consisting of three digits, followed by a minus sign, four digits, another minus sign, and another four digits.

However, this generated regex lacks precision. Not only does it match the above string, but it also matches other strings, such as `123-4567-8901` or `foo-123-5553-7676-bar`. According to this regular expression, the digits can be arbitrary and the pattern can include other characters because the regex does not use delimiters such as `\b` for word boundary, `^` for beginning of line, and `$` for end of line.

Instead, A regex of `^415\-5553\-7676$` would be more precise and easier to read, resulting in the three digits `415` followed by a minus sign, three times the number `5` followed by a `3`, another minus sign, and then two times the sequence of digits `76` including characters for the beginning of the line (`^`) and the end of the line (`$`).

rex

If you use the `rex` tool from Python's Test-Driven Data Analysis (*tdda*) package [18], there is a very neat, practical use case. The example shown in Listing 3 determines the regular expression for naming image files. All of the file names start with the three letters `DSC`, followed by five numbers, a period, and the three letters `.JPG`.

The regular expression is correct, as far as it goes; it also includes two additional delimiters, `^` for beginning of line and `$` for end of line. The regex excludes false positives as long as the pattern consists of the three uppercase letters `DSC` followed by any five sections and the strings `.JPG`. However, a more precise regular expression would be `DSC067((4[35])|(5[14]))\.JPG`.

Again, the results returned by `rex` from the *tdda* library end up in the middle of the pack in terms of precision. While `rex` successfully matches the `DSC` and `JPG` portions of the pattern, it can produce a false positive for the digits in the sequence.

RegExTractor

To demonstrate the capabilities of RegExTractor [19], another Python regex extractor, I used random German license plates. Listing 4 first outputs the license plates followed by the regular expression that matches all license plates.

However, the regular expression that RegExTractor outputs is not correct. In particular, the last partial pattern of `[0-9][0-9A-Z]{1,4}` allows any digit followed by a combination of one to four capital letters and digits.

A more accurate solution would be the subexpression `[0-9]{1,4}E+` for one to four digits, followed only by the capital letter `E` for electric cars. In testing, this problem occurred repeatedly. Consequently, I cannot recommend RegExTractor.

Conclusions

Deriving regular expressions based on existing text fragments and patterns helps to analyze and recognize similarities in more complex patterns. The tools I tested work well but not always without error. Some tools generated regular expressions that were more generic than they actually should be based on the text fragments, resulting in searches that returned more matches than desired. In particular, these tools may include results that don't actually match the search patterns, resulting in false positives and some fuzziness.

Regular expressions are complex, inherently mapping a fragment and pattern differently. The performance of these tools does deserve credit given the complexity of the tasks. For these generators to be more useful in the future, increased precision would be desirable. ■■■

Listing 3: rex from tdda

```
$ ls images/*.JPG
DSC06743.JPG
DSC06745.JPG
DSC06751.JPG
DSC06754.JPG
$ ls images/*.JPG | python3 tdda/rexpy/rexpy.py
^DSC\d{5}\.JPG$
```

Listing 4: Running RegExTractor

```
$ python main.py
Kennzeichen:
A-BC 1234
CB-LN 5246E
FR-CG 1554
TUT-R 712
AA-LN 5E

The regex for this:
[A-Z][A-Z]{0,2}\-[A-Z][A-Z]?\ [0-9][0-9A-Z]{1,4}
```

Acknowledgement

The author would like to thank Axel Beckert and Arne Wichmann for their help and critical comments during the preparation of this article.

Author

Frank Hofmann works on the road – preferably from Berlin, Geneva, and Cape Town – as a developer, trainer, and author. He is also a co-author of the *Debian Package Management* book (*http://www.dpmb.org*).

Info

[1] Regular expression: *https://en.wikipedia.org/wiki/Regular_expression*

[2] grep everything: *http://noone.org/blog/English/Computer/Shell/grep%20everything.futile*

[3] xmlgrep: *https://linux.die.net/man/1/xmlgrep*

[4] "Search more efficiently with Ugrep" by Karsten Günther, *Linux Magazine*, issue 245, April 2021, *https://www.linux-magazine.com/Issues/2021/245/Tracked-Down/(language)/eng-US*

[5] Regular-Expressions.info: *http://www.regular-expressions.info/*

[6] Regex DB: *https://rgxdb.com/*

[7] Friedly, Jeffrey. *Mastering Regular Expressions*, O'Reilly Media, Inc., 2006: *http://regex.info/book.html*

[8] RegEx101: *https://regex101.com/*

[9] txt2re: *http://www.txt2re.com/index_php3.html*

[10] grex: *https://github.com/pemistahl/grex*

[11] regexgen: *https://github.com/devongovett/regexgen*

[12] Regex Generator by Olaf Neumann: *https://regex-generator.olafneumann.org/*

[13] rgxg: *https://rgxg.github.io/*

[14] txt2regex: *https://aurelio.net/projects/txt2regex/*

[15] Fink: *https://www.finkproject.org/*

[16] RegexGenerator: *https://github.com/dbuhlbrown/Regex-Generator*

[17] regex-generator-lib (Python): *https://pypi.org/project/regex-generator-lib/*

[18] tdda: *http://www.tdda.info/*

[19] RegExTractor: *https://github.com/iuliux/RegExTractor*

Track down race conditions with Go

Rat Race

If program parts running in parallel keep interfering with each other, you may have a race condition. Mike Schilli shows how to instruct the Go compiler to detect these conditions and how to avoid them in the first place. *By Mike Schilli*

If programmers are not careful, program parts that are running in parallel will constantly get in each other's way, whether as processes, threads, or goroutines. If you leave the order in which system components read or modify data to chance, you are adding time bombs to your code. They will blow up sooner or later, leaving you with runtime errors that are difficult to troubleshoot. But how do you avoid them?

The common assumption that components will run in the same order that a program calls them is a fallacy – one easily refuted with an example such as in Listing 1. But coincidence can also be a factor. It is quite possible for something to work once but then crash after a small, and often unrelated, change to the code. The load on the system you are using can also play a role: Something may work flawlessly in slack times but fall apart unexpectedly under a heavy load.

The fact that unsynchronized goroutines do not run in the order in which they are defined, even if the program starts them one after the other, is nicely illustrated by Listing 1 [1] and the output in the upper part of Figure 1. Although the `for` loop starts goroutine `0` first, followed by `1`, then `2`, and so on, as defined by the index numbers in `i`, the upper part of Figure 1 makes it clear from the compiled program's output that chaos reigns, and the goroutines write their messages to the output as a wildly confusing mess.

Listing 1: orderfail.go

```
package main
import (
  „fmt“
)

func main() {
  done := make(chan bool)
  n := 10

  for i := 0; i < n; i++ {
    go func(id int) {
      fmt.Printf(„goroutine %d\n“,
  id)
      done <- true
    }(i)
  }

  for i := 0; i < n; i++ {
    <-done
  }
}
```

```
$ go build orderfail.go
$ ./orderfail
Running 6
Running 1
Running 0
Running 5
Running 7
Running 9
Running 3
Running 4
Running 8
Running 2

$ go build orderok.go
$ ./orderok
Running 0
Running 1
Running 2
Running 3
Running 4
Running 5
Running 6
Running 7
Running 8
Running 9
```

Figure 1: Unsynchronized goroutines run in an idiosyncratic order.

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.

Each of the 10 `go func()`s created in the `for` loop passes the current loop index as a parameter to the respective goroutine, completely according to the textbook, so that they do not all share the same variable. Also, to stop the program from terminating immediately after the `for` loop ends – instead of making it wait until all the goroutines have completed their work – each goroutine sends a message to the `done` channel at the end of its working life. The final `for` loop starting in line 17 collects the messages from there and does not terminate until the last goroutine has said goodbye.

One by One

But if you really want goroutine `0` to start first, then goroutine 1, and so on, you need to use a synchronization mechanism, such as channels or mutex

Listing 2: orderok.go

```
01 package main
02 import (
03   „fmt“
04 )
05
06 func main() {
07   done := make(chan bool)
08   n := 10
09
10   starters := make([](chan bool),
   n)
11   for i := 0; i < n; i++ {
12     starters[i] = make(chan bool)
13   }
14
15   for i := 0; i < n; i++ {
16     go func(id int) {
17       <-starters[id]
18       fmt.Printf(„Running %d\n“,
   id)
19       if id < n-1 {
20         starters[id+1] <- true
21       }
22       // [... DO WORK ...]
23       done <- true
24     }(i)
25   }
26
27   starters[0] <- true
28
29   for i := 0; i < n; i++ {
30     <-done
31   }
32 }
```

constructs, to make sure that the Go runtime maintains the desired order, defying the natural chaos.

Listing 2 demonstrates this with an array of 10 channels. The goroutines all start blocking, shortly after they are called, and wait until a message arrives on the channel assigned to them. This unblocks the read statement from the channel array `starters` in line 17, and the goroutine moves on to printing its "Running" message. At first, none of the channels will have a message, but line 27 after the `for` loop then starts a chain of events by writing a value to the first channel.

This releases the goroutine with the `id` of `0`, because the block in its read statement in line 17 is now lifted. The routine then outputs its message and, to keep things ticking along, writes to the channel with the `id+1` (i.e., 1). This in turn triggers goroutine 1, which in turn triggers goroutine 2. This merry dance continues in a controlled manner until goroutine `9` initiates the completion of the program.

This approach naturally reduces the concurrency of all goroutines, which now do not all start quasi-simultaneously but wait for each other – but only as long as the individual goroutine needs to trigger the next one in the channel. What happens afterwards within the individual goroutines (commented in line 22 with the placeholder `DO WORK`), is again a quasi-simultaneous affair.

Listing 3: airline.go

```
01 package main
02 import (
03   „fmt“
04   „time“
05 )
06
07 func main() {
08   seats := 1
09
10   for i := 0; i < 2; i++ {
11     go func(id int) {
12       time.Sleep(100 * time.Millisecond)
13       if seats > 0 {
14         fmt.Printf(„%d booked!\n“, id)
15         seats = 0
16       } else {
17         fmt.Printf(„%d missed out.\n“, id)
18       }
19     }(i)
20   }
21
22   time.Sleep(1 * time.Second)
23   fmt.Println(„“)
24 }
```

There Can Only Be One Winner

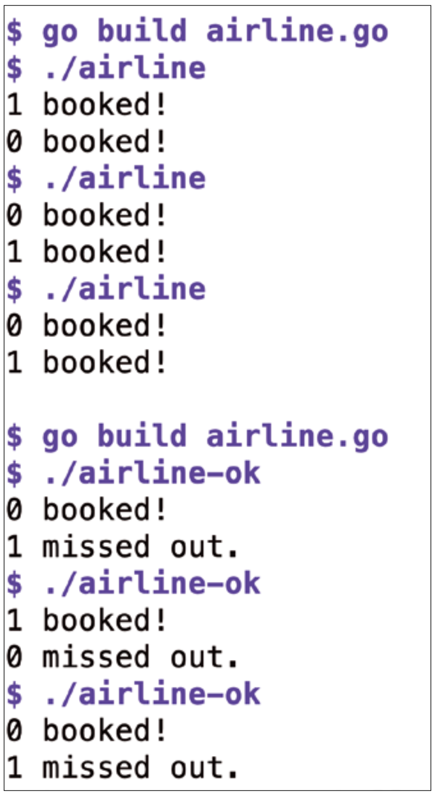
The disastrous consequences that race conditions can cause in an application are illustrated by an airline's booking program in Listing 3. It detects in line 13 that there is still one seat available on the plane in the variable `seats`, which is shared by two different goroutines. It then outputs a success message to the user and sets the number of remaining seats to zero.

However, there are two parallel goroutines fighting over the booking in the `for` loop starting in line 10. While one rejoices and prints the success message, the second goroutine also tests the variable `seats`, which is still set to 1, and proceeds to book the seat as well. The result is an overbooked plane and angry passengers.

The output at the top of Figure 2 shows that Listing 3 does indeed allow repeated double-bookings – exacerbated by the length of the microsleep instruction at line 12, simulating the actual booking process. This is not what a customer, or an airline, wants.

The root of the problem is obvious: Two concurrent program threads share the variable `seats` during the time that elapses between the check `seats > 0` in line 13 and the variable being reset by `seats = 0` in line 15. If the second goroutine is performing a check while the first is booking the seat, the second goroutine erroneously thinks it has a free seat because `seats` is still set to 1. A booking error is inevitable.

The problem can be solved by either performing the check and setting



```
$ go build airline.go
$ ./airline
1 booked!
0 booked!
$ ./airline
0 booked!
1 booked!
$ ./airline
0 booked!
1 booked!

$ go build airline.go
$ ./airline-ok
0 booked!
1 missed out.
$ ./airline-ok
1 booked!
0 missed out.
$ ./airline-ok
0 booked!
1 missed out.
```

Figure 2: Without synchronization, two goroutines happily book the same seat.

Listing 4: airline-ok.go

```
01 package main
02 import (
03   „fmt"
04   „time"
05 )
06
07 func main() {
08   seats := 1
09   booking := make(chan bool, 1)
10
11   for i := 0; i < 2; i++ {
12     go func(id int) {
13       time.Sleep(100 * time.Millisecond)
14       booking <- true
15       if seats > 0 {
16         fmt.Printf(„%d booked!\n", id)
17         seats = 0
18       } else {
19         fmt.Printf(„%d missed out.\n", id)
20       }
21       <-booking
22     }(i)
23   }
24
25   time.Sleep(1 * time.Second)
26   fmt.Println(„")
27 }
```

the variable in a single atomic statement or by declaring the program area containing both statements to be a critical section that locks out other goroutines as long as one goroutine is working in it.

Listing 4 shows a possible solution to the problem using a buffered `booking` channel with a depth of 1, as created by the `make` statement in line 9. Thanks to the buffer, one goroutine can write a value into the channel without it immediately blocking [2]. But if the next goroutine tries to send a value into the channel, it blocks until someone else has extracted the buffered value, and this happens at the end of the critical section in line 21.

With this safeguard in place, only one goroutine traverses the critical section at any given time, and it doesn't matter how long it takes to check or set the `seats` variable, because no one can interfere in the meantime. The lower part of Figure 2 then also shows that only one goroutine at a time makes the booking, while the other goroutine reports that there are no more seats available – to the disappointment of the passenger who wants to book. But that's how things have to be.

Reporting Speeders

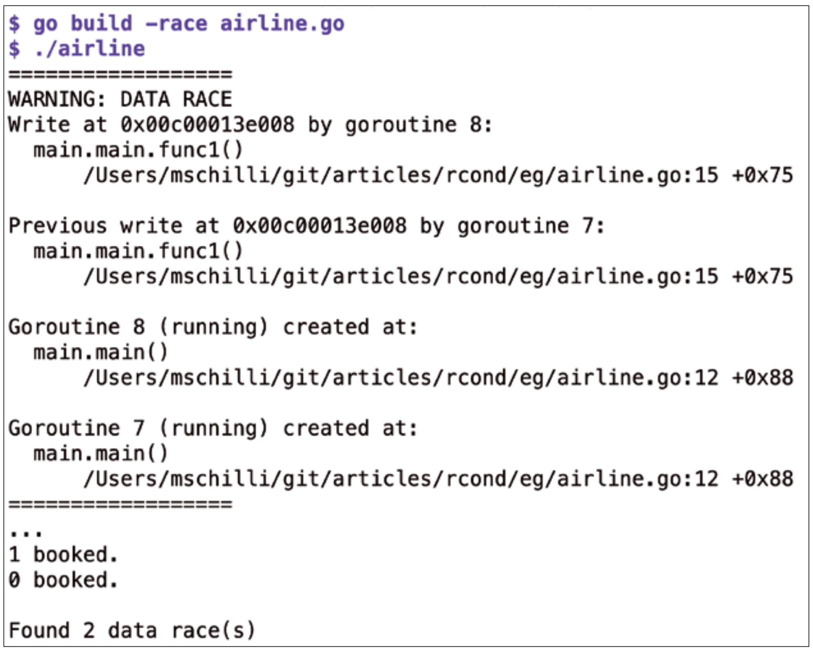
During development, Go helps you detect race conditions – if you compile the source code with the `-race` option. If two goroutines then race for a variable, the Go runtime detects this in the moment and outputs a corresponding error message (Figure 3). However, this requires the program to enter the subrange that triggers the problem during the test run. This makes it important for the test suite to cover the code as completely as possible.

Classic Computer Science

The classical method for slowing down racers relies on what are known as mutex (mutually exclusive) constructs. Go's *sync* package provides mutex elements that use `Lock()` to create a roadblock for subsequent goroutines and clear it again with `Unlock()` after the critical region has been navigated.

Listing 5 shows the implementation of the `book()` airplane-seat booking function; the call to the function in Line 16 is surrounded by a pair of mutexes. Before the call, line 15 uses `mutex.Lock()` to block the section for subsequent goroutines. After the job is done, line 17 uses `mutex.Unlock()` to lift the block again.

For the sake of clarity, the `book()` function now encapsulates the posting process starting in line 25. As its input, it expects the ID of the goroutine and a pointer to the `seats` variable, whose value it adjusts accordingly in case of a successful booking.



```
$ go build -race airline.go
$ ./airline
==================
WARNING: DATA RACE
Write at 0x00c00013e008 by goroutine 8:
  main.main.func1()
      /Users/mschilli/git/articles/rcond/eg/airline.go:15 +0x75

Previous write at 0x00c00013e008 by goroutine 7:
  main.main.func1()
      /Users/mschilli/git/articles/rcond/eg/airline.go:15 +0x75

Goroutine 8 (running) created at:
  main.main()
      /Users/mschilli/git/articles/rcond/eg/airline.go:12 +0x88

Goroutine 7 (running) created at:
  main.main()
      /Users/mschilli/git/articles/rcond/eg/airline.go:12 +0x88
==================
...
1 booked.
0 booked.

Found 2 data race(s)
```

Figure 3: The program, built with the `-race` option in place, detects race conditions at runtime.

Listing 5: airline-mutex.go

```
01 package main
02 import (
03   „fmt“
04   „sync“
05   „time“
06 )
07
08 func main() {
09   seats := 1
10   mutex := &sync.Mutex{}
11
12   for i := 0; i < 2; i++ {
13     go func(id int) {
14       time.Sleep(100 * time.Millisecond)
15       mutex.Lock()
16       book(id, &seats)
17       mutex.Unlock()
18     }(i)
19   }
20
21   time.Sleep(1 * time.Second)
22   fmt.Println(„“)
23 }
24
25 func book(id int, seats *int) {
26   if *seats > 0 {
27     fmt.Printf(„%d booked!\n“, id)
28     *seats = 0
29   } else {
30     fmt.Printf(„%d missed out.\n“, id)
31   }
32 }
```

Listing 6: airline-wg.go

```
01 package main
02 import (
03   „fmt“
04   „sync“
05   „time“
06 )
07
08 func main() {
09   seats := 1
10   wg := &sync.WaitGroup{}
11
12   for i := 0; i < 2; i++ {
13     go func(id int) {
14       time.Sleep(100 * time.Millisecond)
15       wg.Wait()
16       wg.Add(1)
17       book(id, &seats)
18       wg.Done()
19     }(i)
20   }
21
22   time.Sleep(1 * time.Second)
23   fmt.Println(„“)
24 }
25
26 func book(id int, seats *int) {
27   if *seats > 0 {
28     fmt.Printf(„%d booked!\n“, id)
29     *seats = 0
30   } else {
31     fmt.Printf(„%d missed out.\n“, id)
32   }
33 }
```

Height-Adjustable Barrier

One alternative to the mutex construct is provided by the `sync.WaitGroup` construct, also from the *sync* package of the Go core library. Its `Add()` function sets up a block with an adjustable height. A subsequent `Wait()` waits for the block to be lifted, which a call to the `Done()` function is happy to do.

The smart thing here is that multiple program parts can set up any number of blocks through subsequent calls to `Add()`. You then need the same number of calls to `Done()` before `Wait()` grants passage again.

The code in Listing 6 waits for this to happen before starting the critical section with a prophylactic `Wait()` statement in line 15, but the code continues immediately because there are no blocks in the way in the initial state. Line 16 then calls `Add(1)` to add a barrier just before booking. Afterwards, `book()` in line 17 can calmly make the booking, before line 18 uses `Done()` to take down the barrier again.

Conclusions

As always in Go, there are several possible solutions to the challenges of controlling racing goroutines – after all, these are well-known programming problems.

If two related instructions such as checking and placing a shared variable cannot be done atomically, you need to use a synchronization tool to temporarily block the critical time window between the instructions – to keep out any program threads attempting to rush in.

And always remember to clear the block later on, even if the booking fails for some reason. Otherwise, you have built a permanent lock into your program, which might consume resources or block access to some resources entirely. Test every case to make sure! ■■■

Info

[1] Go: "Programming Snapshot – Golang" by Mike Schilli, Linux Magazine issue 250, September 2021, p. 54-60

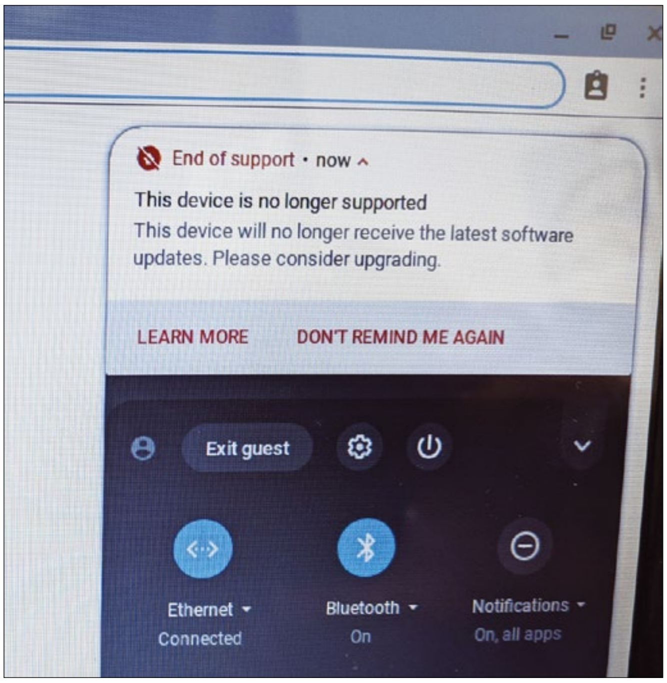
[2] Listings for this article: *ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/251/*

Breathe life back into your old Chromebook with GalliumOS

Heavy Metal

A tailor-made image like GalliumOS "Bismuth" will keep your Chromebook healthy after its expiration date. *By DeeAnn Little and Markus Feilner*

Every computer user is familiar with the scenario. Your favorite piece of hardware tells you "End of Support – sorry, this device is no longer supported" (Figure 1) or "This device will no longer receive software updates," followed by the laconic advice to "Please consider upgrading."



End of support • now

This device is no longer supported

This device will no longer receive the latest software updates. Please consider upgrading.

Figure 1: Is this the end of my beloved Chromebook? Or could this be the time to turn to Linux?

No software is supported forever. The continual task of coding updates and bug fixes takes time and energy, and after some predetermined support interval, the vendor just stops providing patches for old versions in order to focus on later systems.

With Chrome OS and Chromebook computers, the task of providing support is even more complicated. The term *Chromebook* does not define a single hardware system. Instead, Google contracts with several different hardware vendors to produce Chromebooks, including Samsung, HP, Toshiba, Asus, Dell, and many others. Google must maintain separate ISO images for each hardware system. Every Chromebook has a published expiration date, known as the Auto Update Expiration (AUE) date, which defines the last date when the operating system will receive an update. In theory, you could continue to operate the Chromebook after the AUE, but the system will be insecure, and the fact that Chromebooks are so heavily reliant on the Internet makes it even more dangerous to operate the system without security updates.

I bought my little Acer Chromebook C710 at a Walmart in 2014, and it has been a special companion ever since: under shady trees in California, on the icy Westfjords of Iceland, or in a tiny Prague hotel room the size of a double bed. Today, my Chromebook acts as a kind of media station that plays videos from YouTube, as well as from media libraries and various portals on a 40-inch HD-ready TV (Figure 2). Hard disk space or battery life are no longer important. (See Table 1 for Acer C710 hardware specs.)

When I got the message that the system had reached its AUE and would no longer be supported, I resolved to find another solution that would keep my Chromebook alive. All this took me quite quickly to GalliumOS [1], which is specially tailored for Chromebook users and offers images for all processor families (Listing 1). Installation is simple and makes no special demands. A recovery image of Chrome OS is available in case you need to revert to the previous system at any time.

GalliumOS is based on Xubuntu and uses Xfce as its desktop, which sits well with the aged Acer Chromebook and its meager hardware resources. GalliumOS for Chromebooks in version 3.x goes by the name of “Bismuth” – like chromium and gallium, bismuth is a metal in the periodic table. Version 3.1, which I use in this article, is a maintenance update.

Depends on the Hardware

The steps for installing GalliumOS on a Chromebook depend heavily on the hardware. In the simplest case, Chromebooks behave like full-fledged PCs and let you boot from a USB stick. In that case, you could install GalliumOS or any other Linux variant in much the same way you would install Linux on any computer. However, some models (like my trusty Acer, for example) require a firmware update. Check out the GalliumOS hardware compatibility list [2] to determine if your Chromebook will need a firmware update.

A sticker on the underside of your Chromebook should provide system hardware details. You can also enter the local URL *chrome://system* or check the menu *Settings* | *About Chrome OS* | *Google Chrome OS*.

Look for the model number and hardware ID, which you can use to find your device in the GalliumOS hardware compatibility list. Dual boot, virtual machine (VM), or complete replacement are possible options, but every option might not be available for every device.

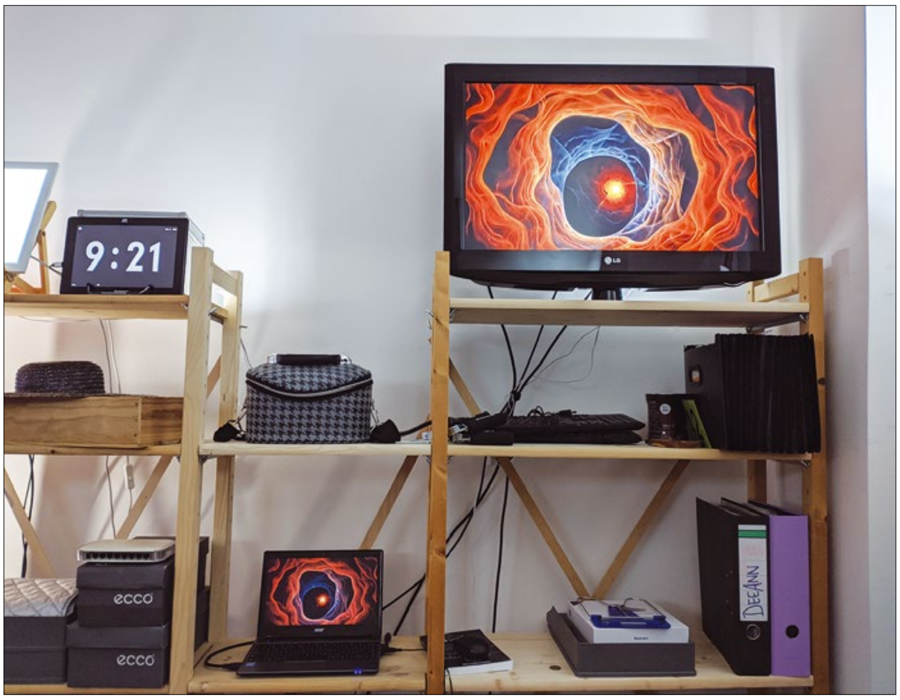


Figure 2: Thanks to the Ubuntu-based GalliumOS and an external screen, the Acer C710 – bottom left in the picture – provides useful service as a media station and is also happy to play DRM-protected content.

Table 1: Acer C710 Hardware

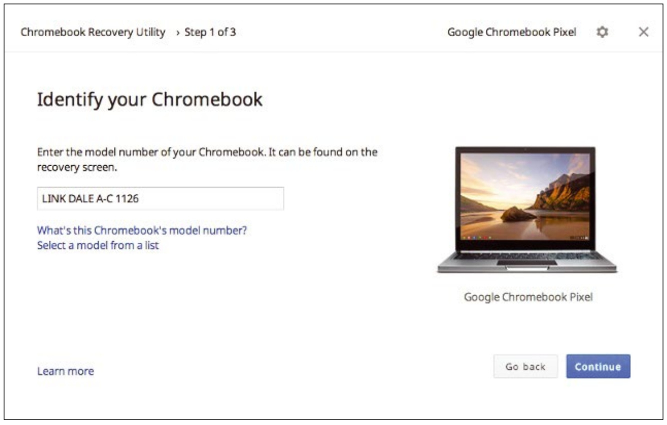
CPU	1.1GHz Intel Celeron 847 (dual-core)
RAM	2GB DDR3 SDRAM
Hard disk	SSD (16GB)
Display	1366x768, HDMI, VGA
Webcam	1 megapixel
Network	Broadcom BCM57785 (Gigabit Ethernet), Qualcomm Atheros AR9462 (802.11 a/b/g/n), Foxconn T77H348.02 (Bluetooth)
Miscellaneous	3 USB 2.0, SD card reader
Price	New: ~$200 (2013) Used: ~$50 to $120 (2021)

Listing 1: GalliumOS Images by CPU Type

```
../
TORRENTS/                             22-Dec-2019 22:17          -
CHECKSUMS-galliumos-3.1.gpg           22-Dec-2019 22:00       3364
galliumos-3.1-apollolake.iso          22-Dec-2019 20:16 1231513600
galliumos-3.1-baytrail.iso            22-Dec-2019 20:16 1231521792
galliumos-3.1-braswell.iso            22-Dec-2019 20:16 1231521792
galliumos-3.1-broadwell.iso           22-Dec-2019 20:16 1231513600
galliumos-3.1-haswell.iso             22-Dec-2019 20:16 1231523840
galliumos-3.1-kabylake.iso            22-Dec-2019 20:16 1231513600
galliumos-3.1-samus.iso               22-Dec-2019 20:16 1231697920
galliumos-3.1-sandyivy.iso            22-Dec-2019 20:16 1231513600
galliumos-3.1-skylake.iso             22-Dec-2019 20:16 1231521792
galliumos-generic-coreimage-3.1.tgz   16-Dec-2019 03:29 1160948557
```

Listing 2: Onto the USB Stick

```
$ dd if=/home/dlittle/Downloads/galliumos-3.1-sandyivy.iso of=/dev/sdb1
  status=progress
```


Chromebook Recovery Utility › Step 1 of 3

Google Chromebook Pixel

Identify your Chromebook

Enter the model number of your Chromebook. It can be found on the recovery screen.

LINK DALE A-C 1126

What's this Chromebook's model number?
Select a model from a list

Google Chromebook Pixel

Learn more

Figure 3: The Chrome Book Recovery Utility creates the recovery system for emergencies.

Listing 3: Launching the Firmware Utility

```
$ curl -LO mrchromebox.tech/firmware-util.sh
$ sudo install -Dt /usr/local/bin -m 755 firmware-util.sh
$ sudo firmware-util.sh
```

ISO and Recovery Image

The first thing to do is to download the appropriate ISO file from the Gallium FTP directory [3] and drop it onto a USB stick, either using dd (Listing 2) or an alternative disk tool such as the SUSE Studio Imagewriter, Ubuntu's Startup Disk Creator, or a Windows tool like Rufus [4].

The second step is to create a backup and a recovery image. Back up any important files by moving them to a USB stick or to the cloud. Several methods are available for creating a recovery disk. Linux users can take the long route via a shell script provided by Google; however, it is better to install the browser *Recovery Extension* and follow the steps. Both methods have their advantages; however, keep in mind that the Chrome OS browser extension is only available if the Chromebook is still fully functional (Figure 3).

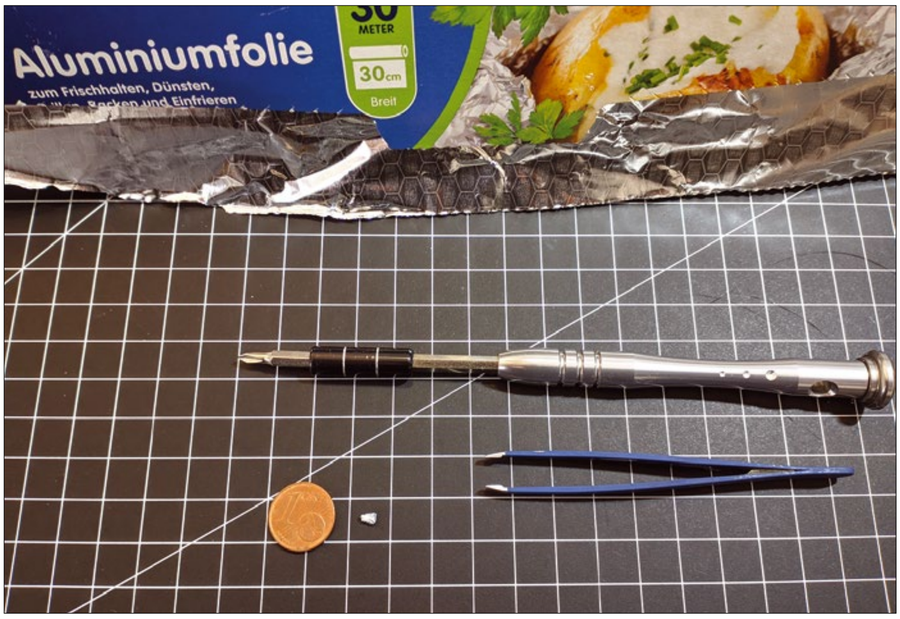


Figure 4: The required aluminum rice grain compared to the size of a euro cent coin.

Firmware Update

The steps for replacing the firmware can vary depending on the hardware. On some systems (such as the small Acer), changing the firmware even requires bridging a write-protect jumper. If you don't have a suitable cover for the jumper, you can use aluminum foil and tweezers. I found a YouTube video with the necessary steps for my system [5]. Be sure you read the instructions for the firmware and device – any wrong procedure can brick your cherished hardware. Detailed documentation describes both the recovery [6] and developer modes [7].

The tools you need are a Phillips screwdriver, tweezers, and a strip of aluminum foil (about 3 centimeters long and 1 centimeter wide) (Figure 4). First, switch off the Chromebook and disconnect it from the mains. Then you have to remove the battery (the Acer Chromebook has a small lever for this) and unlock the cover at the bottom (only one screw) and pull it out. Usually, the screw is located under the sticker for the long-expired warranty ("Warranty void if seal broken.") The jumper is found under a plastic cover (Figure 5).

Now roll up the aluminum foil and compress it to the size of a grain of rice; you can press it into the jumper with

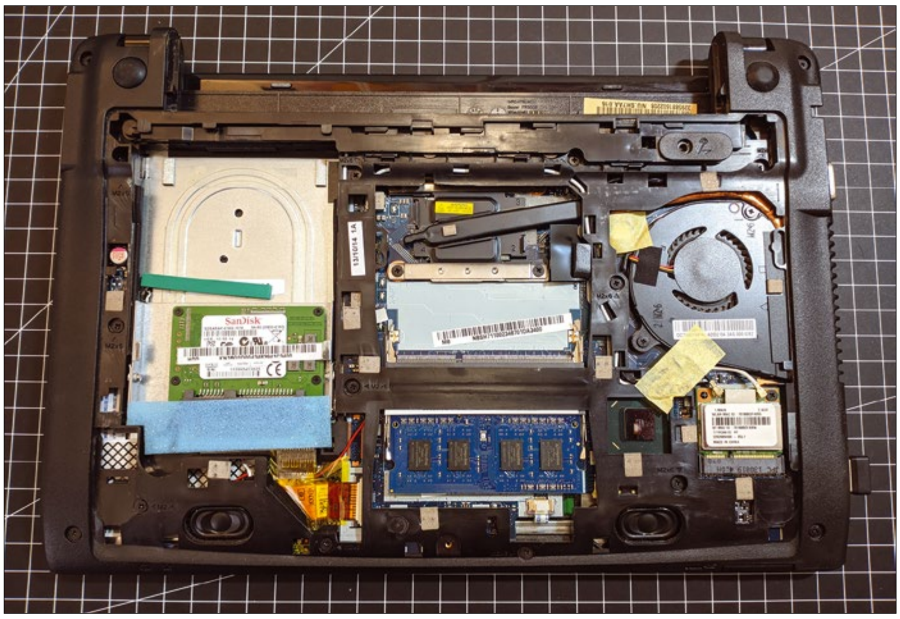


Figure 5: The open underside of the Acer C710 with the plastic cover over the jumper (bottom left center in the picture).

the help of the tweezers, where it bridges the contacts and thus lifts the firmware's write protection (Figure 6). This is fine for installing the new operating system, but certainly not as a permanent solution. Now you can reassemble the Chromebook, plug in the battery, connect the power supply, and start recovery mode.

Recovery and Developer Mode

The keyboard shortcut Esc + F3 + Power boots the Chromebook's recovery system; when you get there, pressing Ctrl + D – not displayed on the screen – boots into Developer mode. In the graphical login manager, log in as a guest and start a Chrome Shell (Crosh) with Ctrl + Alt + T. In a second browser tab, go to the MrChromebox [8] website, which provides the *Chrome OS Device Firmware Utility Script* (Listing 3), which can be downloaded using `curl` at the Crosh terminal (Listing 3, first line).

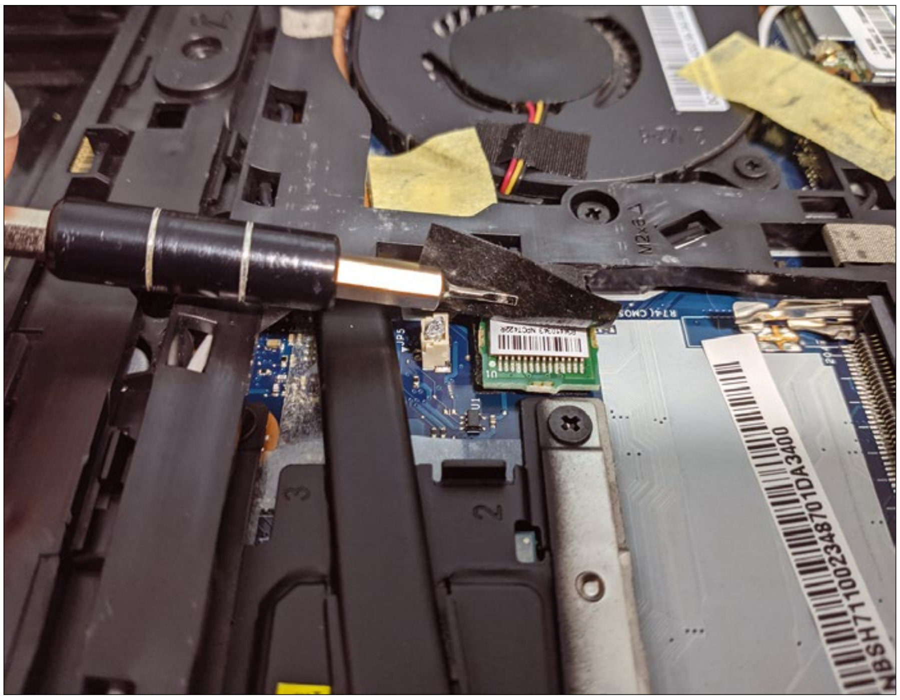


Figure 6: The jumper has already been fitted with an aluminum "rice grain" for bridging.

Listing 4: firmware-util.sh

```
#!/bin/bash
#
# This script offers provides the ability to update the
# Legacy Boot payload, set boot options, and install
# a custom coreboot firmware for supported
# Chrome OS devices
#
# Created by Mr.Chromebox <mrchromebox@gmail.com>
#
# May be freely distributed and modified as needed,
# as long as proper attribution is given.
#

#where the stuff is
script_url="https://raw.githubusercontent.com/MrChromebox/
  scripts/master/"

#ensure output of system tools in en-us for parsing
export LC_ALL=C

#set working dir]\
if cat /etc/lsb-release | grep "Chrom" > /dev/null 2>&1; then
  # needed for ChromeOS/ChromiumOS v82+
  mkdir -p /usr/local/bin
  cd /usr/local/bin
else
  cd /tmp
fi

#get support scripts
echo -e "\nDownloading supporting files..."
rm -rf firmware.sh >/dev/null 2>&1
rm -rf functions.sh
 >/dev/null 2>&1
rm -rf sources.sh >/dev/null 2>&1
curl -sLO ${script_url}firmware.sh
rc0=$?
curl -sLO ${script_url}functions.sh
rc1=$?
curl -sLO ${script_url}sources.sh
rc2=$?
if [[ $rc0 -ne 0 || $rc1 -ne 0 || $rc2 -ne 0 ]]; then
  echo -e "Error downloading one or more required files;
    cannot continue"
  exit 1
fi

source ./sources.sh
source ./firmware.sh
source ./functions.sh

#set working dir
cd /tmp

#do setup stuff
prelim_setup
[[ $? -ne 0 ]] && exit 1

#show menu
menu_fwupdate
```

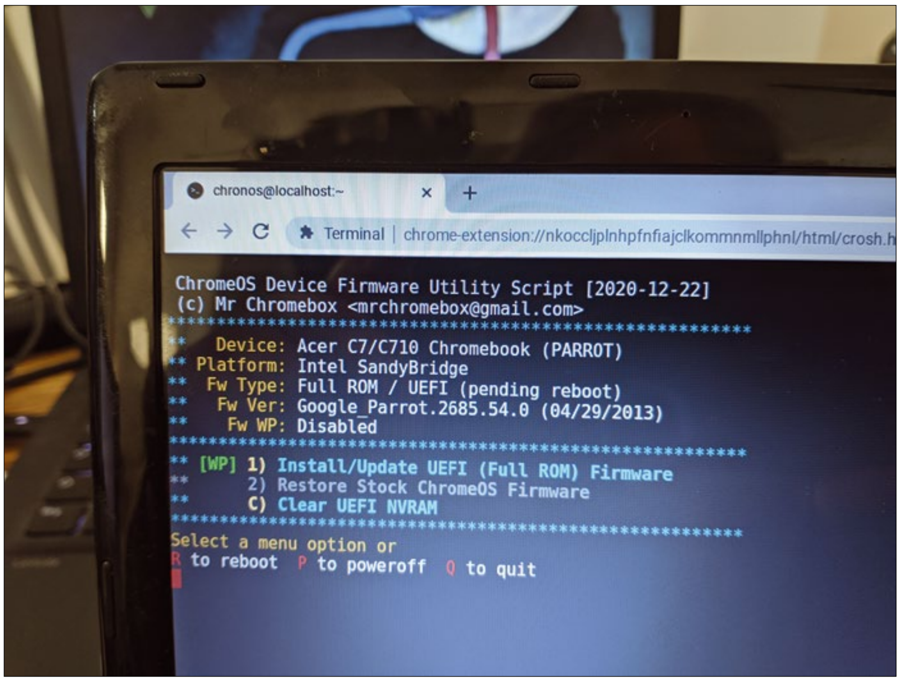
Then proceed to install and launch (second and third line).

The largely self-explanatory utility script (Listing 4) installs either a UEFI update or a stock recovery image for Chrome OS. Some dead devices can also be persuaded to cooperate again by pressing *C* (*Clear UEFI NVRAM*) (Figure 7). Again, caution is advised; only start the firmware update if you have the correct hardware specs as identified by the script. If the aluminum rice grain is still in the right place, the script shows the note `Fw WP: Disabled` in the fifth line of the hardware information. Then it should run without any problems, including booting from the USB stick.

Testing and Installing Gallium

At this point, the paths with and without tinfoil meet up again; now the USB stick with GalliumOS, created earlier on, is used for booting. As with all modern Linuxes, you can see how things will work using a live system before an installation on the hard disk gives you the hard facts. This includes accessing the hard disk previously used by Chrome OS, although recovery mode has permanently deleted it for security reasons.

Anyone who decides to install GalliumOS on the hard disk will quickly find themselves at home in Ubuntu's default installer, Ubiquity. From then on, the Chromebook behaves like any other PC or laptop system. The advantage is that the Linux image is already adapted to the existing hardware, much like with Apple. Now, at last, assuming everything worked out, the aluminum rice grain has to be removed. From now on, the Chromebook will boot willingly from USB.


chronos@localhost:~
Terminal | chrome-extension://nkoccljplnhpfnfiajclkommnmllphnl/html/crosh.h

```
ChromeOS Device Firmware Utility Script [2020-12-22]
(c) Mr Chromebox <mrchromebox@gmail.com>
*********************************************************
**     Device: Acer C7/C710 Chromebook (PARROT)
**   Platform: Intel SandyBridge
**    Fw Type: Full ROM / UEFI (pending reboot)
**     Fw Ver: Google_Parrot.2685.54.0 (04/29/2013)
**      Fw WP: Disabled
*********************************************************
** [WP] 1) Install/Update UEFI (Full ROM) Firmware
**      2) Restore Stock ChromeOS Firmware
**      C) Clear UEFI NVRAM
*********************************************************
Select a menu option or
R to reboot  P to poweroff  Q to quit
```

Figure 7: The Chrome OS Device Firmware Utility Script finds and installs the matching firmware image.

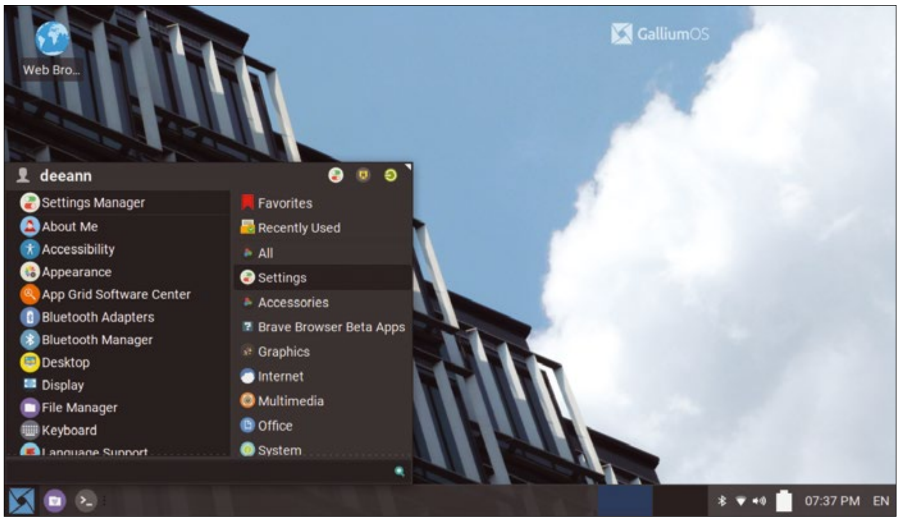


Figure 8: GalliumOS comes with the lean Xfce desktop by default, a perfect choice for the ancient hardware of the Acer C710.

Xfce Desktop with HDMI Output

The Xfce desktop on GalliumOS (Figure 8) has everything a modern desktop needs. The layout and many settings are well tailored to the small display and low-powered hardware of the Acer.

Having said this, the start menu already has a surprising number of entries, for example, a separate menu item for *GalliumOSUpdate* – a link to a root shell that executes `apt -qq update` (Figure 9). The `-qq` parameter ensures that all output except error messages is suppressed.

Enough Power

The Chromebook's performance feels the same under GalliumOS, even if the fans now sound a little louder, especially during videos. However, `lm-sensors` quickly shows that there is no reason for concern. The (still first) battery lasts for around two hours, so not quite as long as with Chrome OS.

Info

[1] GalliumOS: *https://galliumos.org/*

[2] Hardware compatibility and firmware: *https://wiki.galliumos.org/Hardware_Compatibility*

[3] GalliumOS download: *https://galliumos.org/download*

[4] Rufus: *https://rufus.ie*

[5] Disabling Acer C710 Write Protect jumper: *https://www.youtube.com/watch?v=xDCMjM2-oGo*

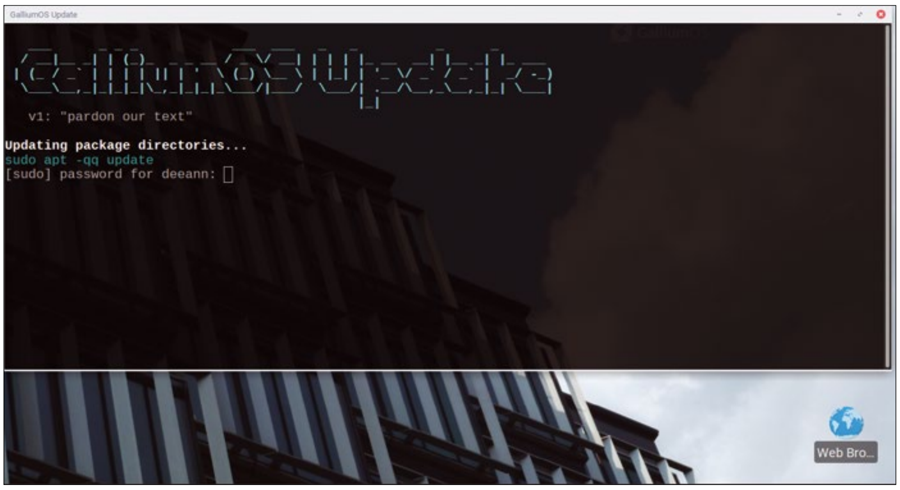
[6] Recovery mode: *https://support.google.com/chromebook/answer/1080595*

[7] Developer mode: *https://chromium.googlesource.com/chromiumos/docs/+/HEAD/developer_mode.md*

[8] MrChromebox: *https://mrchromebox.tech/#fwscript*

Touchpad, audio, webcam, WiFi, and Ethernet work without further intervention, as does my LG TV connected via HDMI with 720p resolution. The TV is about the same age as the Chromebook and a perfect match. YouTube, Amazon Prime, and Jitsi Meet work without any problems; only videoconferencing causes frequent lag. (The processor or I/O systems are probably struggling with the data demands of videoconferencing.)

It looks like these life-extending measures have paid off. A USB mouse and a USB keyboard complete the setup. The ensemble has been running smoothly for several weeks, delivering music, videos, TV, or movies to the office every day. Take that, Google! ■■■



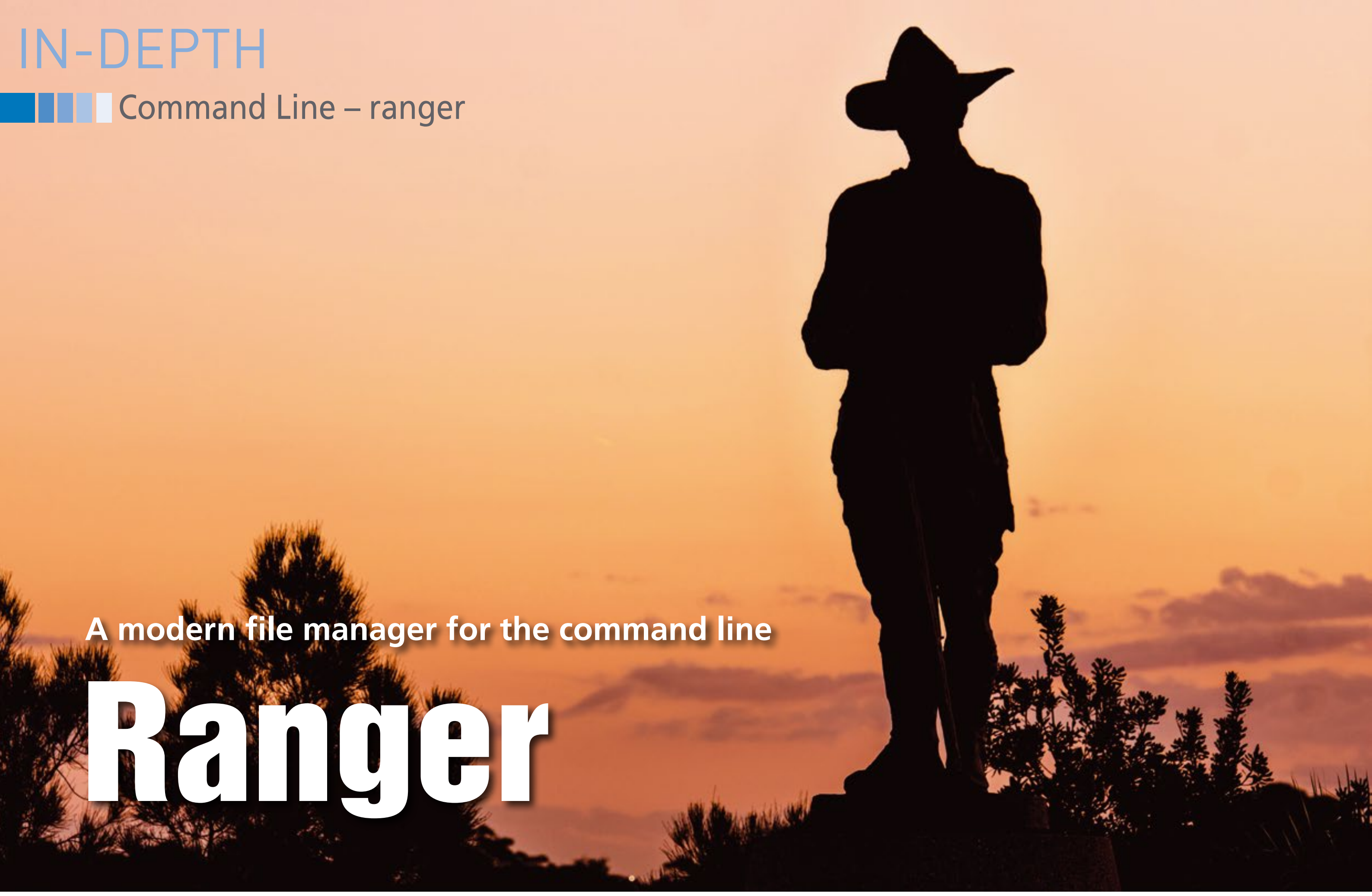
```
  v1: "pardon our text"

Updating package directories...
sudo apt -qq update
[sudo] password for deeann:
```

Figure 9: **GalliumOSUpdate shows its Ubuntu/Debian roots when the updater runs** `sudo apt -qq update`**.**

The Authors

DeeAnn Little from California has been living in Regensburg, Germany, since 2016. She has worked in the open-source environment for more than 25 years on hardware, development, research, and documentation.

Markus Feilner, technology and network policy editor at Mailbox.org, has been involved with Linux since 1994 and was deputy editor-in-chief of Linux Magazin and iX. His company, Feilner IT, specializes in documentation and OSI layers 8, 9, and 10.



Ranger

Ranger offers a wealth of commands – many with alternatives – and less reliance on a mouse. *By Bruce Byfield*

Linux has all the necessary commands for manipulating files, everything from `ls` and `cp` to `rm` and `mv`. However, file managers have the advantage of centralizing all the commands and requiring less knowledge in all except the most advanced circumstances. In fact, file managers remain so useful that Midnight Commander (`mc`) continues to be used at the command line 27 years after it was first released. A more recent command-line file manager is ranger [1], which uses many Vim keybindings and is written in Python, which allows for the easy creation of custom scripts.

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/*.

Ranger is available in many distributions, and the latest version can be downloaded from GitHub [1]. It requires Python 3 and includes a long list of suggested packages for full functionality, such as *w3m* for graphics display, *poppler-utils* for PDF display, and *python-pygments* for code highlighting. It creates the directory `~/.config/ranger/rc.conf` with the configuration files in Table 1 (shown in Figure 1).

The configuration files also function as documentation on how to customize; although, unless you have a knowledge of Python, these files may be of limited use. A few notes are given in the GitHub repository [2].

Upon startup, ranger first references the global configuration in `/etc/ranger/rc.conf` by default and then

Table 1: Ranger Configuration Files

`rifle.conf`	Configuration for `rifle`, the ranger file launcher
`commands.py`	Defines the default ranger console commands
`commands_full.py`	Reference file for custom commands
`rc.conf`	The main configuration file
`scope.sh`	Defines how to handle preview files

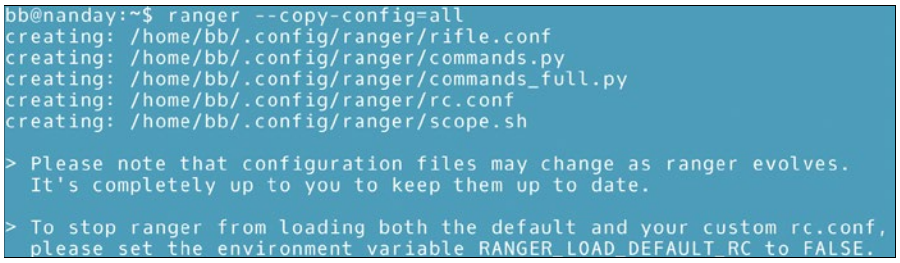
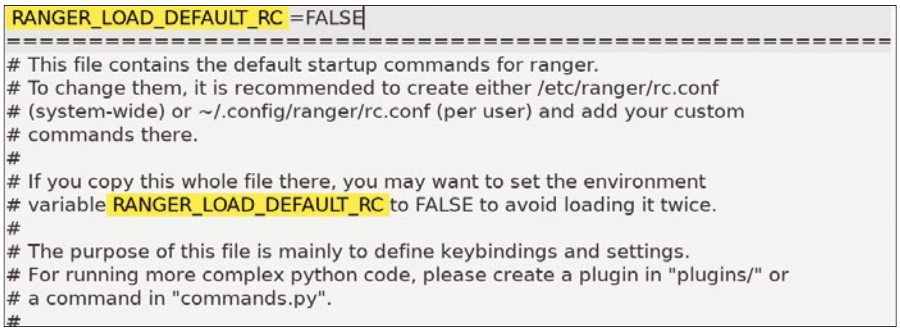

```
bb@nanday:~$ ranger --copy-config=all
creating: /home/bb/.config/ranger/rifle.conf
creating: /home/bb/.config/ranger/commands.py
creating: /home/bb/.config/ranger/commands_full.py
creating: /home/bb/.config/ranger/rc.conf
creating: /home/bb/.config/ranger/scope.sh

> Please note that configuration files may change as ranger evolves.
  It's completely up to you to keep them up to date.

> To stop ranger from loading both the default and your custom rc.conf,
  please set the environment variable RANGER_LOAD_DEFAULT_RC to FALSE.
```

Figure 1: Ranger distinguishes carefully between global and personal config files, suggesting you bypass the global copy and edit the personal copy.

`~/.config/ranger/rc.conf`. To avoid potential conflicts, you should add to the top of `rc.conf` the line `RANGER_LOAD_DEFAULT_RC =FALSE` so that ranger bypasses the global configuration, and make all your edits in `~/.config/ranger` (Figure 2). While making this change, you might also scan the local `rc.conf` for other customizations, such as whether ranger displays hidden files or if deletions require confirmation. The other files can be ignored unless you want to customize.



```
RANGER_LOAD_DEFAULT_RC =FALSE
==================================================================
# This file contains the default startup commands for ranger.
# To change them, it is recommended to create either /etc/ranger/rc.conf
# (system-wide) or ~/.config/ranger/rc.conf (per user) and add your custom
# commands there.
#
# If you copy this whole file there, you may want to set the environment
# variable RANGER_LOAD_DEFAULT_RC to FALSE to avoid loading it twice.
#
# The purpose of this file is mainly to define keybindings and settings.
# For running more complex python code, please create a plugin in "plugins/" or
# a command in "commands.py".
#
```

Figure 2: The basic configuration file for ranger.

Table 2: Basic ranger Navigation

Up arrow, Page Up	Up a page
Ctrl+U	Up half page
Down arrow	Go down
Ctrl+F, Page Down	Down a page
Ctrl+D	Go down half page
Left arrow	Go left
Right arrow	Go right
Home	Move to top
End	Move to bottom
cd	Change the working directory to a specific directory
gh	Change the working directory to your home
gm	Change the working directory to `/media`
gM	Change the working directory to `/mnt`
gu	Change the working directory to `/usr`
H	Back in history
L	Forward in history
q	Quit

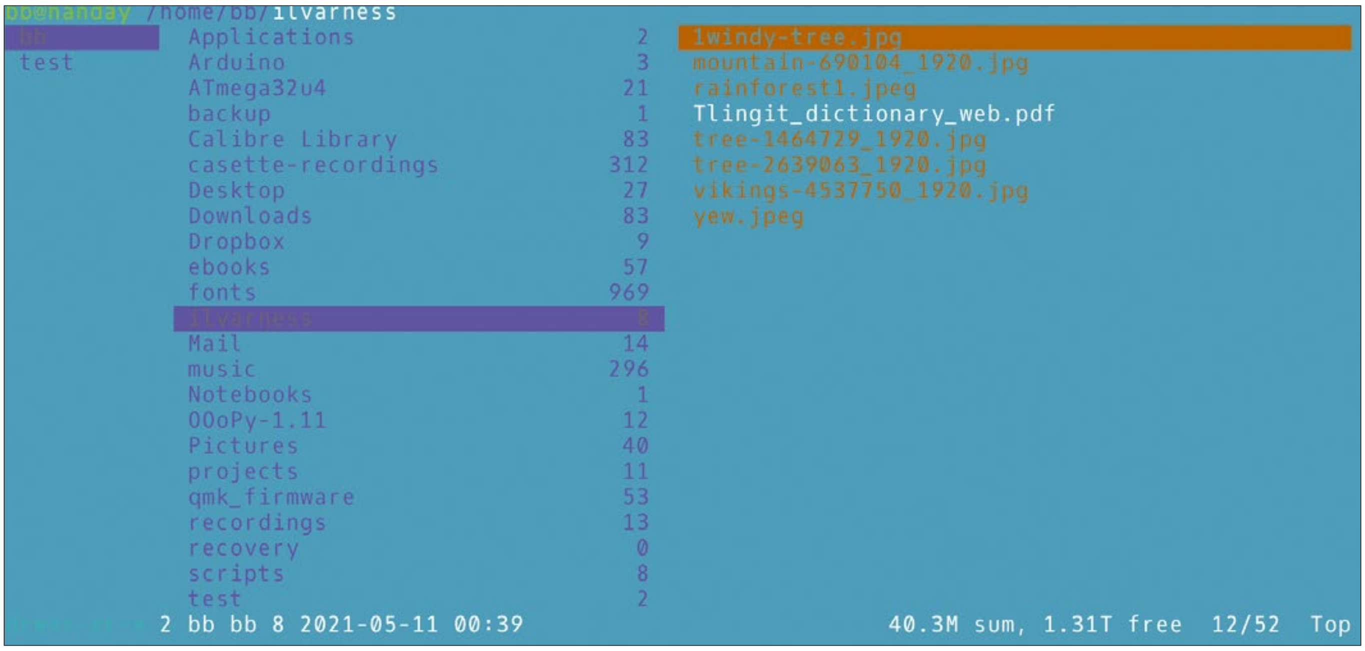
Basic Navigation

If ranger is started as a bare command, it displays your home directory. However, you can also follow the command with a space-separated list of directories and files. A few options are available, but most are unnecessary, especially if you add a list of directories. For many, the only useful option may be `--show-only-dirs`. Each directory or file specified is opened in a tab in the order in which it is listed in the command. The number of directories or files open is listed in the upper right corner. To switch to another tab, press Tab or *gt* to move to the next tab, or Shift + Tab or *gT* to move to the previous tab. The keybinding *gc* will close the current tab, and Ctrl + N or *gn* opens a new tab. In addition, you may want to enter the command:

```
:set show hidden true
```

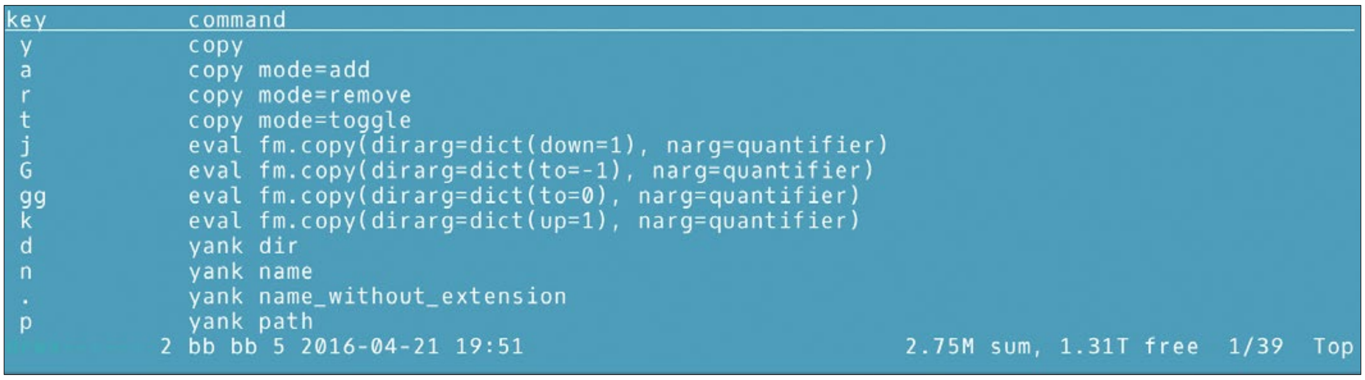
or add it to `~/.config/ranger/rc.conf` to show hidden files as the permanent default.

Each tab has three panes (Figure 3). The left-hand pane shows the top level directories, and the middle one the second level subdirectories. The right-hand pane shows the third-level subdirectories and files, with statistics at the bottom. Types of files are color-coded – for instance, PDF files are white, JPEGs orange, and FLAC files pink. Basic navigation is with arrow keys,



```
bb@handay /home/bb/ilvarness
 bb         Applications               2   1windy-tree.jpg
 test       Arduino                    3   mountain-690104_1920.jpg
            ATmega32u4                21   rainforest1.jpeg
            backup                     1   Tlingit_dictionary_web.pdf
            Calibre Library           83   tree-1464729_1920.jpg
            casette-recordings       312   tree-2639063_1920.jpg
            Desktop                   27   vikings-4537750_1920.jpg
            Downloads                 83   yew.jpeg
            Dropbox                    9
            ebooks                    57
            fonts                    969
            ilvarness                  8
            Mail                      14
            music                    296
            Notebooks                  1
            OOoPy-1.11                12
            Pictures                  40
            projects                  11
            qmk_firmware              53
            recordings                13
            recovery                   0
            scripts                    8
            test                       2
 2 bb bb 8 2021-05-11 00:39                  40.3M sum, 1.31T free  12/52  Top
```

Figure 3: The ranger main screen.



```
key             command
 y              copy
 a              copy mode=add
 r              copy mode=remove
 t              copy mode=toggle
 j              eval fm.copy(dirarg=dict(down=1), narg=quantifier)
 G              eval fm.copy(dirarg=dict(to=-1), narg=quantifier)
 gg             eval fm.copy(dirarg=dict(to=0), narg=quantifier)
 k              eval fm.copy(dirarg=dict(up=1), narg=quantifier)
 d              yank dir
 n              yank name
 .              yank name_without_extension
 p              yank path
              2 bb bb 5 2016-04-21 19:51                          2.75M sum, 1.31T free  1/39  Top
```

Figure 4: Ranger includes built-in cheat sheets for easy use.

with more complex commands assigned to other keys and key combinations (see Table 2). Default key assignments are based on Vim or `mc`, making ranger easy for experienced command line users to learn. An online summary of basic navigation displays when you press the *g* key.

Copying, Moving, and Deleting

To access basic file management, press *y* for a list of copy and paste commands and *d* for move or delete commands (Figure 4). Note that, like Vim, ranger uses the term "yank" instead of paste. Besides the basic commands, the lists also contain advanced options for the basic keystrokes (Table 3). Alternatively, you can press *r* with a file selected or the *s* key and use the standard shell commands instead.

For actions involving more than one file, you can use copy or cut modes. Pressing *a* adds a selected file to the multiple files, while *r* removes the selected file. When you are finished creating a list of files, you can proceed with the operation by pressing *t* to toggle to the copy or cut mode.

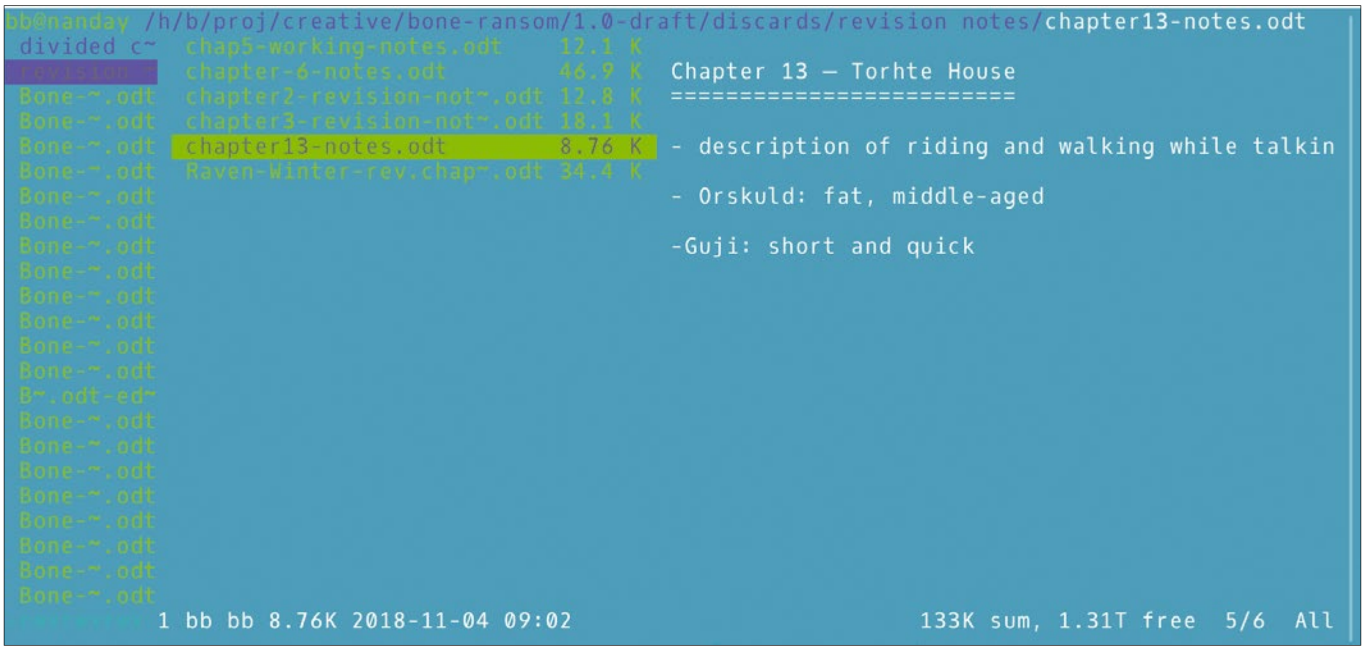
Table 3: Basic File Manipulation

yy	Copy
yp	Paste (yank)
yd	Paste directory (yank)
dd	Cut
dD	Delete (yank)
ya, da	Add selection to a list of files
yr, dr	Remove selection from a list of files
yt, dt	Toggle list of files on or off

To rename or move a selection, press *wa*. For renaming multiple files or directories, you can also use the `:bulkrename` command.

Viewing and Searching

Although ranger is a command-line application, it can open other applications to view formats that include PDF, OpenDocument (LibreOffice), BitTorrent, HTML, and SVG. The supported formats and how they are handled are listed in the `scope.sh` configuration file. You can use the examples in `scope.sh` to add support for other formats. If a file is properly configured in `scope.sh`, all you need to do is select the file and press the Enter key. Some files display in the right-hand pane (Figure 5), and others open in an application in a new window. If a file opens in the right-hand pane, but you would like to view it in its native application, you can use



```
bb@nanday /h/b/proj/creative/bone-ransom/1.0-draft/discards/revision notes/chapter13-notes.odt
 divided c~  chap5-working-notes.odt     12.1 K
 revision ~  chapter-6-notes.odt         46.9 K  Chapter 13 — Torhte House
 Bone-~.odt  chapter2-revision-not~.odt 12.8 K  =========================
 Bone-~.odt  chapter3-revision-not~.odt 18.1 K
 Bone-~.odt  chapter13-notes.odt         8.76 K  - description of riding and walking while talkin
 Bone-~.odt  Raven-Winter-rev.chap~.odt 34.4 K
 Bone-~.odt                                      - Orskuld: fat, middle-aged
 Bone-~.odt
 Bone-~.odt                                      -Guji: short and quick
 Bone-~.odt
 Bone-~.odt
 Bone-~.odt
 Bone-~.odt
 Bone-~.odt
 B~.odt-ed~
 Bone-~.odt
 Bone-~.odt
 Bone-~.odt
 Bone-~.odt
 Bone-~.odt
 Bone-~.odt
 Bone-~.odt
 Bone-~.odt
            1 bb bb 8.76K 2018-11-04 09:02                        133K sum, 1.31T free  5/6  All
```

Figure 5: Ranger views text files in the main window and opens other applications for viewing other file formats.

the `:travel` command. Should a file type not be supported, you can open a command line and run the `rifle` command, which supports a wide variety of formats.

At times, you may just want to view file attributes with `:linemode`. The command can be completed by `filename`, `permissions`, `fileinfo`, `mtime` (last time modified), or `sizemtime` (the size and time of the last modification). You can modify these completions by using them with `:meta`.

The `:travel` command can also be used to quickly find directories. Just enter the name of the directory at the end of the command. Another alternative is to create a bookmark for a selected directory by pressing *m* and assigning a letter or number. A list of existing bookmarks displays as a crib, and you can create the bookmark by pressing *m* a second time. To jump to a bookmark, press ` followed by the bookmark's letter or number.

Within directories, you can use / or *f* to find a file. If the file is not found, you can search for it in the next file by pressing *n*, or in the previous directory by pressing *N*. Visual searches can be enhanced with `:filter REGEX` to reduce the number of files displayed. You can also sort by file size with *os* or arrange sub-directories by size with *du*.

One Last Feature

For those who prefer the command line, ranger has many advantages over `mc`. It has an exhaustive set of commands, many of which have more than one alternative, and integrates well with a desktop environment to view files, in addition to providing convenient cheat sheets that are built into the interface. Also, customization is mostly a matter of adapting samples in the default configuration files. All of these are on top of an often unsung advantage of both `mc` and Vim: Ranger keeps your hands firmly on the keyboard, reducing the repetitive stress of using a mouse. For that reason alone, some users might consider replacing Nautilus or Dolphin with ranger. ■■■

Info

[1] ranger: *https://ranger.github.io/*

[2] Custom commands: *https://github.com/ranger/ranger/wiki/Custom-Commands*

Fast tools for checking disk utilization

Disk Usage

Three modern tools, gdu, godu, and duf, make the task of checking the utilization level of hard disks easier thanks to fast execution speed and a good graphical implementation. *By Ferdinand Thommes*

Hard disks have three states: empty, dead, and full. You can reach a full disk faster than expected due to carelessly hoarded data, much like in a kitchen junk drawer. Error messages logged once a millisecond, such as `.xsession-errors`, can also quickly fill up a logfile causing even large hard disks to overflow overnight.

If your disk is filling up, you first need to identify the cause, specifically which directories or files are hogging the most space. Plasma and Gnome have graphical tools such as Filelight, GdMap, or Baobab, which display the utilization level. However, these tools are very slow, making them unsuitable for network servers or full hard disks, because a graphical user interface will often fail to launch if the disk utilization level exceeds 95 percent.

Instead, you need a terminal-based solution. Linux on-board tools, such as `du` or `df` can show disk utilization, but they are not necessarily renowned for their clarity. For example, you can use the `du` command in Listing 1 to display the 20 largest directories in `home` but in a pretty awkward way. Another option is `ncdu` [1], an Ncurses-based tool available in the archives of all major distributions, which I covered in an earlier issue [2]. While `ncdu` does a fantastic job, it lacks speed during indexing.

If you are looking for speed, three lesser-known tools can do the job: `gdu` [3], `godu` [4], and `duf` [5]. These three tools replace the `ncdu` tool completely or partially and also do their job at lightning speed. All three owe their speed to the Go programming language in which they are written (`ncdu` is written in C). All three tools are optimized for SSD use, which also makes full use of parallel processing. According to the `gdu` developer, the performance gain is lower with HDDs but still noticeable.

Listing 1: Checking with Du

```
$ du -a /home/ | sort -n -r | head -n 20
```

gdu

Currently the most popular, `gdu` can be found in the archives of Debian, Devuan, Manjaro, PureOS, and Raspberry Pi OS. You will also find packages for 32- and 64-bit Linux, ARM, and many BSD variants on `gdu`'s website. Arch Linux users will find `gdu` in the Arch User Repository (AUR).

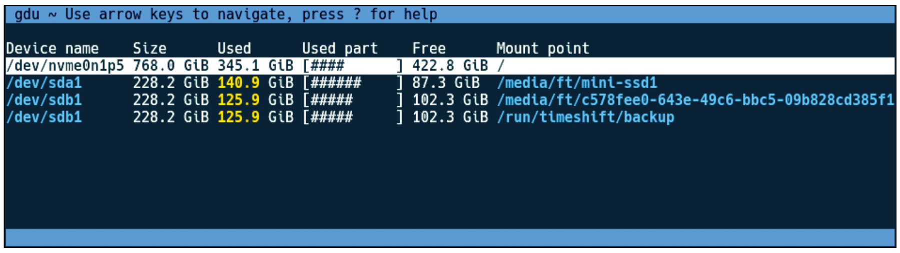
If you use Snap, you can install `gdu` with the command-line call from the first line of Listing 2; the project's GitHub page reveals the further steps. If your distribution does not have `gdu`, the tool can also be installed quickly at the command line (lines 2 to 4).

First launching with `gdu -h` lists the startup parameters. `gdu -d` shows you all the mounted partitions with their sizes and the used and free space (Figure 1). The `-i` parameter excludes directories, and you can view the history log with `gdu -l`.

Normally, `gdu` runs in interactive mode, which can be switched off for use in scripts with `-n`. If you prefer monochrome, `-c` disables the colored display. To navigate, you use the arrow keys in `gdu`, just like in `ncdu`. Pressing Enter or the right arrow key lets you

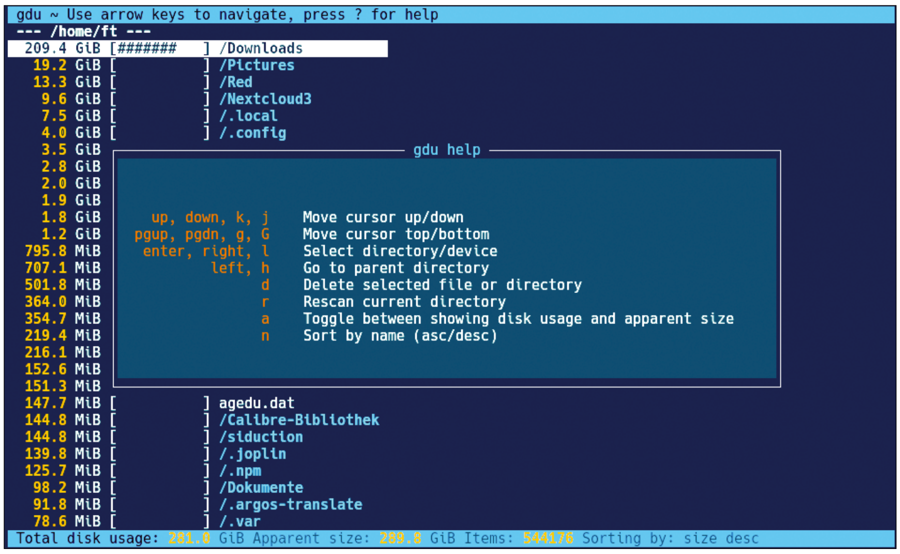
Listing 2: Installing gdu

```
01 $ sudo snap install gdu-disk-usage-analyzer
02 $ curl -L https://github.com/dundee/gdu/releases/latest/download/gdu_linux_amd64.tgz | tar xz
03 $ chmod +x gdu_linux_amd64
04 $ sudo mv gdu_linux_amd64 /usr/bin/gdu
```



```
gdu ~ Use arrow keys to navigate, press ? for help

Device name    Size      Used      Used part     Free      Mount point
/dev/nvme0n1p5 768.0 GiB 345.1 GiB [####      ] 422.8 GiB /
/dev/sda1      228.2 GiB 140.9 GiB [######    ] 87.3 GiB  /media/ft/mini-ssd1
/dev/sdb1      228.2 GiB 125.9 GiB [#####     ] 102.3 GiB /media/ft/c578fee0-643e-49c6-bbc5-09b828cd385f1
/dev/sdb1      228.2 GiB 125.9 GiB [#####     ] 102.3 GiB /run/timeshift/backup
```

Figure 1: The `gdu -d` command gives you an overview of the mounted disks and the partition usage.



```
gdu ~ Use arrow keys to navigate, press ? for help
--- /home/ft ---
 209.4 GiB [#######    ] /Downloads
  19.2 GiB [           ] /Pictures
  13.3 GiB [           ] /Red
   9.6 GiB [           ] /Nextcloud3
   7.5 GiB [           ] /.local
   4.0 GiB [           ] /.config
   3.5 GiB                         gdu help
   2.8 GiB
   2.0 GiB
   1.9 GiB
   1.8 GiB      up, down, k, j    Move cursor up/down
   1.2 GiB    pgup, pgdn, g, G    Move cursor top/bottom
 795.8 MiB     enter, right, l    Select directory/device
 707.1 MiB             left, h    Go to parent directory
 501.8 MiB                   d    Delete selected file or directory
 364.0 MiB                   r    Rescan current directory
 354.7 MiB                   a    Toggle between showing disk usage and apparent size
 219.4 MiB                   n    Sort by name (asc/desc)
 216.1 MiB
 152.6 MiB
 151.3 MiB
 147.7 MiB [           ] agedu.dat
 144.8 MiB [           ] /Calibre-Bibliothek
 144.8 MiB [           ] /siduction
 139.8 MiB [           ] /.joplin
 125.7 MiB [           ] /.npm
  98.2 MiB [           ] /Dokumente
  91.8 MiB [           ] /.argos-translate
  78.6 MiB [           ] /.var
Total disk usage: 281.0 GiB Apparent size: 289.8 GiB Items: 544176 Sorting by: size desc
```

Figure 2: Pressing ? displays a help window.

change to the selected directory, while the left arrow key moves you up one directory level again. Pressing *Q* returns to the terminal.

If you press *?*, `gdu` lists the available options (Figure 2). Table 1 shows more keyboard shortcuts. If you call `gdu` without parameters, it shows the contents of the current working directory. By specifying a path, you can steer `gdu` precisely to a location in the filesystem without being in that directory (Figure 3).

The only destructive parameter, *D*, deletes the selected file or directory. As an alternative to the arrow keys, use *K* to move up and *J* to move down. *L* moves you to the selected directory.

Overall, `gdu` leans heavily on `ncdu` in both appearance and operation, though `ncdu` has more options. The biggest difference, however, is in speed. `ncdu` took 7.4 seconds to scan a home directory with around 280GB of content on an NVMe disk in the test, while `gdu` completed the same task within milliseconds – so quickly that the duration could not be measured with a stopwatch. It would be interesting to compare the two tools on an HDD, but there was no HDD available for the test.

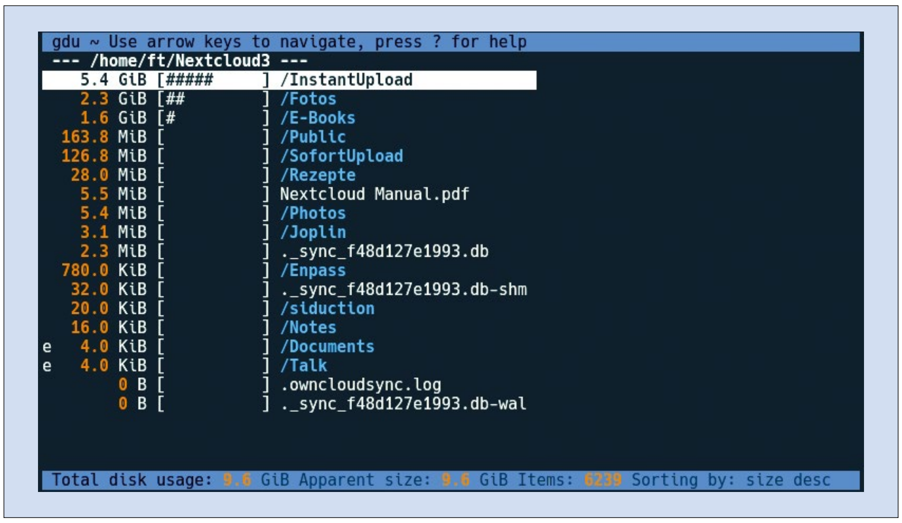
godu

Pursuing the same goal, `godu` takes a slightly different approach. The difference lies in the representation: `godu` uses Miller columns [6] to allow for more flexible visualization (Figure 4). The columns let you open multiple hierarchy levels at the same time and provide a better overview of where you are in the directory tree.

The easiest way to install `godu` on your machine is to download the appropriate archive containing the binary [7], which you then unzip and move to `~/.local/bin/`. If this directory does not already exist, create it, and add it to the path by adding the command in Listing 3 to `~/.bashrc`. A subsequent `source ~/.bashrc` activates the change. If you want all users to use `godu`, store it in `/usr/local/bin/`. Other installation methods include Homebrew or a Golang environment. However, the latter can be tricky for nonexpert users.

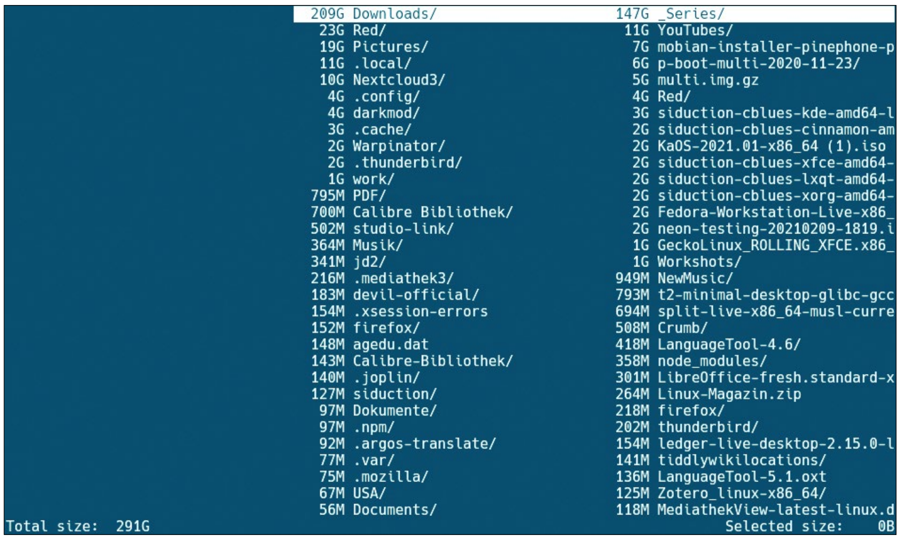
Table 1: gdu Keyboard Shortcuts

Key	Function
Up arrow	Move up in the output
Down arrow	Move down in the output
Right arrow	Change to selected directory
Left arrow	Move one directory level up
Enter	Change to selected directory
`A`	Change actual size/occupied space
`C`	Sort by number of contents
`D`	Delete selected file
`J`	Move down in the output
`K`	Move up in the output
`L`	Change to selected directory
`N`	Sort by file name
`Q`	Quit gdu
`R`	Recompute selected element
`S`	Sort by file size



```
gdu ~ Use arrow keys to navigate, press ? for help
--- /home/ft/Nextcloud3 ---
    5.4 GiB [#####      ] /InstantUpload
    2.3 GiB [##         ] /Fotos
    1.6 GiB [#          ] /E-Books
  163.8 MiB [           ] /Public
  126.8 MiB [           ] /SofortUpload
   28.0 MiB [           ] /Rezepte
    5.5 MiB [           ] Nextcloud Manual.pdf
    5.4 MiB [           ] /Photos
    3.1 MiB [           ] /Joplin
    2.3 MiB [           ] ._sync_f48d127e1993.db
  780.0 KiB [           ] /Enpass
   32.0 KiB [           ] ._sync_f48d127e1993.db-shm
   20.0 KiB [           ] /siduction
   16.0 KiB [           ] /Notes
e   4.0 KiB [           ] /Documents
e   4.0 KiB [           ] /Talk
        0 B [           ] .owncloudsync.log
        0 B [           ] ._sync_f48d127e1993.db-wal


Total disk usage: 9.6 GiB Apparent size: 9.6 GiB Items: 6239 Sorting by: size desc
```

Figure 3: After specifying the path, `gdu` shows you the contents for a specific directory.



```
                               209G Downloads/                   147G _Series/
                                23G Red/                          11G YouTubes/
                                19G Pictures/                      7G mobian-installer-pinephone-p
                                11G .local/                        6G p-boot-multi-2020-11-23/
                                10G Nextcloud3/                    5G multi.img.gz
                                 4G .config/                       4G Red/
                                 4G darkmod/                       3G siduction-cblues-kde-amd64-l
                                 3G .cache/                        2G siduction-cblues-cinnamon-am
                                 2G Warpinator/                    2G KaOS-2021.01-x86_64 (1).iso
                                 2G .thunderbird/                  2G siduction-cblues-xfce-amd64-
                                 1G work/                          2G siduction-cblues-lxqt-amd64-
                               795M PDF/                           2G siduction-cblues-xorg-amd64-
                               700M Calibre Bibliothek/            2G Fedora-Workstation-Live-x86_
                               502M studio-link/                   2G neon-testing-20210209-1819.i
                               364M Musik/                         1G GeckoLinux_ROLLING_XFCE.x86_
                               341M jd2/                           1G Workshots/
                               216M .mediathek3/                 949M NewMusic/
                               183M devil-official/              793M t2-minimal-desktop-glibc-gcc
                               154M .xsession-errors             694M split-live-x86_64-musl-curre
                               152M firefox/                     508M Crumb/
                               148M agedu.dat                    418M LanguageTool-4.6/
                               143M Calibre-Bibliothek/          358M node_modules/
                               140M .joplin/                     301M LibreOffice-fresh.standard-x
                               127M siduction/                   264M Linux-Magazin.zip
                                97M Dokumente/                   218M firefox/
                                97M .npm/                        202M thunderbird/
                                92M .argos-translate/            154M ledger-live-desktop-2.15.0-l
                                77M .var/                        141M tiddlywikilocations/
                                75M .mozilla/                    136M LanguageTool-5.1.oxt
                                67M USA/                         125M Zotero_linux-x86_64/
                                56M Documents/                   118M MediathekView-latest-linux.d
Total size:  291G                                                       Selected size:     0B
```

Figure 4: **To accommodate as much content as possible in the terminal, `godu` uses Miller columns to display the content.**

Typing `godu -h` reveals that `godu` provides only one parameter apart from the help and the version information: `godu -l` lets you exclude files if they are below a specified size. For example,

```
godu -l 100 /
```

scans the entire file tree but only considers files larger than 100MB. `gdu` lacks this convenient option. If you do not specify anything else, `godu` uses 10MB as the limit for `-l`.

If you want to exclude certain directories from the display, enter them in the `.goduignore` file in the home directory, which you need to create for this purpose. This can also speed up content scanning, especially if you exclude directories with a large number of small files.

Navigation in `godu` also relies on the arrow keys. Up to three levels can be opened side by side. If you move further down the directory tree, the level shown on the left disappears and makes room for a new one on the right (Figure 5). Pressing the space bar lets you select several directories or files for deletion or moving.

To delete or move the selected entries, exit `godu` by pressing *Q* and use

```
godu -print0 | xargs -0 COMMAND
```

where `COMMAND` is the appropriate command for the desired action. For example, to move previously selected entries, use the command shown in line 1 of Listing 4. To simplify the process, you can create an appropriate alias in `.bashrc` (line 2).

`godu` provides a better overview by outputting all previously selected objects for checking in the standard output after exiting. It is worth looking through this list carefully before you let `xargs` loose on it.

Listing 3: Supplementing the Path

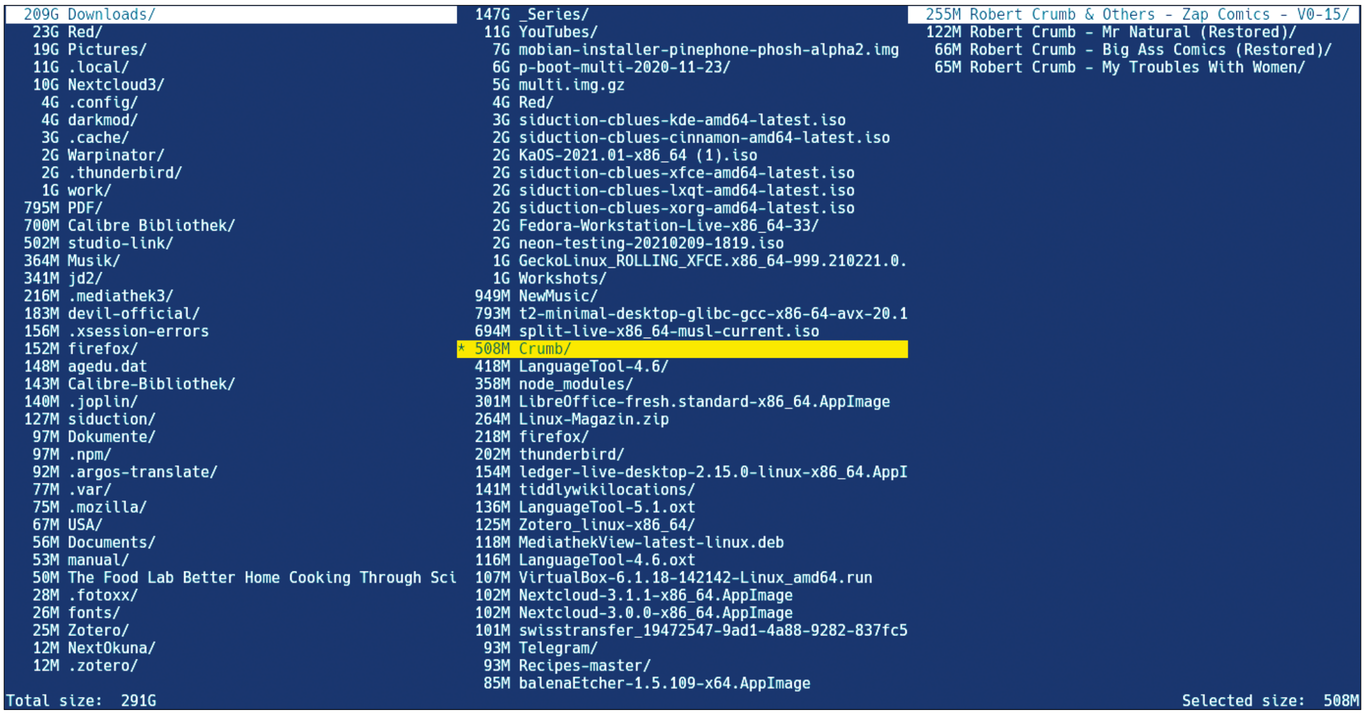
```
export PATH="/home/$USER/.local/bin:$PATH"
```

Listing 4: Moving Selected Entries

```
01 $ godu -print0 | xargs -0 -I _ mv _ <I>Ziel<I>

02 $ alias gmv="godu -print0 | xargs -0 -I _ mv _ "
```

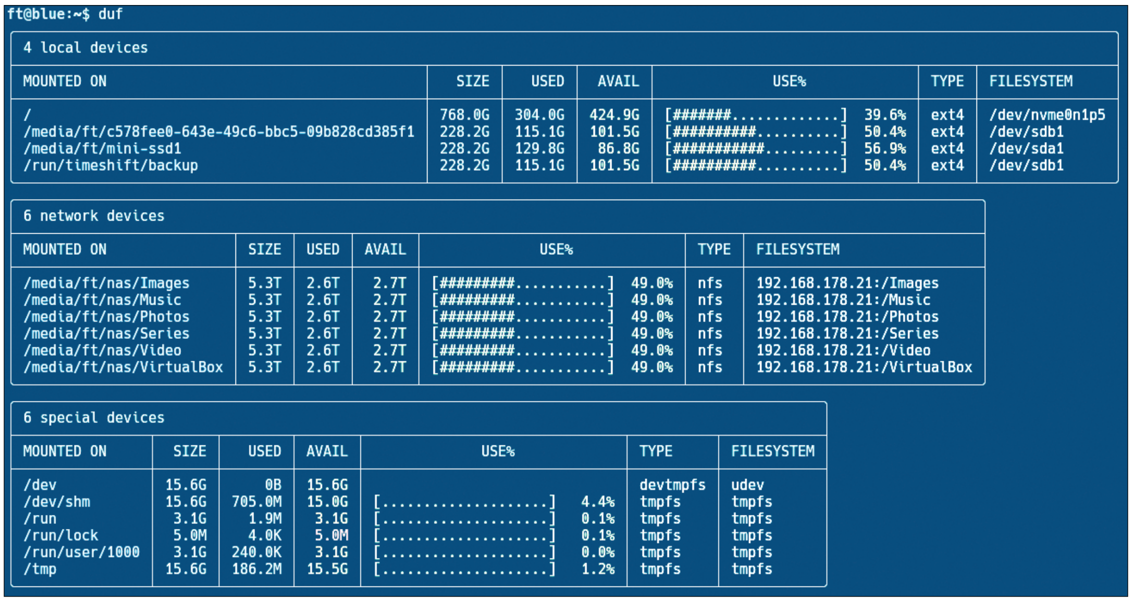
duf

The third tool, `duf`, stands for Disk Usage/Free. It differs from `gdu` and `godu`



```
  209G Downloads/                                  147G _Series/                                255M Robert Crumb & Others - Zap Comics - V0-15/
   23G Red/                                         11G YouTubes/                                 122M Robert Crumb - Mr Natural (Restored)/
   19G Pictures/                                     7G mobian-installer-pinephone-phosh-alpha2.img  66M Robert Crumb - Big Ass Comics (Restored)/
   11G .local/                                       6G p-boot-multi-2020-11-23/                  65M Robert Crumb - My Troubles With Women/
   10G Nextcloud3/                                   5G multi.img.gz
    4G .config/                                      4G Red/
    4G darkmod/                                      3G siduction-cblues-kde-amd64-latest.iso
    3G .cache/                                       2G siduction-cblues-cinnamon-amd64-latest.iso
    2G Warpinator/                                   2G KaOS-2021.01-x86_64 (1).iso
    2G .thunderbird/                                 2G siduction-cblues-xfce-amd64-latest.iso
    1G work/                                         2G siduction-cblues-lxqt-amd64-latest.iso
  795M PDF/                                          2G siduction-cblues-xorg-amd64-latest.iso
  700M Calibre Bibliothek/                           2G Fedora-Workstation-Live-x86_64-33/
  502M studio-link/                                  2G neon-testing-20210209-1819.iso
  364M Musik/                                        1G GeckoLinux_ROLLING_XFCE.x86_64-999.210221.0.
  341M jd2/                                          1G Workshots/
  216M .mediathek3/                                949M NewMusic/
  183M devil-official/                             793M t2-minimal-desktop-glibc-gcc-x86-64-avx-20.1
  156M .xsession-errors                            694M split-live-x86_64-musl-current.iso
  152M firefox/                                  * 508M Crumb/
  148M agedu.dat                                   418M LanguageTool-4.6/
  143M Calibre-Bibliothek/                         358M node_modules/
  140M .joplin/                                    301M LibreOffice-fresh.standard-x86_64.AppImage
  127M siduction/                                  264M Linux-Magazin.zip
   97M Dokumente/                                  218M firefox/
   97M .npm/                                       202M thunderbird/
   92M .argos-translate/                           154M ledger-live-desktop-2.15.0-linux-x86_64.AppI
   77M .var/                                       141M tiddlywikilocations/
   75M .mozilla/                                   136M LanguageTool-5.1.oxt
   67M USA/                                        125M Zotero_linux-x86_64/
   56M Documents/                                  118M MediathekView-latest-linux.deb
   53M manual/                                     116M LanguageTool-4.6.oxt
   50M The Food Lab Better Home Cooking Through Sci 107M VirtualBox-6.1.18-142142-Linux_amd64.run
   28M .fotoxx/                                    102M Nextcloud-3.1.1-x86_64.AppImage
   26M fonts/                                      102M Nextcloud-3.0.0-x86_64.AppImage
   25M Zotero/                                     101M swisstransfer_19472547-9ad1-4a88-9282-837fc5
   12M NextOkuna/                                   93M Telegram/
   12M .zotero/                                     93M Recipes-master/
                                                    85M balenaEtcher-1.5.109-x64.AppImage
Total size:  291G                                                                                              Selected size:  508M
```

Figure 5: **If you click on a directory in the right column above, it opens to the right, while `godu` hides the left column to make the necessary space.**



```
ft@blue:~$ duf
╭──────────────────────────────────────────────────────────────────────────────────────────────────────────────────────╮
│ 4 local devices                                                                                                      │
├──────────────────────────────────────────────┬────────┬────────┬────────┬─────────────────────────────┬──────┬────────────────┤
│ MOUNTED ON                                   │   SIZE │   USED │  AVAIL │            USE%             │ TYPE │ FILESYSTEM     │
├──────────────────────────────────────────────┼────────┼────────┼────────┼─────────────────────────────┼──────┼────────────────┤
│ /                                            │ 768.0G │ 304.0G │ 424.9G │ [#######.............]  39.6% │ ext4 │ /dev/nvme0n1p5 │
│ /media/ft/c578fee0-643e-49c6-bbc5-09b828cd385f1 │ 228.2G │ 115.1G │ 101.5G │ [##########..........]  50.4% │ ext4 │ /dev/sdb1      │
│ /media/ft/mini-ssd1                          │ 228.2G │ 129.8G │  86.8G │ [###########.........]  56.9% │ ext4 │ /dev/sda1      │
│ /run/timeshift/backup                        │ 228.2G │ 115.1G │ 101.5G │ [##########..........]  50.4% │ ext4 │ /dev/sdb1      │
╰──────────────────────────────────────────────┴────────┴────────┴────────┴─────────────────────────────┴──────┴────────────────╯
╭──────────────────────────────────────────────────────────────────────────────────────────────────────────╮
│ 6 network devices                                                                                        │
├─────────────────────────┬──────┬──────┬───────┬──────────────────────────────┬──────┬─────────────────────────────┤
│ MOUNTED ON              │ SIZE │ USED │ AVAIL │             USE%             │ TYPE │ FILESYSTEM                  │
├─────────────────────────┼──────┼──────┼───────┼──────────────────────────────┼──────┼─────────────────────────────┤
│ /media/ft/nas/Images    │ 5.3T │ 2.6T │  2.7T │ [#########...........]  49.0% │ nfs  │ 192.168.178.21:/Images      │
│ /media/ft/nas/Music     │ 5.3T │ 2.6T │  2.7T │ [#########...........]  49.0% │ nfs  │ 192.168.178.21:/Music       │
│ /media/ft/nas/Photos    │ 5.3T │ 2.6T │  2.7T │ [#########...........]  49.0% │ nfs  │ 192.168.178.21:/Photos      │
│ /media/ft/nas/Series    │ 5.3T │ 2.6T │  2.7T │ [#########...........]  49.0% │ nfs  │ 192.168.178.21:/Series      │
│ /media/ft/nas/Video     │ 5.3T │ 2.6T │  2.7T │ [#########...........]  49.0% │ nfs  │ 192.168.178.21:/Video       │
│ /media/ft/nas/VirtualBox│ 5.3T │ 2.6T │  2.7T │ [#########...........]  49.0% │ nfs  │ 192.168.178.21:/VirtualBox  │
╰─────────────────────────┴──────┴──────┴───────┴──────────────────────────────┴──────┴─────────────────────────────╯
╭────────────────────────────────────────────────────────────────────────────────────────────╮
│ 6 special devices                                                                          │
├────────────────┬───────┬────────┬───────┬───────────────────────────────┬──────────┬────────────┤
│ MOUNTED ON     │  SIZE │   USED │ AVAIL │             USE%              │ TYPE     │ FILESYSTEM │
├────────────────┼───────┼────────┼───────┼───────────────────────────────┼──────────┼────────────┤
│ /dev           │ 15.6G │     0B │ 15.6G │                               │ devtmpfs │ udev       │
│ /dev/shm       │ 15.6G │ 705.0M │ 15.0G │ [....................]   4.4% │ tmpfs    │ tmpfs      │
│ /run           │  3.1G │   1.9M │  3.1G │ [....................]   0.1% │ tmpfs    │ tmpfs      │
│ /run/lock      │  5.0M │   4.0K │  5.0M │ [....................]   0.1% │ tmpfs    │ tmpfs      │
│ /run/user/1000 │  3.1G │ 240.0K │  3.1G │ [....................]   0.0% │ tmpfs    │ tmpfs      │
│ /tmp           │ 15.6G │ 186.2M │ 15.5G │ [....................]   1.2% │ tmpfs    │ tmpfs      │
╰────────────────┴───────┴────────┴───────┴───────────────────────────────┴──────────┴────────────╯
```

Figure 6: `duf` **provides the best overview of all mounted filesystems and shows virtual filesystems in addition to local partitions and network devices.**

both in terms of operation and how it displays the results. `duf` is available for Alpine, NixOS, and Void Linux in their respective repositories; on Arch Linux, you will find `duf` in the AUR. The developer also offers a variety of binaries on GitHub [8] for various Linux and BSD distributions, macOS, Windows, and ARM. `duf` also runs on Android. To try this, enter the command

```
pkg install duf
```

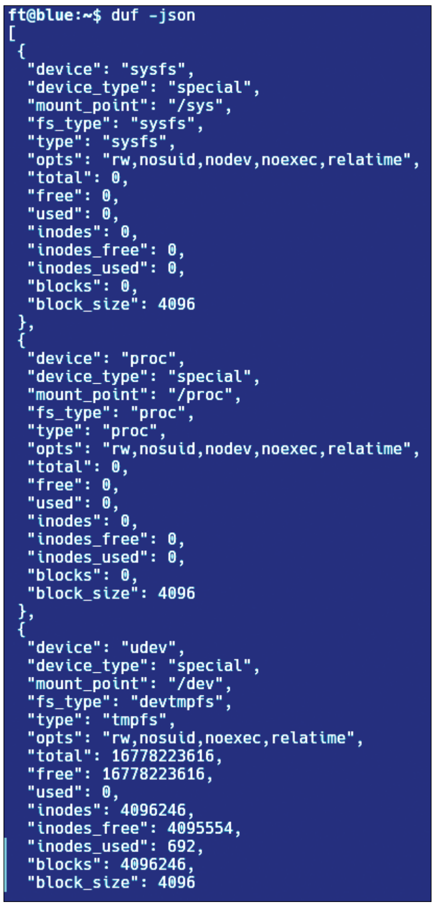
in Android's Termux app.

At first glance, entering `duf` without any further parameters results in impressive output. Of the three tools, `duf` offers the clearest and most detailed view of the mounted devices, distinguishing between local and network devices. As a third category, `duf` displays directories mounted in memory via `tmpfs`. The `-all` parameter expands the display again to include mounted FUSE devices, as well as the pseudo filesystems under `/proc` and `/sys` (Figure 6).

Entering `duf -h` shows more than a dozen other ways to shape the output. `duf -hide AREA` lets you hide individual areas, while `duf -only AREA` only shows certain areas. The default dark display can be changed to a light theme with `duf -theme light`.

You do not navigate with `duf`; you just define the mount point view via the options at startup. The output is sorted using the `--sort` and `--output` parameters, to which you can assign keywords that you will find in the help if needed. For example, `duf --sort size` sorts the mount points by size, while `duf --output mountpoint` simply lists the mount points without any other information.

`duf -json` gives you output that can be further processed with other applications (Figure 7). However, data manipulation is left to `gdu` and `godu`; `duf` does not offer an interactive mode.



```
ft@blue:~$ duf -json
[
 {
  "device": "sysfs",
  "device_type": "special",
  "mount_point": "/sys",
  "fs_type": "sysfs",
  "type": "sysfs",
  "opts": "rw,nosuid,nodev,noexec,relatime",
  "total": 0,
  "free": 0,
  "used": 0,
  "inodes": 0,
  "inodes_free": 0,
  "inodes_used": 0,
  "blocks": 0,
  "block_size": 4096
 },
 {
  "device": "proc",
  "device_type": "special",
  "mount_point": "/proc",
  "fs_type": "proc",
  "type": "proc",
  "opts": "rw,nosuid,nodev,noexec,relatime",
  "total": 0,
  "free": 0,
  "used": 0,
  "inodes": 0,
  "inodes_free": 0,
  "inodes_used": 0,
  "blocks": 0,
  "block_size": 4096
 },
 {
  "device": "udev",
  "device_type": "special",
  "mount_point": "/dev",
  "fs_type": "devtmpfs",
  "type": "tmpfs",
  "opts": "rw,nosuid,noexec,relatime",
  "total": 16778223616,
  "free": 16778223616,
  "used": 0,
  "inodes": 4096246,
  "inodes_free": 4095554,
  "inodes_used": 692,
  "blocks": 4096246,
  "block_size": 4096
```

Figure 7: **The** `duf -json` **command prepares the requested data in JSON format allowing for downstream processing with other tools.**

Conclusions

Even if you already use `ncdu`, `gdu` is well worth a look. `godu` is as fast as `gdu` and offers a broader overview. `godu` will be your favorite if you want to exclude directories and files or only read files above a certain size. In addition, `godu` lets you delete previously selected objects.

If you only want to see all mounted devices and their sizes, `duf` offers the best overview. In all other aspects, however, its competitors impress with greater flexibility. Having said this, all three candidates offer more than the standard tools `du` and `df` in one respect or another. ■■■

Info

[1] ncdu: *https://dev.yorhel.nl/ncdu*

[2] "Weighing in with ncdu" by Ferdinand Thommes, *Linux Magazine*, issue 214, September 2018: *https://www.linux-magazine.com/Issues/2018/214/Ncdu*

[3] gdu: *https://github.com/dundee/gdu*

[4] godu: *https://github.com/viktomas/godu*

[5] duf: *https://github.com/muesli/duf*

[6] Miller columns: *https://en.wikipedia.org/wiki/Miller_columns*

[7] godu download: *https://github.com/viktomas/godu/releases/latest*

[8] duf download: *https://github.com/muesli/duf/releases*

Author

Ferdinand Thommes lives and works as a Linux developer, freelance writer, and tour guide in Berlin.

The sys admin's daily grind: Customizing Vim

Vim Primping

Working with the infamous Vim is part of every sys admin's daily work. Charly spices up the veteran editor with personal settings for syntax highlighting and indentations. *By Charly Kühnast*

The Vim text editor accompanies every sys admin throughout their entire professional life, just like the classic corny joke about it: "I've been using Vim for 15 years – but mainly because I have no idea how to quit it." It takes a few years for some admins to grudgingly resign themselves to the text editor. Others claim to really like Vim, but that could be Stockholm syndrome. Either way, the basics have to be drilled in because Vim is one of the constants you can and must rely on with any Linux system.

Beyond the basics, Vim offers a few functions that enhance the user experience. There are two ways to customize the editor: globally or per user. The global settings are located in the `/etc/vim/vimrc` file; each user can also create a file named `.vimrc` in their home directory. Settings made there overwrite and supplement the global settings. Many distributions do not specify global settings from the outset (i.e., `/etc/vim/vimrc` is empty), knowing that adjustments are always a matter of taste.

Let's look at the most important customizations that can be made as a user in `.vimrc`. The most popular of these relates to syntax highlighting. When programming, Vim displays certain elements such as variables, mathematical characters, loop entry and exit points, comments, and much more in different colors. The `syntax on` line in `.vimrc` enables highlighting.

My personal favorite function: If I move the cursor to a closing bracket, Vim highlights the corresponding opening bracket – if it's the wrong bracket, I must have forgotten one somewhere. You can enable the function by adding `set showmatch` to your `.vimrc` file.

Two other entries useful for programming are shown in Listing 1. The first tells Vim to automatically continue indentations. If you have indented a line by four characters, this indentation will be applied the next time you press the Enter key.

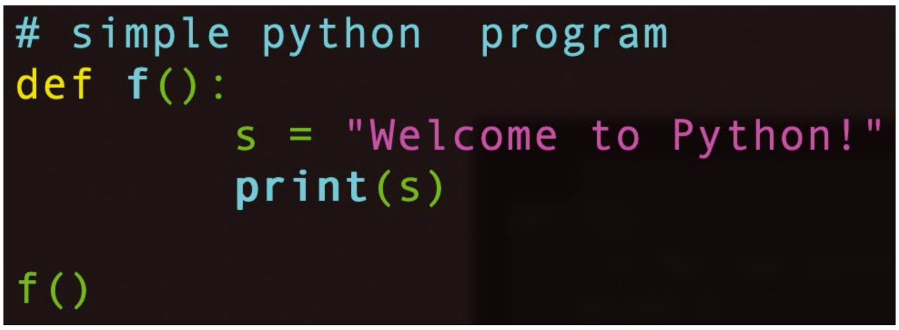
Listing 1: Useful Functions

```
set autoindent
set smartindent
```

The second parameter intelligently tries to guess what you are programming and continues indentations just like `autoindent`, but it automatically terminates them at the end of a loop construction or if-then-else block. `smartindent` is optimized for C programming but also works quite well with many other (scripting) languages, such as Python. Figure 1 shows a very short program in Python with syntax highlighting and "smart" indentations.

If the indentations steal too many characters, which can quickly happen in deeply nested loops, you can shorten them. To indent only three characters deep, add `set shiftwidth=3` to your `.vimrc` file.

These customizations are built into Vim, but they are not typically active by default. The editor also offers the possibility of including plugins that retrofit many useful functions, such as a language-specific spellchecker and much more. ■■■



```
# simple python  program
def f():
        s = "Welcome to Python!"
        print(s)

f()
```

Figure 1: Smart indentations with Vim.

Author

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

Remove unnecessary files with Czkawka

File Diet

Czkawka helps find and remove duplicate and obsolete files to free up valuable disk space. By Erik Bärwaldt

Today's operating systems typically consist of well over 100,000 files, and more files are quickly added during daily work. Many of these added files are temporary files or logfiles. Over time, you might also accumulate duplicate files, configuration files for deleted programs, and the remnants of broken files. Resourceful programmers have developed tools to remove these unneeded files quickly and efficiently. One cleanup tool that is gaining attention with Linux users is Czkawka [1]. According to the project developers, Czkawka will help you discover and delete:

- *Duplicate Files*: Finds duplicates based on file name, size, hash, and a hash of just first 1MB of a file
- *Empty Folders*: Finds empty folders with the help of an advanced algorithm
- *Big Files*: Finds the biggest files in given location
- *Empty Files*: Looks for empty files across the drive
- *Temporary Files*: Finds temporary files
- *Similar Images*: Finds images that are not exactly the same (different resolution, watermarks)
- *Zeroed Files*: Finds files that are filled with zeros (usually corrupted)
- *Same Music*: Searches for music with the same artist, album, etc.
- *Invalid Symlinks* – Shows symbolic links that point to nonexistent files/directories
- *Broken Files*: Finds files with an invalid extension or that are corrupted

Czkawka, which gets its name from the Polish word for hiccup, is available in a GUI variant and as a command-line tool. The GUI variant is based on the GTK libraries.

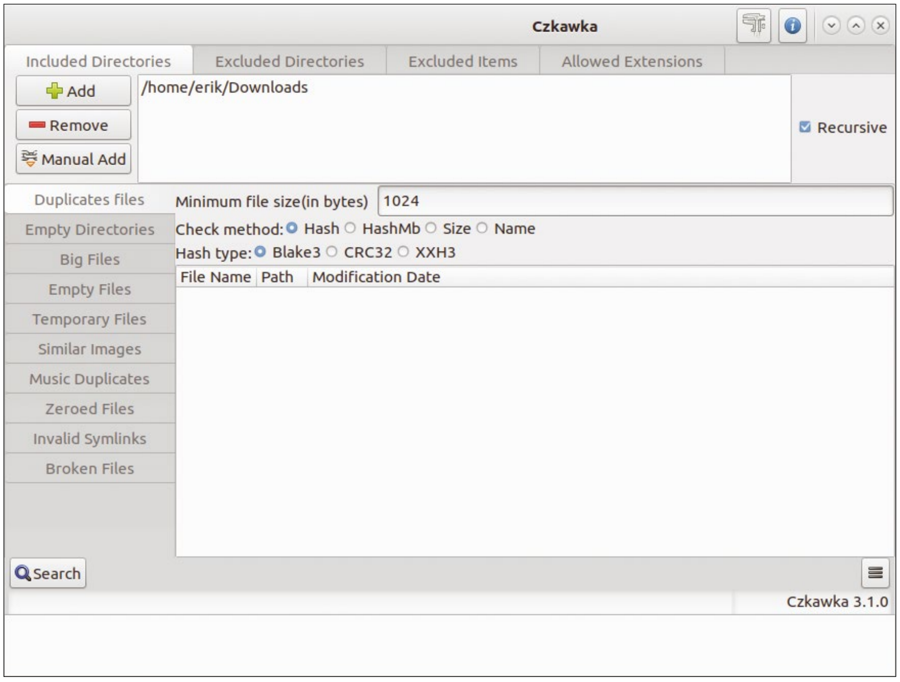
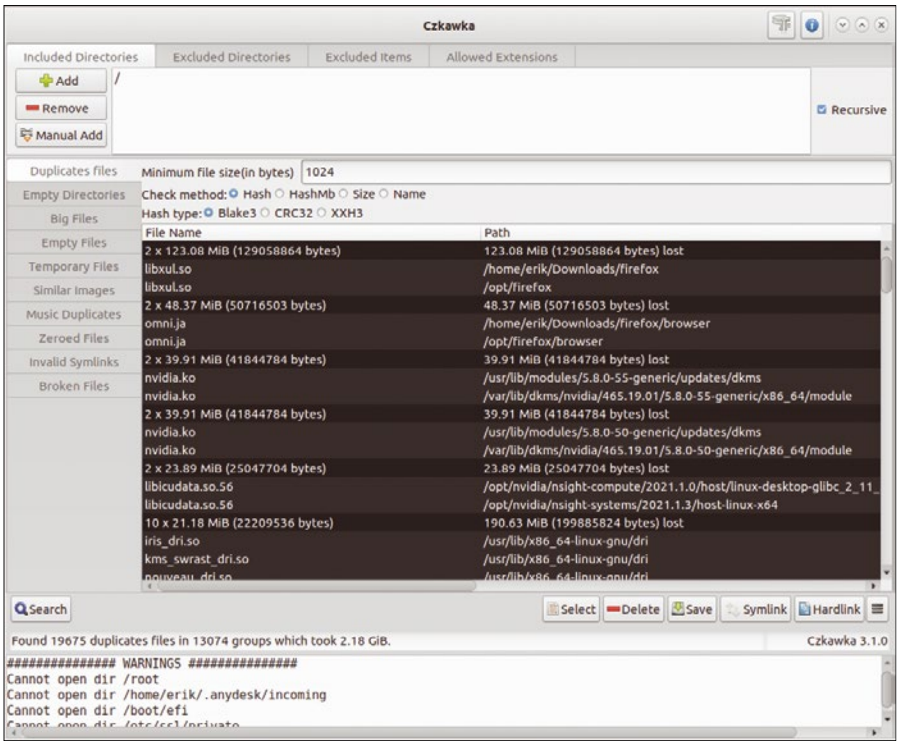


Figure 1: The interface of Czkawka seems a bit confusing at first sight, but the GUI does not pose an obstacle, even for newcomers.



/

☑ Recursive

Minimum file size(in bytes) 1024

Check method: [✓ Hash] HashMb Size Name

Hash type: [✓ Blake3] CRC32 XXH3

File Name	Path
2 x 123.08 MiB (129058864 bytes)	123.08 MiB (129058864 bytes) lost
libxul.so	/home/erik/Downloads/firefox
libxul.so	/opt/firefox
2 x 48.37 MiB (50716503 bytes)	48.37 MiB (50716503 bytes) lost
omni.ja	/home/erik/Downloads/firefox/browser
omni.ja	/opt/firefox/browser
2 x 39.91 MiB (41844784 bytes)	39.91 MiB (41844784 bytes) lost
nvidia.ko	/usr/lib/modules/5.8.0-55-generic/updates/dkms
nvidia.ko	/var/lib/dkms/nvidia/465.19.01/5.8.0-55-generic/x86_64/module
2 x 39.91 MiB (41844784 bytes)	39.91 MiB (41844784 bytes) lost
nvidia.ko	/usr/lib/modules/5.8.0-50-generic/updates/dkms
nvidia.ko	/var/lib/dkms/nvidia/465.19.01/5.8.0-50-generic/x86_64/module
2 x 23.89 MiB (25047704 bytes)	23.89 MiB (25047704 bytes) lost
libicudata.so.56	/opt/nvidia/nsight-compute/2021.1.0/host/linux-desktop-glibc_2_11_
libicudata.so.56	/opt/nvidia/nsight-systems/2021.1.3/host-linux-x64
10 x 21.18 MiB (22209536 bytes)	190.63 MiB (199885824 bytes) lost
iris_dri.so	/usr/lib/x86_64-linux-gnu/dri
kms_swrast_dri.so	/usr/lib/x86_64-linux-gnu/dri
nouveau_dri.so	/usr/lib/x86_64-linux-gnu/dri

Found 19675 duplicates files in 13074 groups which took 2.18 GiB.   Czkawka 3.1.0

```
############### WARNINGS ###############
Cannot open dir /root
Cannot open dir /home/erik/.anydesk/incoming
Cannot open dir /boot/efi
Cannot open dir /etc/ssl/private
```

Figure 2: The software displays any duplicates it finds grouped vertically.

Installation

At this writing, Czkawka is only available in the repositories of OpenMandriva Cooker; for all other distributions, you need to install manually. A Snap package allows for quick integration into distributions that support Snap; Czkawka is also available as a Flathub package and an AppImage.

On the project's GitHub page, you will also find two generic packages, the source code, a Docker image, and installation instructions for various distributions. The project also maintains an unofficial PPA for Ubuntu and Debian-based distributions [2].

Interface

When you first start Czkawka, a somewhat confusing program window opens. The complexity of the GUI is your first indication of the large feature set and massive number of options (Figure 1).

The parameters are organized in tabs. After the launch, you'll see the *Included Directories* box at the top of the dialog. Select the directories you want to clean up. You can add more folders by pressing the *Add* button.

By default, the software lists the current directory of the logged-in user. To the right, you can define whether the tool searches for files in the specified directories recursively (i.e., including files found in subdirectories within the directory tree). *Manual Add* lets you specify a path for a directory without the detour via the file manager.

Click the *Excluded Directories* tab to define directories you want to exclude from search and delete operations. The exclusion list already contains numerous system folders. You can manage this list with the buttons to the left.

The *Excluded Items* tab lets you specify criteria for excluding files that you want the program to ignore. Entries are mostly recursive; asterisks serve as placeholders.

In the *Allowed Extensions* tab, enter file extensions that you want to include in the search. Separate multiple extensions by commas. In addition, the software lets you define entire file types, such as *IMAGE* or *VIDEO*, using an integrated macro.

Further down, you can enter additional criteria. Czkawka presents several tabs with options in a vertical arrangement on the left margin. For example, you can search for duplicates by clicking *Duplicate files* on the left. Set the minimum size of the duplicates and the check method. You can choose to list matching files by hash values, file names, or file size. However, the program only lets you select one option; the combination of *Size* and *Name*, for example, does not work.

The application searches the specified folders after you click *Search* in the lower-left corner. Note that if you enter the root directory (Figure 2), the software will search the entire partition and the check could take a long time, depending on the speed of the disk.

In the list of duplicates, you will see the file size and path. The tool also gives the disk space occupied by the duplicate versions. At the bottom of the window, you can see a table with warnings that the program logs. For example, if you enable all folders to search for duplicates, but you are operating under user privileges, the log will tell you that the tool cannot open system directories. To hide and show the log, click the hamburger menu at the bottom right of the program window.

Use the buttons below the table to determine what the software does with the files that match the criteria. Select an entry to open a context menu with different options.

You can delete the items by pressing *Delete* (Figure 3). It is also possible to save the results from the table as a text file using the *Save* button.

General

To access general configuration settings, click on the button with a symbolized caliper at the top-right of the program window. You can now define how the software handles changed options,

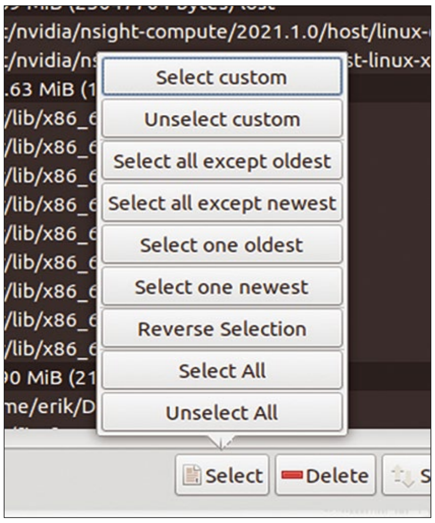


Figure 3: Czkawka offers very detailed selection functions.

whether it moves the files to be deleted to the recycle bin instead of deleting them directly, and whether it displays reduced images of the contents of image files.

Other Options

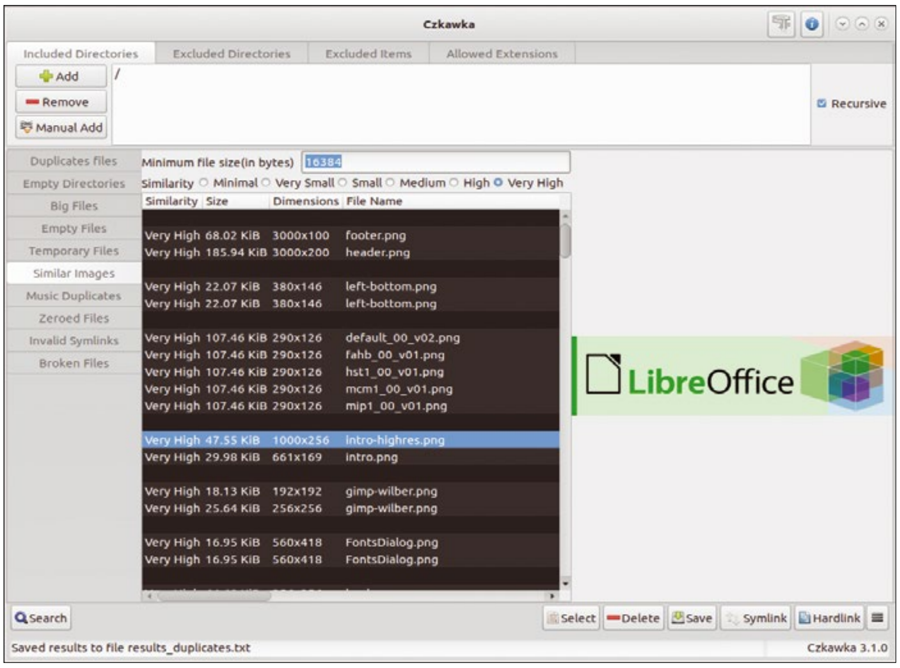
Czkawka also lets you delete *Empty Directories*, *Big Files*, *Empty Files*, and *Temporary Files*. You can even search multimedia content to find matches. In the *Similar Images* tab, search for image files. You'll need to specify the minimum size and also determine the degree of matching.

Once the entries appear in the table, click an entry to view the data. A visual preview of image file data helps to ensure that the files listed as duplicates actually match (Figure 4).

Another unique feature of Czkawka is the *Music Duplicates* tab, which detects duplicate audio files. In this dialog, you can also specify a minimum size. Additional criteria define metadata, such as the title, artist, or album title, which the tool takes into account during the search.

Additional parameters let you include files that no longer contain data, invalid symbolic links, and corrupt files in the list. The *Zeroed Files* option primarily means logfiles emptied by the system, which therefore return a file size of zero bytes when queried.

The *Zeroed Files* option, like the *Invalid Symlinks* and *Broken Files* options, offers no further setting. Pressing *Search* starts the search directly. For invalid symbolic links, as well as for corrupt files, the table gives the error sources, so you can decide whether you want to delete the files. In all categories, select the datasets you want to remove, and then click the *Delete* button.



Minimum file size(in bytes) 16384
Similarity Minimal Very Small Small Medium High [✓ Very High]

```
Similarity Size       Dimensions File Name

Very High 68.02 KiB  3000x100   footer.png
Very High 185.94 KiB 3000x200   header.png

Very High 22.07 KiB  380x146    left-bottom.png
Very High 22.07 KiB  380x146    left-bottom.png

Very High 107.46 KiB 290x126    default_00_v02.png
Very High 107.46 KiB 290x126    fahb_00_v01.png
Very High 107.46 KiB 290x126    hst1_00_v01.png
Very High 107.46 KiB 290x126    mcm1_00_v01.png
Very High 107.46 KiB 290x126    mip1_00_v01.png

Very High 47.55 KiB  1000x256   intro-highres.png
Very High 29.98 KiB  661x169    intro.png

Very High 18.13 KiB  192x192    gimp-wilber.png
Very High 25.64 KiB  256x256    gimp-wilber.png

Very High 16.95 KiB  560x418    FontsDialog.png
Very High 16.95 KiB  560x418    FontsDialog.png
```

Saved results to file results_duplicates.txt
Czkawka 3.1.0

Figure 4: Czkawka offers a preview for graphical files.

Conclusions

Czkawka gives users a universal tool for removing old data files. The software takes an approach that is different from tools such as BleachBit. Instead of application-specific data, Czkawka takes duplicates and temporary content, which can consist of multimedia data and emptied logfiles. Czkawka is also useful for deleting oversized files such as ISO images. ■■■

Info

[1] Czkawka: *https://github.com/qarmin/czkawka*

[2] Installation guides: *https://github.com/qarmin/czkawka/blob/master/instructions/Installation.md*

The latest challenge to copyleft

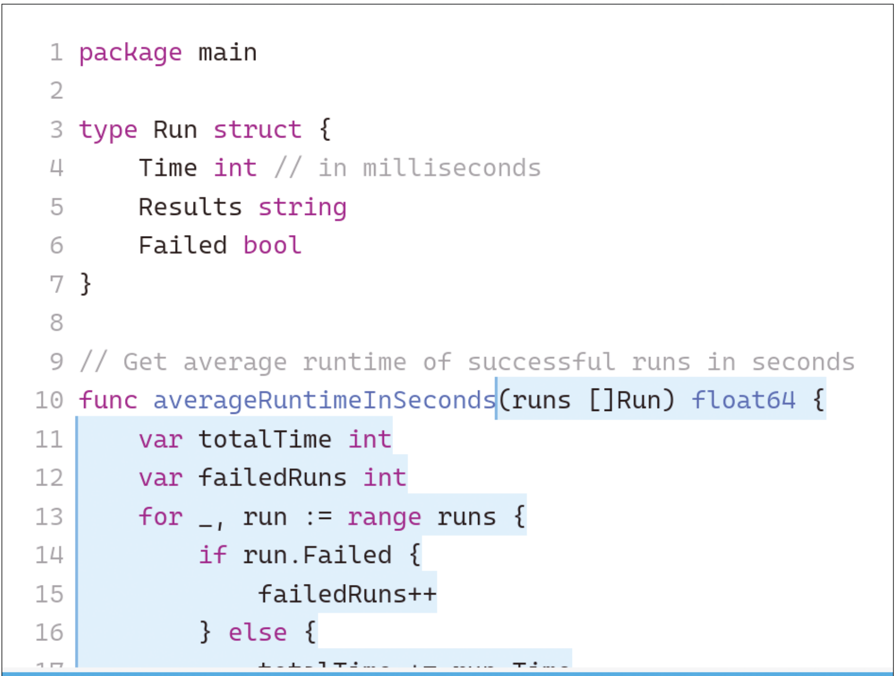
Copilot

GitHub's Copilot takes code autocompletion to a new level but raises copyleft licensing issues. *By Bruce Byfield*

In a world where most people carry a cell phone, autocomplete might seem too widely used to cause contention. However, Copilot [1], a new code autocompletion service for Visual Studio Code (Figure 1), is already raising licensing issues, especially for copyleft licenses such as the different versions of the GNU General Public License (GPL) that are the backbone of free software.

Code assistants are not new. However, Copilot claims to be in a class of its own. Developed by GitHub, Copilot is built on Codex, the new AI system created by OpenAI, and trained with all the public code on GitHub in dozens of programming languages. With this backing, Copilot includes the potential to reduce the time coders take to write or find examples or to learn a new programming language. It also makes possible innovative features such as the conversion of code descriptions in comments to code, autofill for repetitive code, suggestions for code tests, and lists of alternatives, taking autocompletion to entirely new levels.

However, along with these completions comes new licensing problems. Copilot's developers quibble that it is a code synthesizer [2], not a search engine, and insist that "the vast majority of the code that it suggests is uniquely generated and has never been seen before. We found that about 0.1% of the time, the suggestion may contain some snippets that are verbatim from the training set. Many of these cases happen when you don't provide sufficient context (in particular, when editing an empty file),



```go
package main

type Run struct {
    Time int // in milliseconds
    Results string
    Failed bool
}

// Get average runtime of successful runs in seconds
func averageRuntimeInSeconds(runs []Run) float64 {
    var totalTime int
    var failedRuns int
    for _, run := range runs {
        if run.Failed {
            failedRuns++
        } else {
```

Figure 1: Copilot is autocompleter for code that raises challenges for copyleft licenses.

Photo by Arie Wubben on Unsplash

or when there is a common, perhaps even universal, solution to the problem. We are building an origin tracker to help detect the rare instances of code that is repeated from the training set, to help you make good real-time decisions." In other words, exact copying is rare, usually the user's fault, and should be detectable in the future.

Meanwhile, and perhaps even after an origin tracker is included, the possibility remains for unintentional copyright violations. For example, public domain code may still have copyright restrictions. If you borrow public domain code with restrictions, have you violated copyright?

The potential for violations is even stronger with copyleft, which is likely to be common in the code used for training. How much code, if any, can be copied from an application released under a version of the GPL? Does borrowing via Copilot make your code a derivative of GPL code that must therefore also be released under the same license? If so, then under the terms of the GPL, must you include copyright notices and disclaimers?

GitHub's assumption seems to be that using code suggested by Copilot is not a violation of copyleft licenses, because it is based on the training data en masse, not the work of any individual. However, that position causes its own problems. If GitHub is correct, Copilot becomes a means to sneak copyleft code into proprietary programs, making copyleft licenses ineffectual. This will result in challenges from institutions such as the Linux Foundation or the Software Freedom Law Center, which might drag on for years, no doubt accompanied by conspiracy theorists reminding everybody that GitHub is owned by Microsoft. Whichever way you look at it, Copilot seems to be a recipe for chaos.

Hunting for a Solution

Julia Reda, a former German politician who advocates copyright reform and is currently a member of the Berkman Klein Center for Internet & Society at Harvard, has blogged in detail about these alternative viewpoints [3]. Her position is not at all what many free software supporters would like to see. Rather, she warns that applying copyright to Copilot's position would be impractical and self-defeating.

To start with, Reda maintains that copyright infringement does not apply to small snippets of code. Rather, she maintains that copyright violation only applies when the "excerpt used is in turn original and unique enough to reach the threshold of originality." Otherwise, accusations of copyright violations would be made constantly over trivial or unavoidable borrowings. Most borrowing through Copilot, she maintains, is likely to be too limited to reach this threshold – similar to how short quotations in the media aren't copyright violations of novels.

Reda goes on to argue that while considering Copilot's suggestions as derivative works, which would place them under the original copyleft licenses, might be satisfying to free software advocates, this position would create its own problems. In particular, it would mean that all AI-generated material would also be subject to copyright. Although her suggestion that a music label might train "an AI with its music catalogue to automatically generate every tune imaginable" seems far-fetched, it would amount to an extension of copyright – the exact opposite of what most copyleft advocates would desire. As she points out, companies at the World Intellectual Property Organization (WIPO) are already lobbying to apply copyright to AI-generated works, a change that would most benefit major corporations such as Microsoft. In other words, in winning the Copilot battle, free software could lose the war.

However, Reda also rejects GitHub's position. Code, she points out, is not the anonymous work of machines, but of individual coders. According to her, "Copyright law has only ever applied to intellectual creations – where there is no creator, there is no work. This means that machine-generated code like that of GitHub Copilot is not a work under copyright law at all, so it is not a derivative work either. The output of a machine simply does not qualify for copyright protection – it is in the public domain. That is good news for the open movement and not something that needs fixing."

The trouble with this position is that Copilot would remain a means for bypassing the provisions of the GPL. In addition many free software supporters would not find Reda's solution – to leave things as they are – acceptable, especially those who fear the hand of Microsoft behind Copilot. For this reason, the issues raised are unlikely to have a quick solution, particularly at a time when the Free Software Foundation is divided and attempting to reorganize. Will the corporations of the Linux Foundation move to protect their investment in free software instead? So far, the only thing that is clear is that the problem will not disappear easily or quickly. ■■■

Info

[1] Copilot: *https://copilot.github.com/*

[2] GitHub Copilot: *https://copilot.github.com/#faq-who-owns-the-code-github-copilot-helps-me-write*

[3] "GitHub Copilot is not infringing your copyright" by Julia Reda: *https://juliareda.eu/2021/07/github-copilot-is-not-infringing-your-copyright/*

MakerSpace

Create a custom Raspberry Pi OS image

Tailor Made

A few simple steps let you create a preconfigured Raspberry Pi operating system that lets you cut the time required to burn and configure your OS images. *By Michael G. Spohn*

To get a Raspberry Pi up and running, you need to download the latest operating system (OS) release image, burn it to an SD card, slide the card into the Pi, and power it up. After the Pi boots, you log in with the default credentials and run `sudo raspi-config` to configure WiFi, locale, keyboard, and time zone and to enable SSH, I2C, a camera, and whatever else you need for I/O. Next, you update the OS with `sudo apt update`, `sudo apt upgrade`, and reboot before finally logging back in and installing all

One-Time Partition Resizing

Figure 1 provides a simplified view of the root partition resize process. When the Raspberry Pi first boots, it reads command-line parameters from the `/boot/cmdline.txt` file (step 1), with parameters for the output console, TTY speed, root partition ID, and so on.

In step 2, the `init_resize.sh` script performs a number of housekeeping chores, such as preparing and mounting the filesystems. Surprisingly, this script does not resize the physical partition; it simply calls the partition editor (`parted`) to modify the SD card partition table, which determines how big the SD card is and then modifies the second partition entry to expand it to fill the remaining empty space, making the initial resize a quick process. The script also modifies the `cmdline.txt` file, removing the term `quiet` plus removing itself so it will never get called again. Once the housekeeping tasks are performed, the script reboots the Pi (step 3).

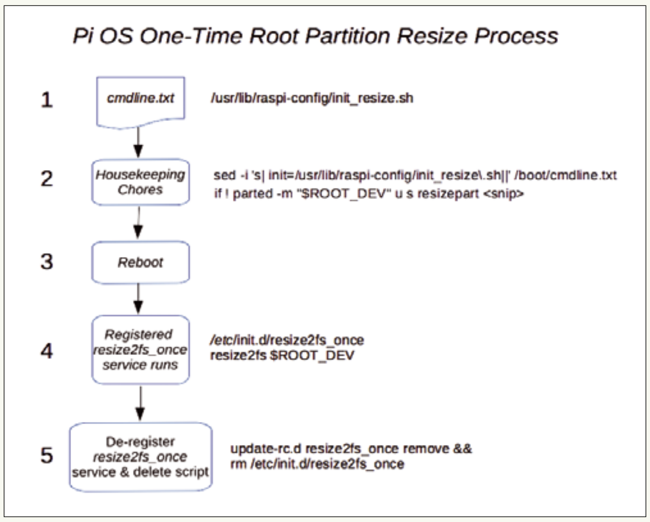

Pi OS One-Time Root Partition Resize Process
1
cmdline.txt
/usr/lib/raspi-config/init_resize.sh
2
Housekeeping Chores
sed -i 's| init=/usr/lib/raspi-config/init_resize\.sh||' /boot/cmdline.txt
if ! parted -m "$ROOT_DEV" u s resizepart <snip>
3
Reboot
4
Registered resize2fs_once service runs
/etc/init.d/resize2fs_once
resize2fs $ROOT_DEV
5
De-register resize2fs_once service & delete script
update-rc.d resize2fs_once remove &&
rm /etc/init.d/resize2fs_once

Figure 1: Raspberry Pi OS partition resize process.

The task to resize the root filesystem is done by a registered service, the short `/etc/init.d/resize2fs_once` shell script (step 4), that runs at system init time. This short-lived script does three things: It calls `resize2fs` to resize the root filesystem to match the partition table modified in step 2 in the flowchart and then performs the other two tasks described in step 5.

Once the root filesystem has been resized, this never has to be done again. In the final step, the partition resize script deletes itself as a registered service and then deletes its own file from the filesystem.

In summary, the root filesystem partition resize task has split responsibilities. An init script executed from the boot command line modifies the SD card partition table to expand the root partition to fill the unused space on the SD card. After a reboot, a registered service physically resizes the partition. This process only needs to run once, so both scripts ensure that they never run again.

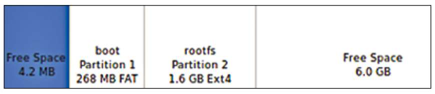


Figure 2: Raspberry Pi OS Lite partition layout.

your favorite software that is not installed on the base image.

For casual Pi users, this procedure is a one-time or rare task. However, for experienced makers who have gone through this drill dozens – if not hundreds – of times, it is a real pain.

In this article, I present a method to create your own custom Raspberry Pi OS image that is just under 2GB in size, making it very fast to burn to any size SD card. No more waiting for an 8 or 16GB image to burn. This custom image not only burns fast, but when it boots, it expands the root filesystem to the full size of the target SD card. When the boot completes, you have a *clean* Pi OS system with your preferred customizations. No more `raspi-config` and wasting time installing your preferred software.

A custom OS on an SD card is a huge time saver if you need to create clean Raspberry Pi OS systems to test software installation scripts. If you burn a lot of Pi OS images in your lab, you will wonder how you got along without this capability in your toolbox.

Most makers never think about the first-time boot of a Raspberry Pi OS image. Behind the familiar on-screen Raspberries, one for each CPU core, and the brief display that says *Resized root filesystem* … a lot is happening. (See the "One-Time Partition Resizing" box.)

Building a Custom OS Image

When you create your super-shiny custom Raspberry Pi OS image, if you do not re-enable the resize process, your image will boot a Pi and work, but it will not automatically resize the root partition to the size of the host SD card. You will have to do it manually, which defeats the whole purpose of the custom image. To create your custom image, you need to do seven things (Table 1). In step 1 it does not matter whether you use the Desktop or Lite Pi OS version.

Burn the Base Image

Again, this is the standard drill we all know and love [2]. It does not matter how big the SD card is as long as the base OS image will fit on it.

Table 1: Custom Image Creation

Step	Action
1	Download the base Pi OS image [1].
2	Burn the base image to an SD card.
3	Modify the image to prevent automatic resizing of the root partition.
4	Boot the image and resize the root partition.
5	Install software and other required customizations.
6	Restore the ability to auto-resize the root partition.
7	Make a `dd` image master file of your customized Pi OS.

Prevent Auto-Resizing

Why would you want to prevent automatically resizing the root partition? Because you want your custom image to be as small as possible. In fact, your custom image will be close to the same size as the Raspberry Pi OS base image – that is, OS Lite (2GB), OS with desktop (4GB), or OS with desktop and recommended software (9GB).

When you burn your custom image to an SD card, the smaller the image, the faster it burns. Image burn time really ads up when you have 8, 16, or 32GB images to burn.

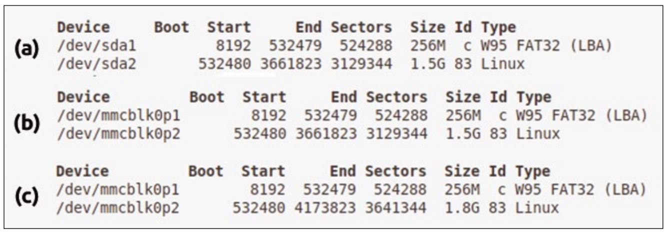
Once you have burned the base OS image to an SD card in a card adapter, plug the card into a USB port on your Linux host. I run Ubuntu Linux on my laptop, and when I plug in a Raspberry Pi OS SD card, it mounts both partitions (`/boot` and `/rootfs`) automatically. The OS Lite image partition layout is shown in Figure 2.

If your system does not mount the partitions, you will have to do it manually. You are only interested in mounting the `/boot` partition (partition 1). On my system, the SD card with the base Pi OS image is device `/dev/sda`.

To mount the first partition (`/boot`), I entered:

```
$ sudo mkdir /media/sdcard
$ sudo mount -o,rw /dev/sda1 ⮐
  /media/sdcard
```

Now the `/boot` partition of the SD card is mounted on `/media/sdcard`. (Be sure to change the `mount` command to match the device ID for your image.)



```
(a) Device     Boot  Start     End Sectors  Size Id Type
    /dev/sda1         8192  532479  524288  256M  c W95 FAT32 (LBA)
    /dev/sda2       532480 3661823 3129344  1.5G 83 Linux

(b) Device         Boot  Start     End Sectors  Size Id Type
    /dev/mmcblk0p1        8192  532479  524288  256M  c W95 FAT32 (LBA)
    /dev/mmcblk0p2      532480 3661823 3129344  1.5G 83 Linux

(c) Device         Boot  Start     End Sectors  Size Id Type
    /dev/mmcblk0p1        8192  532479  524288  256M  c W95 FAT32 (LBA)
    /dev/mmcblk0p2      532480 4173823 3641344  1.8G 83 Linux
```

Figure 3: (a) Mounted `/boot` partition. (b) `/root` partition layout. (c) Resized root partition.

Once the `/boot` partition is mounted (Figure 3a), edit the file named `cmdline.txt` by opening the file in your favorite editor. If you used `sudo` to mount the partition, then you will need to be root to edit any files.

The `cmdline.txt` file contains only one line. The `init` command at the end of the line needs to be removed before you save the file. Be sure you do not add a newline and create a second, empty line, which will break the script.

Now, to unmount the image, use the command:

```
$ sudo umount /media/sdcard
```

Resize the Root Partition

The base OS image is designed to be as small as possible. The OS Lite partition sizes are shown in Figure 3b. This image will barely fit on a 2GB SD card. Analyzing the device reveals that the card does not have much free space. In the stock Pi OS Lite distro, it consumes 92 percent of the root volume.

Remember the concept here: You create a custom image that is as small as possible to save download bandwidth and burn time. Then, at boot time, you expand the root partition to the full size of the host SD card. Simple and elegant.

Because automatic resizing of the `/rootfs` partition is disabled, you do not have much space to add your preferred software packages. You must have enough space on the `/rootfs` partition to accommodate your customizations.

Listing 1: resize_root_part.sh

```
#! /bin/bash
# -------------------------------------------------------------------
# resize_root_part.sh
#
# Script to resize root partition of a Pi OS image.

# Copyright 2021 Michael G. Spohn
# License: Attribution-ShareAlike 4.0 International
#                 (CC BY-SA 4.0)
# https://creativecommons.org/licenses/by-sa/4.0/legalcode
#
# Use this script at your own risk! No warranty of any kind provided.
#
# Contact information:
# mspohn@topmail.com
#
# Version 1.0 - 2021-06-26
# -------------------------------------------------------------------

# Size, in sectors, of additional space to add to root partition.
# Modify this value to suit your needs.
# Value should be divisible by 512.
ADD_SECTOR_COUNT=512000

echo "------------------------------------"
echo "   Script to resize / partition."
echo "       Written by M. Spohn"
echo "          Version 1.0"
echo "           2021-03-11"
echo " Use this script at your own risk."
echo " --------> No warranty <------------"
echo "        mspohn@topmail.com"
echo "------------------------------------"
echo
echo "Current / partition info:"
ROOT_PART_DEV=$(findmnt / -o source -n)
echo -e "\t/ partition is on: " $ROOT_PART_DEV
ROOT_PART_NAME=$(echo "$ROOT_PART_DEV" | cut -d "/" -f 3)
echo -e "\t/ partion name: " $ROOT_PART_NAME
ROOT_DEV_NAME=$(echo /sys/block/*/"${ROOT_PART_NAME}" | cut -d "/" -f 4)
#echo -e "\t/ device name: " $ROOT_DEV_NAME
ROOT_DEV="/dev/${ROOT_DEV_NAME}"
#echo -e "\t/ device: " $ROOT_DEV
ROOT_PART_NUM=$(cat "/sys/block/${ROOT_DEV_NAME}/${ROOT_PART_NAME}/partition")
echo -e "\t/ partion number: " $ROOT_PART_NUM
PARTITION_TABLE=$(parted -m "$ROOT_DEV" unit s print | tr -d 's')
LAST_PART_NUM=$(echo "$PARTITION_TABLE" | tail -n 1 | cut -d ":" -f 1)
ROOT_PART_LINE=$(echo "$PARTITION_TABLE" | grep -e "^${ROOT_PART_NUM}:")
ROOT_PART_START=$(echo "$ROOT_PART_LINE" | cut -d ":" -f 2)
echo -e "\t/ partition start sector: " $ROOT_PART_START
ROOT_PART_END=$(echo "$ROOT_PART_LINE" | cut -d ":" -f 3)
echo -e "\t/ partition end sector: " $ROOT_PART_END
ROOT_DEV_SIZE=$(cat "/sys/block/${ROOT_DEV_NAME}/size")
ORI_PART_SIZE=$(( $ROOT_PART_END - $ROOT_PART_START + 1 ))
echo -e "\t/ partion size in sectors: $ORI_PART_SIZE"
TARGET_END=$(($ROOT_PART_END + $ADD_SECTOR_COUNT))
```

My `resize_root_part.sh` Bash script (Listing 1) default setting is to increase the size of the root partition by 256MB (512,000 sectors):

$$1024 \times 256 = 262{,}144{,}000 \text{ bytes} (256\,\text{MB})$$
$$262{,}144{,}000/512 \text{ (sector size)} = 512{,}000 \text{ sectors}$$

You can modify this by changing the value of the `ADD_SECTOR_COUNT` variable at the top of the script.

Make sure the number of bytes you increase the partition by is divisible by 512 so the partition ends on a sector boundary. Now, copy the script to your Pi and run it as root:

```
$ sudo ./resize_root_part.sh
```

It only takes a few seconds to run. You can see a sample of the output of the script in Figure 4. Now when you look at the SD card size with `fdisk`

```
$ sudo fdisk -l
```

you can see that partition 2 is larger (Figure 3c).

After making disk geometry changes with `sudo reboot`, it is always a good idea to reboot.

Install and Customize Software

Once logged in after rebooting, configure your Pi as desired with `raspi-config`. The first thing I do is configure WiFi and enable SSH, which allows me to connect remotely to the Pi from my laptop. You also can enable a camera and the I2C serial interface, set localization options, and take care of any other settings you require, after which you should exit `raspi-config` and reboot. At this point, *do not* perform a system update or upgrade. An upgrade will download numerous software updates and increase the size of your image.

Now, log back in to your Pi and install your favorite tools and utilities that you want on your custom image. A few of the tools I install include:

- `git`
- `p7zip-full`
- `mc` (Midnight Commander)
- `python3-pip`
- `i2c-tools`
- `cmake`
- `ripgrep`

Configure any of your other favorite tweaks, such as custom `.rc` files, environment variables, and so on. Now is a good time to use `pip` to install your favorite Python modules.

Restore Auto-Resize on First Boot

The final step to complete before you burn a new master image is to restore the ability to automatically resize the root partition, which involves three steps:

- Edit `cmdline.txt` and add the text to call the `raspi-config` init script.
- Put a copy of the `resize2fs_once` file in the `/etc/init.d` folder.
- Register `resize2fs` as an init service.

The `cmdline.txt` file is in the `/boot` folder, and you must be root to edit it. Open the file in an editor and add the following text to the end of line 1:

```
quiet init=/usr/lib/↩
  raspi-config/init_resize.sh
```

Remember, this file can only have one line in it, so do not add a newline before saving the file.

Next, you need to restore the `resize2fs_once` file to `/etc/init.d` in one of two ways. If your Pi is connected to the Internet, you can enter the three commands (Figure 5):

```
$ sudo wget ↩
  -O /etc/init.d/resize2fs_once ↩
  https://raw.githubusercontent.com/↩
  RPi-Distro/pi-gen/master/stage2/↩
  01-sys-tweaks/files/resize2fs_once
$ sudo chmod +x ↩
  /etc/init.d/resize2fs_once
$ sudo systemctl ↩
  enable resize2fs_once
```

This is a quick and easy way to go, but it does have drawbacks. First, you must be connected to the Internet to get it to work. Second, you do not know if or when the file changes, and you do not know the reputation of the source.

An alternative is to copy the `resize2fs_once` file from a current Raspberry Pi OS distro to your Pi. If you do copy a known good file to `/etc/init.d`, you still have to perform the last two commands above.

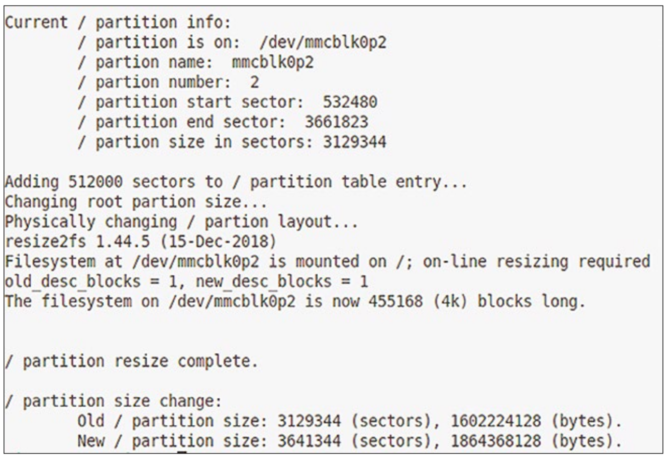
Before creating a custom master image, shut down the Pi and take out the SD card.

Listing 1: resize_root_part.sh (continued)

```
# echo "TARGET_END: " $TARGET_END

# Sanity checks.
if [ "$ROOT_PART_END" -gt "$TARGET_END" ]; then
    echo "Root partition runs past the end of device."
    return 1
fi

if [ "$TARGET_END" -gt "$ROOT_DEV_SIZE" ]; then
    echo "Not enough room on root partition to add $ADD_SECTOR_SCOUNT sectors."
    return 1
fi

echo
echo "Adding $ADD_SECTOR_COUNT sectors to / partition table entry..."

# Use parted to change the partition table.
echo "Changing root partion size..."
if ! parted -m "$ROOT_DEV" u s resizepart "$ROOT_PART_NUM" "$TARGET_END"; then
    echo "Root partition resize failed."
    return 1
fi

# Use resize2fs to physically change partition size.
echo "Physically changing / partion layout..."
resize2fs $ROOT_PART_DEV
if [ $? -ne 0 ]; then
   echo "Error resizing root partion."
   return 1
fi

echo
echo "/ partition resize complete."
echo

ORI_ROOT_SIZE_BYTES=$(($ORI_PART_SIZE * 512))
NEW_ROOT_SIZE_SECTORS=$(($ORI_PART_SIZE + $ADD_SECTOR_COUNT))
NEW_ROOT_SIZE_BYTES=$(($NEW_ROOT_SIZE_SECTORS * 512))
echo "/ partition size change:"
echo -e "\tOld / partition size: $ORI_PART_SIZE (sectors), $ORI_ROOT_SIZE_BYTES
         (bytes)."
echo -e "\tNew / partition size: $NEW_ROOT_SIZE_SECTORS (sectors), $NEW_ROOT_SIZE_
         BYTES (bytes)."
```

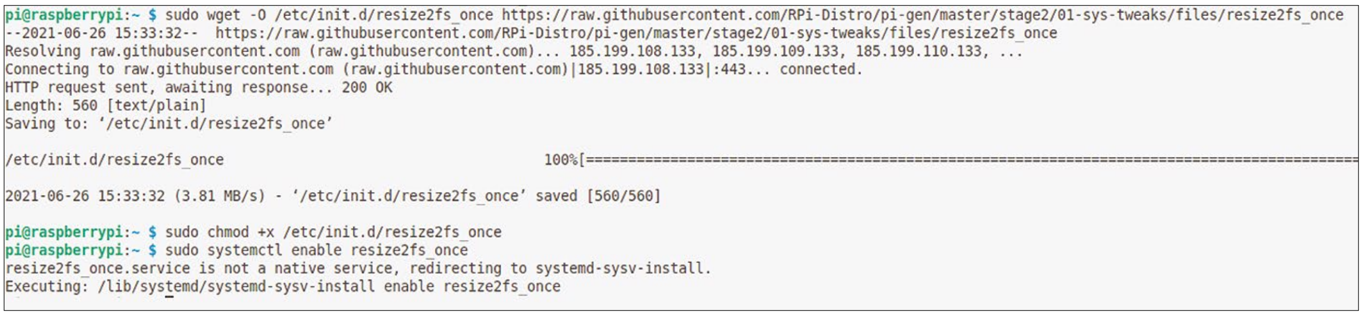
Create a Master File

To make a `dd` image master file of your customized Pi, insert your SD card into a USB adapter and plug it in to your Linux host. My Ubuntu laptop automounts the partitions, although your system might not. Before making



```
Current / partition info:
        / partition is on:  /dev/mmcblk0p2
        / partion name:  mmcblk0p2
        / partion number:  2
        / partition start sector:  532480
        / partition end sector:  3661823
        / partion size in sectors: 3129344

Adding 512000 sectors to / partition table entry...
Changing root partion size...
Physically changing / partion layout...
resize2fs 1.44.5 (15-Dec-2018)
Filesystem at /dev/mmcblk0p2 is mounted on /; on-line resizing required
old_desc_blocks = 1, new_desc_blocks = 1
The filesystem on /dev/mmcblk0p2 is now 455168 (4k) blocks long.


/ partition resize complete.

/ partition size change:
        Old / partition size: 3129344 (sectors), 1602224128 (bytes).
        New / partition size: 3641344 (sectors), 1864368128 (bytes).
```

Figure 4: Output from `resize_root_part.sh`.



```
pi@raspberrypi:~ $ sudo wget -O /etc/init.d/resize2fs_once https://raw.githubusercontent.com/RPi-Distro/pi-gen/master/stage2/01-sys-tweaks/files/resize2fs_once
--2021-06-26 15:33:32--  https://raw.githubusercontent.com/RPi-Distro/pi-gen/master/stage2/01-sys-tweaks/files/resize2fs_once
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 560 [text/plain]
Saving to: '/etc/init.d/resize2fs_once'

/etc/init.d/resize2fs_once                              100%[==========================================================================================

2021-06-26 15:33:32 (3.81 MB/s) - '/etc/init.d/resize2fs_once' saved [560/560]

pi@raspberrypi:~ $ sudo chmod +x /etc/init.d/resize2fs_once
pi@raspberrypi:~ $ sudo systemctl enable resize2fs_once
resize2fs_once.service is not a native service, redirecting to systemd-sysv-install.
Executing: /lib/systemd/systemd-sysv-install enable resize2fs_once
```

Figure 5: Restoration of `resize2fs_once`.

a `dd` image of the SD card, make sure the partitions on the card are not mounted:

```
$ sudo umount /dev/sda1
$ sudo umount /dev/sda2
```

Your SD card partition might not mount on `/dev/sda`, depending on how you computer is configured, so be sure you identify the correct device for your SD card.

Next, determine the last sector that the Raspberry Pi OS occupies on the SD card:

```
$ sudo fdisk -l
```

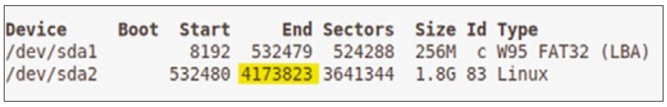
As shown in Figure 6, the last sector used by the second partition in this example is 4,173,823. Make note of this number, because you will need it when you enter the `dd` command.

Note that the `dd` command is unforgiving. It does exactly what you tell it, without annoying confirmation prompts. Use the command with care.

Now, create the image:

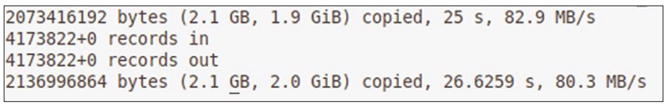
```
$ sudo dd if=/dev/sda of=/home/pi/↩
  pios_lite_2021-01-00_2GB.img ↩
  count=4173822 status=progress
```

The `if=/dev/sda` (infile) parameter specifies that it reads from device `/dev/sda`, the `of` (outfile) parameter names the image file, the number after `count` is one less than the end sector noted in Figure 6, and the `status` parameter shows you what the command is doing.



```
Device     Boot   Start     End Sectors  Size Id Type
/dev/sda1          8192  532479  524288  256M  c W95 FAT32 (LBA)
/dev/sda2        532480 4173823 3641344  1.8G 83 Linux
```

Figure 6: Last sector of Pi OS.



```
2073416192 bytes (2.1 GB, 1.9 GiB) copied, 25 s, 82.9 MB/s
4173822+0 records in
4173822+0 records out
2136996864 bytes (2.1 GB, 2.0 GiB) copied, 26.6259 s, 80.3 MB/s
```

Figure 7: Creating an image with `dd`.

Without using the `count` parameter, `dd` would copy the entire SD card, not just the desired Pi OS sectors. As shown in Figure 7, the command copied 2GB of data to the output file.

Success! You now have a custom Raspberry Pi OS image that burns quickly to an SD card of any size. It will automatically expand to the size of the target SD card and is ready to go on boot – no configuration required.

Remember to update the OS with

```
$ sudo apt update
$ sudo apt upgrade -y
```

when your custom image boots.

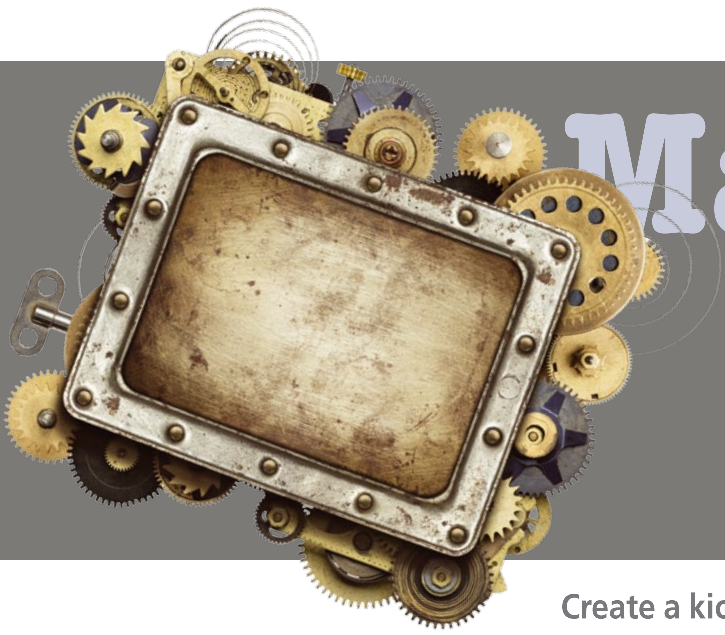
Summary

A custom Raspberry Pi operating system on an SD card can save you time and effort. I hope you found this guide useful and even learned a few things about the Raspberry Pi OS boot process along the way. ■■■

Info

[1] Rasp Pi OS images: *https://raspberrypi.org*

[2] Installing Pi OS images: *https://www.raspberrypi.org/documentation/installation/installing-images/*



MakerSpace

Don't recycle your ePaper: repurpose that old eReader

Up to Date

Create a kiosk display from an old eReader to show data culled from Home Automation, Raspberry Pis, and Arduinos.

By Pete Metcalfe and Leah Metcalfe

My daughter and I looked at trying to repurpose an old eReader. Our goal was to make a kitchen kiosk display that would show items of daily interest. For us, that included news, stocks, and weather data that we pulled from the Internet, along with some local data from Arduino, Home Assistant [1], and Raspberry Pi nodes (Figure 1).

In this article, we look at how to install Linux on an eReader and look at eReader Python apps and kiosk-mode web browser pages.

Getting Started

For this project, we used a Kobo Mini eReader, so the procedure to load a new operating system will vary somewhat according to the eReader model and manufacturer you use.

First, crack open the eReader. The operating system (OS) and data are stored on a microSD card (Figure 2). You should keep the original SD in case you ever need to roll back to the original setup.

eReaders have two main OS choices: Android or Linux. For the requirements of this project, we determined that Linux would be a cleaner option. A Debian image for Kobo eReaders can be found online [2]. If you are trying to repurpose other brands of eReaders, some good how-to guides are available (e.g., for Kindle eReaders [3]).

Installing Debian Linux

A number of different tools can help you move and copy OS images. Raspberry Pi users like to use the Raspberry Pi Imager (`rpi-imager`) utility, which runs on Linux, Windows, and macOS. To install it on Linux, enter:

```
sudo snap install rpi-imager
```

The *Use custom* option lets you put the Debian Linux image on an SD card (Figure 3).

Depending on your eReader and the OS you are using, you might be ready to go after the new microSD card is installed into the eReader. Unfortunately, for our Kobo Mini installation we needed to take another step that moved some files from the original SD chip to the new SD chip:

```
# insert the original microsd
sudo dd if=/dev/mmcblk0 bs=512 ⮐
        skip=1024 count=1 ⮐
        of=/your_path/original.img
# insert the new microSD with the ⮐
  Debian image
sudo dd if=/your_path/original.img ⮐
        bs=512 seek=1024 ⮐
        count=1 of=/dev/mmcblk0
```

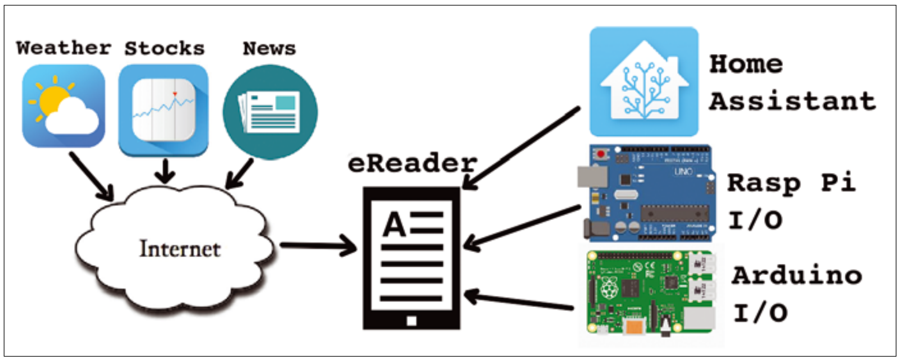


Figure 1: Sources of data for an eReader in kiosk mode.

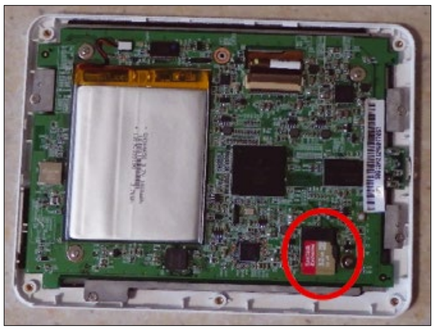


Figure 2: The microSD in the Kobo Mini eReader.

This added step provides all the files required for a clean double-boot installation (Figure 4).

eReader Setup

Now that the OS is installed, you should be able to enable WiFi from the main menus (Figure 5). eReaders have great battery life when they are used as an eReader; however, when they use WiFi 100 percent of the time, their battery life is more like that of a standard tablet.

The eReader has a floating keyboard, so you can work directly on the unit; however, an SSH connection from a PC is definitely easier.

The next step is to add custom apps to the eReader menus. Although we did all our programming remotely, this step allowed us to run and test the apps without an SSH connection.

The menuing on the eReader will vary according to the OS being used. On the Kobo, the Debian OS used the Awesome window manager [4]. Awesome menus are defined in the file `.config/awesome/rc.lua`. Figure 6 shows two extra entries: one for a Python app and the second for a browser kiosk script.

Python on an eReader

The first Python test was to create a full-screen date/time app (Listing 1). This test app showed that fast screen updates were not possible and that it's important to use large buttons for any type of screen access (even to close the app).

Once the basics were worked out, we created a Python app that showed weather data from a Home Assistant node (Figure 7). It took a bit of time to play with font sizing and gray tones before we had something that we liked.

Kiosk-Mode Browser

Another approach for viewing data is to use the eReader as a web client and use another node as the data collector and web server (in this case, a Raspberry Pi). To maximize the eReader screen, the web browser can be used in full-screen or kiosk mode.

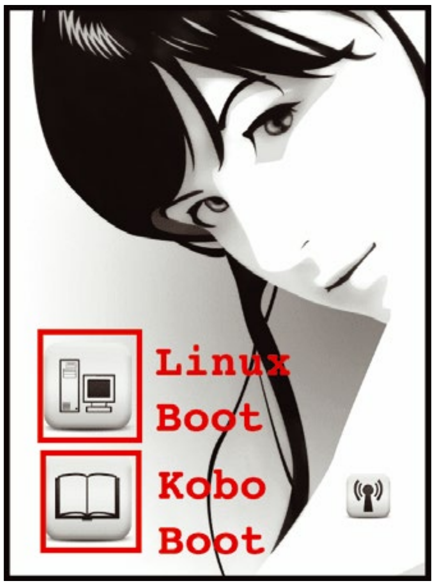


Figure 4: The eReader can now dual-boot to either the original Kobo software or Debian Linux.

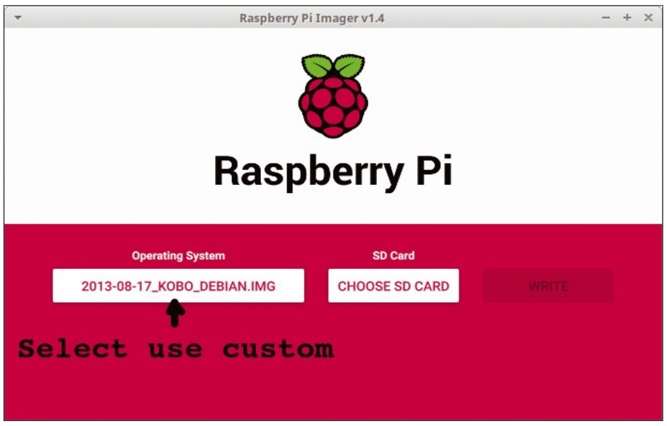
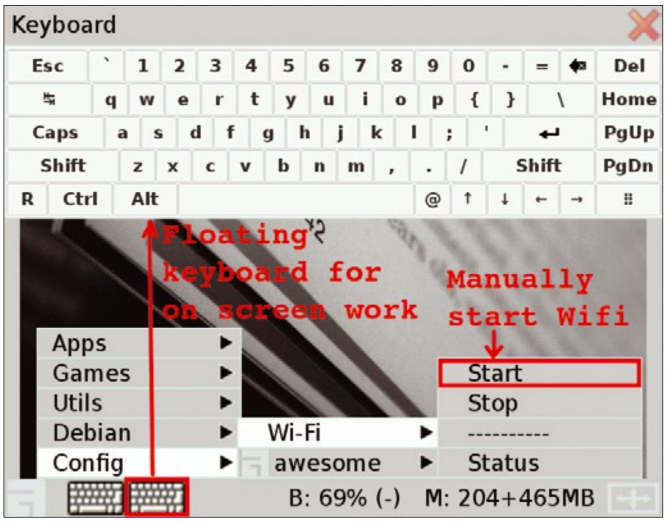


Figure 3: Make an eReader-bootable SD card with the Raspberry Pi Imager utility.



Keyboard
Esc ` 1 2 3 4 5 6 7 8 9 0 - = Del
q w e r t y u i o p { } \ Home
Caps a s d f g h j k l ; ' PgUp
Shift z x c v b n m , . / Shift PgDn
R Ctrl Alt @ ↑ ↓ ← →
Floating keyboard for on screen work
Manually start Wifi
Apps
Games
Utils
Debian
Config
Wi-Fi
awesome
Start
Stop

Status
B: 69% (-) M: 204+465MB

Figure 5: Linux on eReader.

Listing 1: Python Test App

```
# Simple Tkinter Clock
#
from Tkinter import *
import time

def update_clock():
        now = time.strftime("%H:%M")
        lbl_time.configure(text=now)
        today = time.strftime("%A %B %d")
        lbl_date.configure(text=today)
        window.update()
        window.after(5000, update_clock)

window=Tk()

lbl_time=Label(window, text="", font=("Helvetica", 128))
lbl_time.pack()
lbl_date=Label(window, text="", font=("Helvetica", 64))
lbl_date.pack()
btn=Button(window, text="Close",font=("Helvetica",
   32),bg='grey', command=window.destroy)
btn.pack()

window.title('Kitchen Kiosk')
window.attributes("-fullscreen", True)
window.after(1000, update_clock)
window.mainloop()
```

To run Firefox in kiosk mode, enter:

```
firefox ⮐
  --kiosk http://mysite/thepage.htm &
```

For this project, we used Iceweasel, which is a lighter weight browser than Firefox; however, it does not support kiosk mode. A workaround for such browsers is to use `xdotool` to simulate mouse and keyboard actions. To install the `xdotool` utility on Debian and Ubuntu, enter:

```
$ sudo apt-get install xdotool
```

The following script opens Iceweasel and then sends the F11 key to open the browser page in full-screen mode:

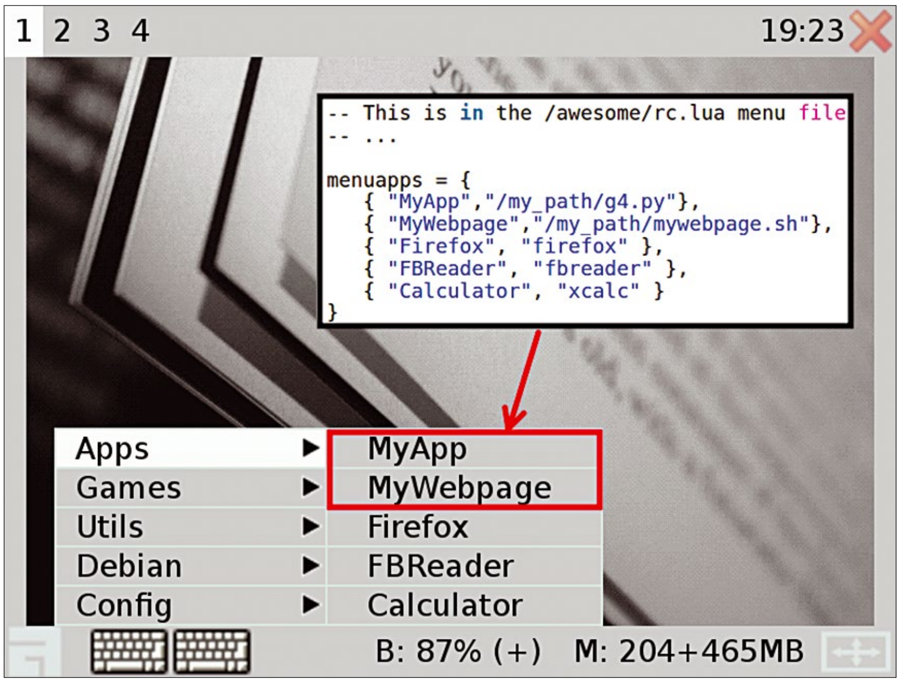
```
#!/bin/bash
# open a web browser to our page and ⮐
  go full screen
#
iceweasel http://our_webserver:80/ &
sleep 15 ; # wait for things ⮐
             to come up
xdotool key F11
```

The Kobo screen is 800x600, so we had to play with the presentation and font sizes. Figure 8 shows an example web page with local weather and stock market information.

Final Comments

It's always nice to reuse or repurpose older equipment. We found that for projects that don't need a fast screen update, an eReader can be an excellent viewer for data fed from a Raspberry Pi or Arduino.

Battery life can be increased by toggling the WiFi on and off when you need to refresh the data. ■■■



```
-- This is in the /awesome/rc.lua menu file
-- ...

menuapps = {
    { "MyApp","/my_path/g4.py"},
    { "MyWebpage","/my_path/mywebpage.sh"},
    { "Firefox", "firefox" },
    { "FBReader", "fbreader" },
    { "Calculator", "xcalc" }
}
```

Figure 6: Adding apps to the eReader menus.

Info

[1] Home Assistant: *https://www.home-assistant.io*

[2] Debian images for Kobo Mini: *https://www.dropbox.com/sh/snsdg1c5cg21kws/AACCAhhLxg5G5OmSgngw3DxYa*

[3] "How to Jailbreak Your Kindle" by Thorin Klosowski, *Lifehacker*, July 21, 2016, *https://www.lifehacker.com.au/2016/07/how-to-jailbreak-your-kindle*

[4] Awesome window manager: *https://awesomewm.org/*

Author

You can investigate more neat projects by Pete Metcalfe and his daughters at *https://funprojects.blog*.

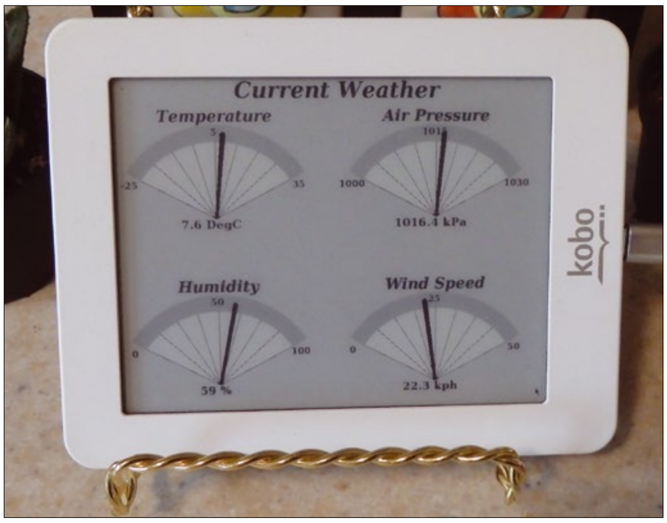
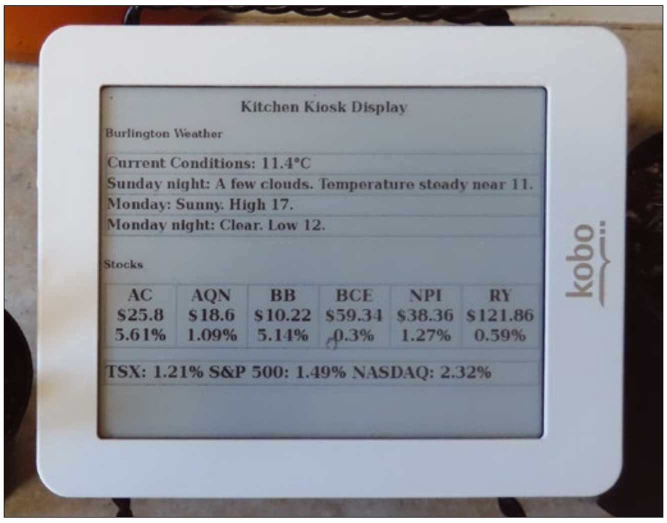


Figure 7: Weather data on an eReader.



Kitchen Kiosk Display

Burlington Weather

Current Conditions: 11.4°C
Sunday night: A few clouds. Temperature steady near 11.
Monday: Sunny. High 17.
Monday night: Clear. Low 12.

Stocks

AC	AQN	BB	BCE	NPI	RY
$25.8	$18.6	$10.22	$59.34	$38.36	$121.86
5.61%	1.09%	5.14%	0.3%	1.27%	0.59%

TSX: 1.21% S&P 500: 1.49% NASDAQ: 2.32%

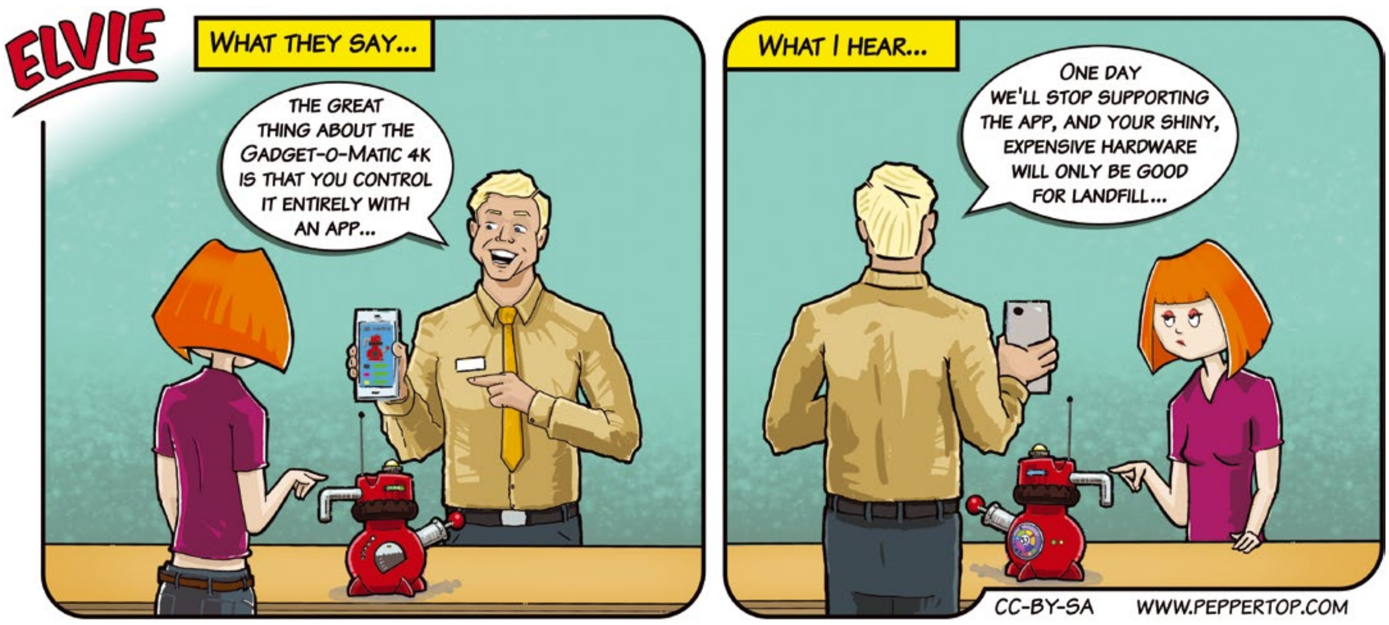
Figure 8: eReader in Kiosk mode.

■■■

You're behind on a big project, you're close to the deadline, and you stay up late, staring at your computer screen. Or maybe it's the middle of the day, and you're staring at your screen because something on the screen is just so darn exciting. Either way, eye strain is the scourge of the modern worker, as we labor through our days online, typing on keyboards and clicking mice. The experts say it is better to look away from your computer screen and focus on something else from time to time to give your eyes a rest. But can you honestly say you heed those wise words of the experts? A free tool called Safe Eyes displays periodic warnings and reminders to protect your eyes during those long dances with your desktop. We help you get started with Safe Eyes in this month's Linux Voice. Also inside, organize your work life with Wekan and customize screen resolutions with xrandr.

Image © Olexandr Moroz, 123RF.com

LINUXVOICE


ELVIE
WHAT THEY SAY...
THE GREAT THING ABOUT THE GADGET-O-MATIC 4K IS THAT YOU CONTROL IT ENTIRELY WITH AN APP...
WHAT I HEAR...
ONE DAY WE'LL STOP SUPPORTING THE APP, AND YOUR SHINY, EXPENSIVE HARDWARE WILL ONLY BE GOOD FOR LANDFILL...
CC-BY-SA WWW.PEPPERTOP.COM

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

MADDOG’S DOGHOUSE

Thinking about the history of Linux, maddog sheds light on why there are so many different flavors of operating systems. BY JON “MADDOG” HALL

Legends of operating system development

Perhaps because I had been reminiscing about the 30 years of Linux kernel development, or perhaps because of a recent discussion over whether the GNU project or the Linux kernel was the most significant part of the system that most people call “Linux,” I began pondering how many people think about software creation as if it were being done today instead of how it was done 40 or 50 (or more) years ago. This kind of thinking often creates “urban legends” about software that are passed on from person to person.

Take the conspiracy theory about how system companies created all these different operating systems to “lock in their customers.” I have been in the computer field for over 50 years and participated in many engineering meetings about new functionality. Not one time did I ever hear customer lock-in as a reason for creating new functionality.

If you remember (or maybe you were not around then, so just trust me on this), computers had relatively small amounts of memory (even mainframes) measured in kilobytes (not gigabytes or even megabytes) and slow, single-core CPUs with slow disk drives.

Normally, these computers were so slow and small that it was hard to create an operating system that could do batch processing, time-sharing, real time, and other loads simultaneously. We also had ideas to create interfaces in the operating system to help make them handle application-specific code, such as medical, manufacturing, educational, and so forth.

If creating an operating system to lock in customers to their systems was the sole reason for the APIs and functionality they supplied, Digital Equipment Corporation (DEC) would have needed only *one* operating system on their PDP-11s, instead of the multitude (over 11) of operating systems DEC offered (and engineered and supported).

As computer speed and memory size started to grow, the idea of having one operating system that could “do it all” began to appear, although it might still be nuanced between business (84 percent of the computer market) and scientific (16 percent of the market).

However, there were still differences in interfaces from vendor to vendor, which meant big differences in APIs and user training that extended to applications that only ran on one or two systems from different vendors.

Out of this came the idea to make portable applications so you could have the same application on various operating systems. Standards development started to help software portability, including standards in languages and language runtime systems resulting from hard work by people from each company to create and test those standards. Some companies made “extensions” to those standards, and (for the most part) the good programmer stayed away from those extensions.

Eventually, as more and more computers were used by more and more people, the concept of portability showed that it was better and cheaper to have the same interfaces across lots of computers rather than having the operating system finely tuned to the actual load.

Since Unix was not created by a system vendor but by a third party, Unix was portable across lots of different hardware and had the same programming interfaces. Of course as Microsoft operating systems rolled out, the same thing could be said for most of the desktop systems that evolved.

This desire to have the same operating system on every platform was demonstrated to me time after time when DEC’s Unix engineers would bring out really great functionality that our customers loved, but the first thing out of the customer’s mouth was “When can I have this on my Sun systems?” Why couldn’t the Unix companies see this coming, when so many of their customers loved using the same software on all of their PCs bought from IBM, Dell, Compaq, etc.?

All of this came back to me in the conversation about GNU and the kernel. Yes, Richard Stallman wanted to build a complete operating system called GNU. But in the environment of the early 1980s with very expensive hardware, no real Internet, and many other limitations, he started with development tools that were freely available and freely changeable that allowed developers to have the same tools across a wide variety of operating systems.

If Richard had first started working on a kernel, he would have had nothing to run on it. And Linus Torvalds did not have to worry about the libraries, compilers, utilities, windowing system, etc., because those already existed.

Some people might say “What about BSD?” The issue with BSD is one of timing. By the time version 1.0 of the Linux kernel was released in early 1994, Linux distributions like SLS, Red Hat, Debian, Slackware, and Yggdrasil (and others) had already released as well, and the BSD lawsuit was still going on.

History and circumstances mean a lot. Many things in the computer industry have happened almost by accident. ■■■

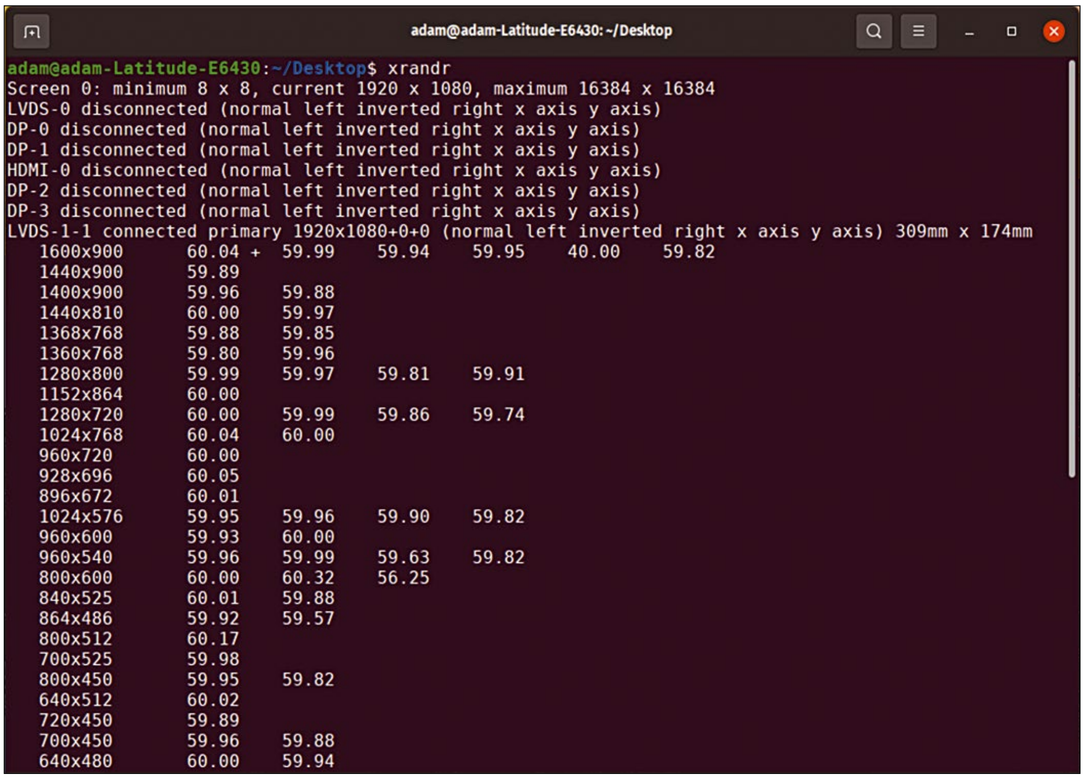
Customizing screen resolutions with xrandr

Screen Makeover

You can boost productivity on an old laptop using xrandr to quickly and easily gain some extra screen real estate. BY ADAM DIX

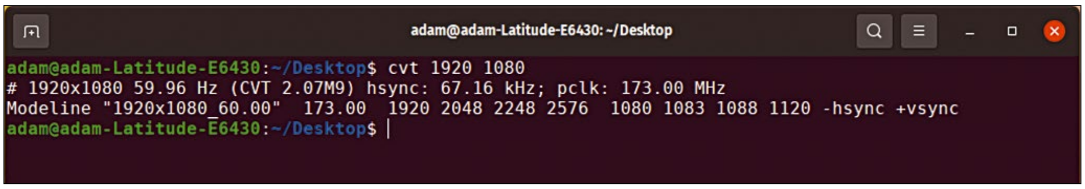
If you are anything like me, you may have an old laptop from the mid-2010s lying around that still works well but has some obvious deficiencies. While the hardware is mostly fine, the 15.6-inch screen has bezels that encroach well into where we now expect to see uninterrupted screen space. With a few modifications, this old laptop is a perfect candidate for a backup Linux machine for when your main laptop inevitably fails (even if you do need to carry the charger with you at all times). After buying a pair of 4GB DDR3 sticks and the least expensive 120GB SSD you can find and installing Ubuntu, this old laptop runs great, but the screen still feels like something from the Windows ME days. All you need is a few simple terminal commands using `xrandr` on Ubuntu [1] to bring the 1366x768-resolution screen up to something a bit more reasonable, greatly improving your overall experience.

There are a few caveats, however. One is that bumping up the resolution will likely cause text to appear fuzzy and less crisp. This is something that most users can become accustomed to in time. It won't look pretty, but at least you will be able to have more than one document or program window open at the same time. Another caveat is that you will likely experience, at the very least, some tearing when moving windows around – nothing that will affect performance in any meaningful way, as long as you don't need high graphic fidelity. Lastly, and obviously, you should probably avoid using this laptop for any video or image production because having an accurate representation on your screen is paramount for those kinds of tasks.



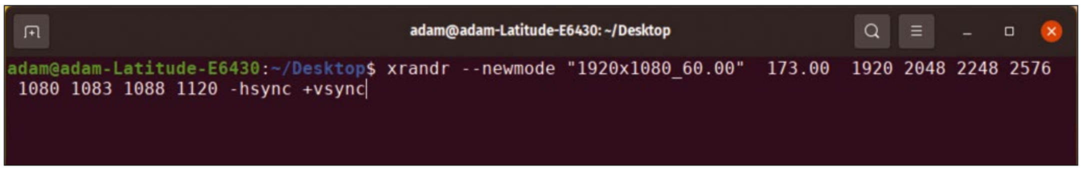
```
adam@adam-Latitude-E6430:~/Desktop$ xrandr
Screen 0: minimum 8 x 8, current 1920 x 1080, maximum 16384 x 16384
LVDS-0 disconnected (normal left inverted right x axis y axis)
DP-0 disconnected (normal left inverted right x axis y axis)
DP-1 disconnected (normal left inverted right x axis y axis)
HDMI-0 disconnected (normal left inverted right x axis y axis)
DP-2 disconnected (normal left inverted right x axis y axis)
DP-3 disconnected (normal left inverted right x axis y axis)
LVDS-1-1 connected primary 1920x1080+0+0 (normal left inverted right x axis y axis) 309mm x 174mm
   1600x900      60.04 +  59.99    59.94    59.95    40.00    59.82
   1440x900      59.89
   1400x900      59.96    59.88
   1440x810      60.00    59.97
   1368x768      59.88    59.85
   1360x768      59.80    59.96
   1280x800      59.99    59.97    59.81    59.91
   1152x864      60.00
   1280x720      60.00    59.99    59.86    59.74
   1024x768      60.04    60.00
   960x720       60.00
   928x696       60.05
   896x672       60.01
   1024x576      59.95    59.96    59.90    59.82
   960x600       59.93    60.00
   960x540       59.96    59.99    59.63    59.82
   800x600       60.00    60.32    56.25
   840x525       60.01    59.88
   864x486       59.92    59.57
   800x512       60.17
   700x525       59.98
   800x450       59.95    59.82
   640x512       60.02
   720x450       59.89
   700x450       59.96    59.88
   640x480       60.00    59.94
```

Figure 1: The `xrandr` command output.



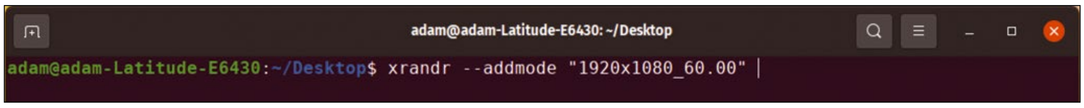
```
adam@adam-Latitude-E6430: ~/Desktop
adam@adam-Latitude-E6430:~/Desktop$ cvt 1920 1080
# 1920x1080 59.96 Hz (CVT 2.07M9) hsync: 67.16 kHz; pclk: 173.00 MHz
Modeline "1920x1080_60.00"  173.00  1920 2048 2248 2576  1080 1083 1088 1120 -hsync +vsync
adam@adam-Latitude-E6430:~/Desktop$ |
```

Figure 2: The `cvt` command output.



```
adam@adam-Latitude-E6430: ~/Desktop
adam@adam-Latitude-E6430:~/Desktop$ xrandr --newmode "1920x1080_60.00"  173.00  1920 2048 2248 2576
 1080 1083 1088 1120 -hsync +vsync
```

Figure 3: The `newmode` command.



```
adam@adam-Latitude-E6430: ~/Desktop
adam@adam-Latitude-E6430:~/Desktop$ xrandr --addmode "1920x1080_60.00" |
```

Figure 4: The `addmode` command.

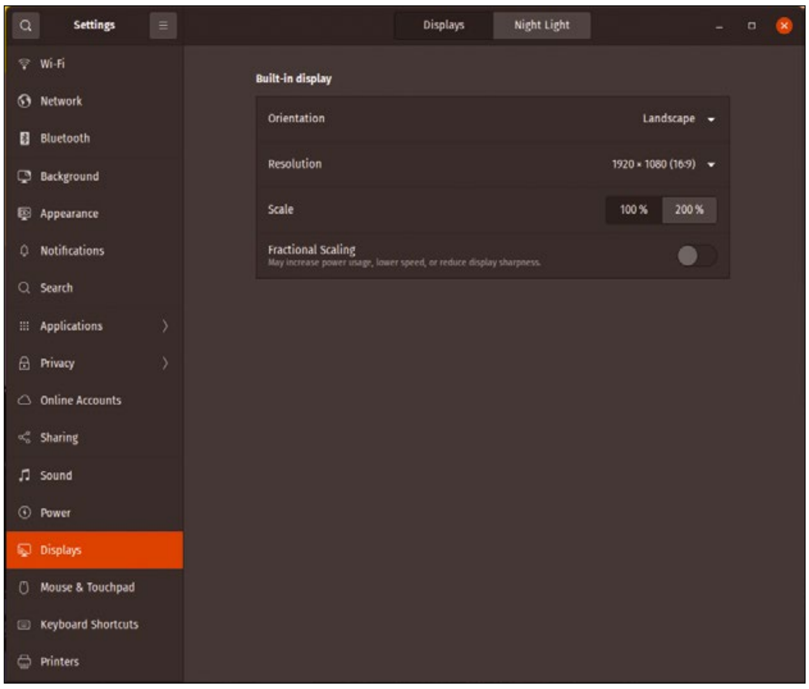
With these caveats in mind, let's break down the process.

Custom Resolution

First you need to determine the base X Window System [2] configuration to which your custom resolution will be added. To do this, open a terminal window and run:

```
$ xrandr
```

For reference later, write down which display output is being used. This will be something like `HDMI-#`, `DP-#`, `LVDS-#`, or `VGA-#` (where the `#` is typically a `0` or `1`). The first section indicates the connection technology, and the digit indicates the monitor number. This will be important to know later. For example, Figure 1 shows the output from `xrandr` for the Dell E6430 14-inch laptop, circa 2012, that I used to write this article.



Built-in display

Orientation Landscape
Resolution 1920 × 1080 (16:9)
Scale 100 % 200 %
Fractional Scaling
May increase power usage, lower speed, or reduce display sharpness.

Figure 5: Your desired resolution now shows up in the Gnome Display Settings panel.

Next, determine your monitor's characteristics at your selected resolution. You will want to continue to use the same aspect ratio (width to height) when scaling to a higher resolution. For instance, a 1366x768 monitor that is 14 inches or larger can typically accommodate any 16:9 resolution up to 2560x1440 while still offering fairly legible text. To determine the characteristics needed, run the following command:

```
$ cvt <desired width in pixels> ⮐
  <desired length in pixels>
```

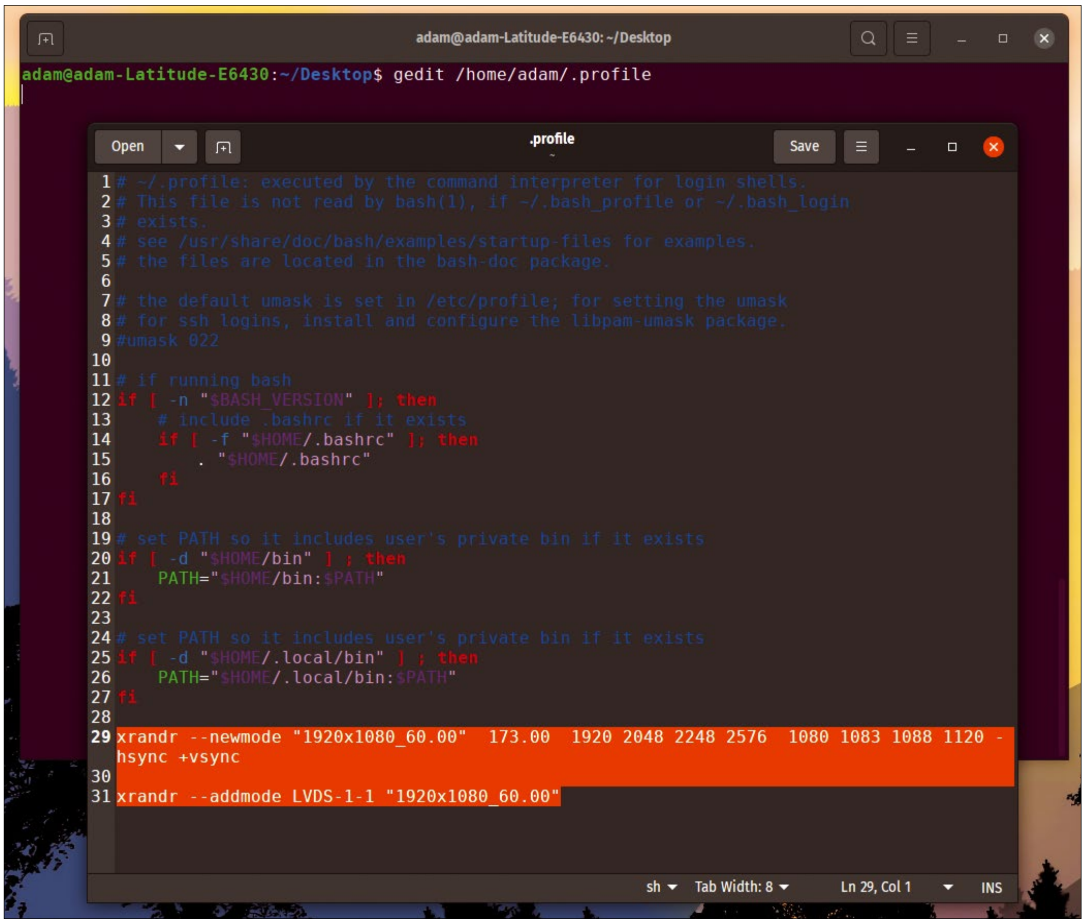
Running this `cvt` command on my Dell for a desired resolution of 1920x1080 results in the information shown in Figure 2.

The modeline listed in the `cvt` command's output shown in Figure 2 should also be noted because that, too, will be used later. At this point, use the `xrandr --newmode` command to add the desired mode to the configuration. Copy and paste the modeline information after `xrandr --newmode` so that the correct timings and resolution will be used (Figure 3).

Finally, use the `xrandr --addmode` command followed by the desired resolution and frequency (Figure 4), which will then be accessible in the Gnome Display Settings panel in the Gnome Control Center (Figure 5).

Persistence

Now you can open the Gnome Control Center and choose the desired resolution. You can then see how good (or bad) the new resolution looks on your particular machine. However, there is one



```
adam@adam-Latitude-E6430:~/Desktop$ gedit /home/adam/.profile
```

```
# ~/.profile: executed by the command interpreter for login shells.
# This file is not read by bash(1), if ~/.bash_profile or ~/.bash_login
# exists.
# see /usr/share/doc/bash/examples/startup-files for examples.
# the files are located in the bash-doc package.

# the default umask is set in /etc/profile; for setting the umask
# for ssh logins, install and configure the libpam-umask package.
#umask 022

# if running bash
if [ -n "$BASH_VERSION" ]; then
    # include .bashrc if it exists
    if [ -f "$HOME/.bashrc" ]; then
        . "$HOME/.bashrc"
    fi
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/bin" ] ; then
    PATH="$HOME/bin:$PATH"
fi

# set PATH so it includes user's private bin if it exists
if [ -d "$HOME/.local/bin" ] ; then
    PATH="$HOME/.local/bin:$PATH"
fi

xrandr --newmode "1920x1080_60.00"  173.00  1920 2048 2248 2576  1080 1083 1088 1120 -
hsync +vsync

xrandr --addmode LVDS-1-1 "1920x1080_60.00"
```

Figure 6: The `.profile` configuration file with custom resolution commands added.

problem at this point: When you sign out or reboot, your custom resolution will no longer be in the Gnome Control Center, forcing you to re-enter these commands in the terminal to access the resolution each time you sign in.

There is a simple workaround for this problem: Add the `newmode` and `addmode` commands to the end of the `.profile` file in your `/home/user directory` (Figure 6). I prefer to simply open `Home`, view hidden files by using Ctrl+H, open the `.profile` file with gedit, copy and paste those two commands at the very bottom, and save. Pressing Ctrl+H again will hide the hidden files once more and resume the default view. Now the resolution change will survive reboots and logout/login for the user whose `.profile` was amended. Purists may want to use Vim or nano, but to each his or her own. If you're feeling indecisive, you can always run

```
gedit /home/<user>/.profile
```

from the terminal and add the two needed lines that way.

Conclusion

With a few simple commands in the terminal, you can use `xrandr` to customize the resolution on older machines, breathing new life into old hardware. ■■■

Info

[1] xrandr: *https://manpages.ubuntu.com/manpages/precise/man1/xrandr.1.html?_ga=2.219529817.1184336464.1626887224-1801470088.1625846779*

[2] X Window System: *https://www.x.org/wiki/*

The Author

Adam Dix is a mechanical engineer and Linux enthusiast posing as an English teacher after playing around a bit in sales and marketing. You can check out some of his Linux work at the EdUBudgie Linux website (*https://www.edubudgie.com*).

Lock Down Microsoft IIS
Secure Containers
ADMIN Network & Security
Secure Containers with a hypervisor DMZ
CYBERSECURITY Machine learning defends the IT infrastructure
4 Password Managers
Safe Containers Automate updates of container components
MinIO Local object store with an S3 interface
Pulumi Multicloud orchestrator
Securing TLS Put an end to fake certificates
.NET Core on Linux
k3OS and Flatcar Container microdistributions
VMware vRealize for Hybrid Clouds
Automated Compliance with Chef InSpec
Create a Private Docker Registry
LEAN WEB Servers Go lighter and get faster
Hiawatha – A light-footed web server
Lighttpd – Lean, fast, and simple to configure
Kea – A modern DHCP server
4 Software-based load balancers
OpenEBS Cloud-native storage
Custom MIBs Get smarter with system monitoring
CALICO Layer 3 SDN
Dockly Visualize containers
11 Tricks for Windows Servers
Tanzu Connect VMware to Kubernetes
StatusBay See what happens behind Kubernetes commands
Bare Metal Deployment
Ironic Automatically install and set up compute nodes
Tinkerbell Granular bare metal deployment and life-cycle management
EKS Automated deployment of Amazon Elastic Kubernetes Service
6 RESCUE SYSTEMS FROM ANTIVIRUS VENDORS
AUTOPSY Forensic analysis
Automation Get ready for DevOps with Ansible, CFEngine, Chef, Puppet, and SaltStack

Preventing computer eye strain

Sight Saver

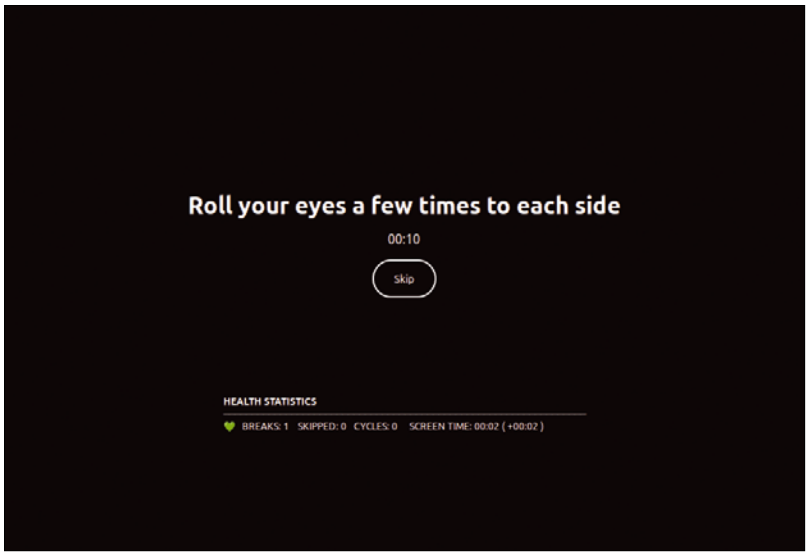
If you work or study from a home office, you probably sit at your computer for hours without a break. Safe Eyes reminds you to take regular breaks and take care of your eyes. BY CHRISTOPH LANGNER

At first glance, working from home seems very convenient. You save time on your commute, money on transportation costs, and avoid overcrowded buses or trains. However, occupational safety often falls by the wayside at home. Most likely, you don't have an ergonomic workstation, and you don't turn away from your screen for a short conversation with a coworker or leave your desk for a meeting on another floor or adjoining building. Instead, your gaze remains fixed on the screen for hours.

At home, the responsibility for taking breaks and evenly spacing out your work falls entirely into your hands. You can use a Pomodoro timer [1] to regularly remind you to take breaks. However, if you continue to stare at the screen during your break – maybe surfing the web or watching a movie – you aren't giving your eyes time to recover.

If you find it difficult to move away from your screen, you may want to add Safe Eyes to your break time routine. An open source application developed for the Gnome desktop, Safe Eyes blacks out and locks your screen for a few seconds for each break (Figure 1), forcing you to actually stop working. Based on recommendations from medical experts, Safe Eyes implements several shorter breaks rather than a few longer ones.

Figure 1: Safe Eyes reminds you to look away from the screen at regular intervals.


Roll your eyes a few times to each side

00:10

HEALTH STATISTICS

BREAKS: 1 SKIPPED: 0 CYCLES: 0 SCREEN TIME: 00:02 (+00:02)

Installation

Safe Eyes is not yet included in the package sources of the popular distributions, but the developers do provide installation instructions for them on both their website [2] and their GitHub page [3]. Safe Eyes offers a PPA for the simplest possible installation on Ubuntu (Listing 1). On Arch Linux, Safe Eyes can be installed using the *safeeyes* package from the Arch User Repository.

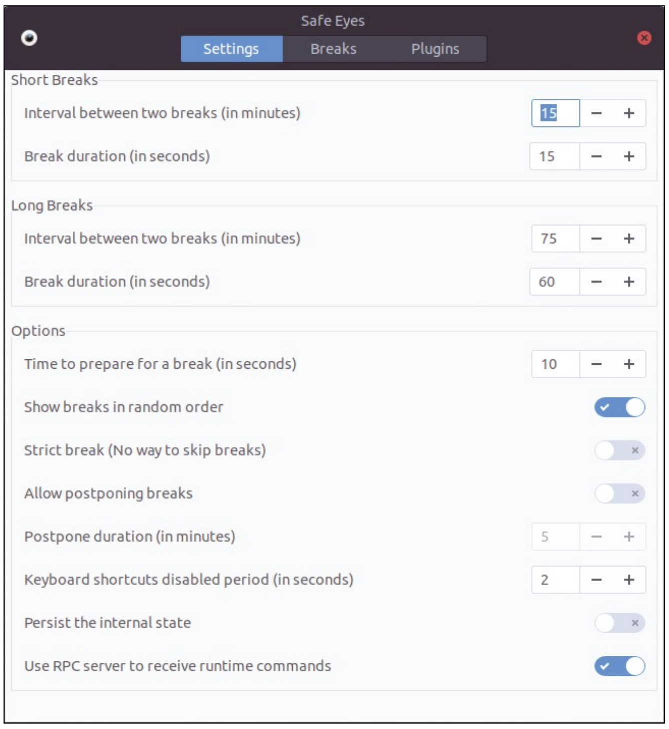
Safe Eyes starts automatically in the background when you log in to your desktop. The startup behavior can be managed in Gnome using the Gnome Tweak Tool (in the application menu below *Optimizations*) by clicking on the *Startup Applications* tab. If you select *Safe Eyes* from the application menu, the program opens its settings. You then have the option of configuring the interval between breaks as well as their length. You can also specify whether breaks can be postponed or skipped (Figure 2).

Respite for the Eyes

In the *Breaks* tab, you define the breaks, which are usually selected at random. Safe Eyes distinguishes between short breaks of 15 seconds (every 15 minutes) and longer breaks of 60 seconds (every 75 minutes). You can adjust or delete the breaks entered in the standard configuration based on your preferences (Figure 3). If you

Listing 1: Installation on Ubuntu

```
$ sudo add-apt-repository ppa:slgobinath/safeeyes
$ sudo apt update
$ sudo apt install safeeyes
```


Safe Eyes
Settings | Breaks | Plugins

Short Breaks
Interval between two breaks (in minutes) 15
Break duration (in seconds) 15

Long Breaks
Interval between two breaks (in minutes) 75
Break duration (in seconds) 60

Options
Time to prepare for a break (in seconds) 10
Show breaks in random order ✓
Strict break (No way to skip breaks) ✗
Allow postponing breaks ✗
Postpone duration (in minutes) 5
Keyboard shortcuts disabled period (in seconds) 2
Persist the internal state ✗
Use RPC server to receive runtime commands ✓

Figure 2: Configuring the individual break times in the Safe Eyes settings.

make a mistake while configuring, you can reenable the application's default settings by clicking on the eye symbol in the top left corner of the application window and selecting the *Reset* menu item.

In the *Plugins* tab, you can enable further functions. For example, the *Do Not Disturb* plugin will not interrupt the display if a program is running in fullscreen mode, for instance, when playing a video (Figure 4). You can also define other programs here that you do not want Safe Eyes to interrupt under any circumstances. The window titles are determined by the command

```
xprop | grep WM_CLASS
```

followed by a click on the application in question. However, this does not work with Wayland.

Conclusions

Safe Eyes, the Pomodoro timer, or shifting your screen's color temperature to warm in the evening (*Settings* | *Screens* | *Night mode* in Gnome) can help protect your eyes and enable concentrated work over longer periods of time. However, it is ultimately up to you to avoid looking at your cell phone during scheduled breaks. Instead, get up, move around, and let your eyes wander into the distance. ■■■

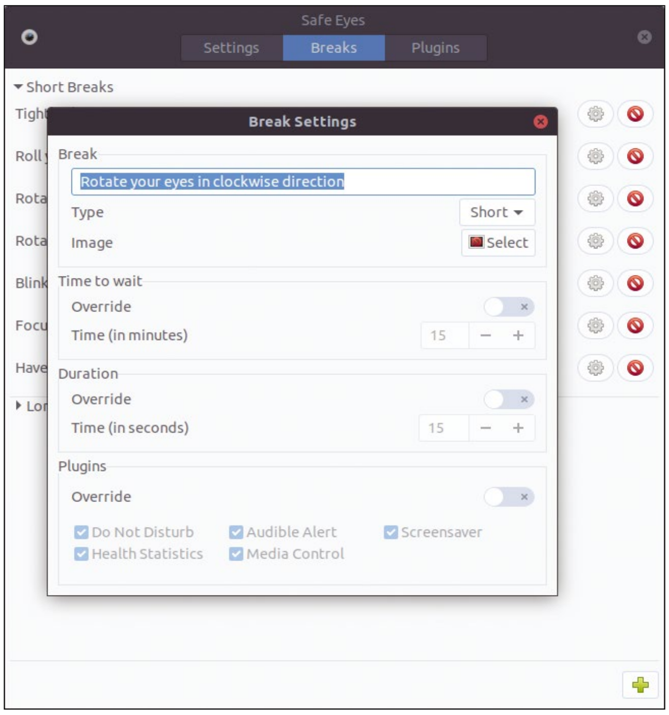
Info

[1] Pomodoro timer: *https://pomofocus.io*

[2] Safe Eyes: *http://slgobinath.github.io/SafeEyes*

[3] Github repository: *https://github.com/slgobinath/SafeEyes*

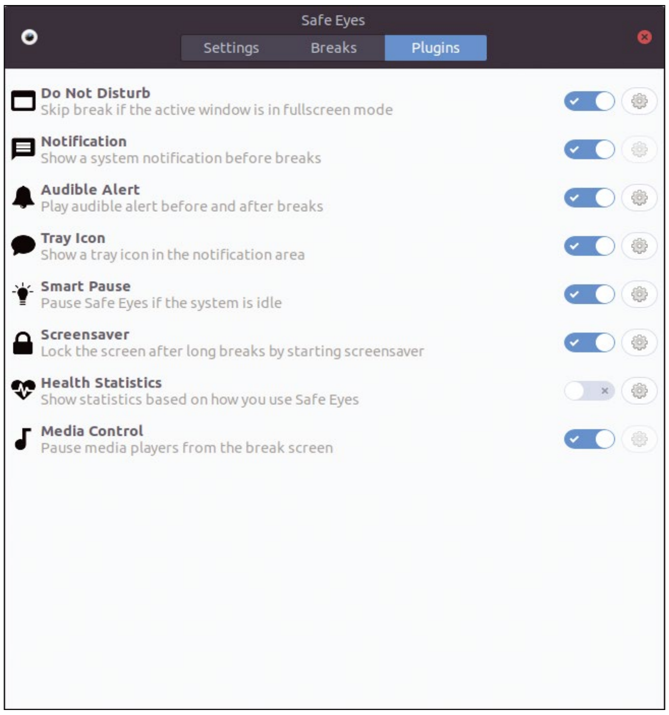
Figure 3: You define what the break and distraction should look like in the *Breaks* tab.


Safe Eyes
Settings | Breaks | Plugins

Short Breaks
Tight
Roll
Rota
Rota
Blink
Focu
Have
Lor

Break Settings
Break
Rotate your eyes in clockwise direction
Type Short
Image Select
Time to wait
Override ✗
Time (in minutes) 15
Duration
Override ✗
Time (in seconds) 15
Plugins
Override ✗
✓ Do Not Disturb ✓ Audible Alert ✓ Screensaver
✓ Health Statistics ✓ Media Control

Figure 4: You can extend Safe Eyes' functionality through plugins. Here, *Health Statistics* was not enabled because the *python-croniter* package was missing.


Safe Eyes
Settings | Breaks | Plugins

Do Not Disturb ✓
Skip break if the active window is in fullscreen mode
Notification ✓
Show a system notification before breaks
Audible Alert ✓
Play audible alert before and after breaks
Tray Icon ✓
Show a tray icon in the notification area
Smart Pause ✓
Pause Safe Eyes if the system is idle
Screensaver ✓
Lock the screen after long breaks by starting screensaver
Health Statistics ✗
Show statistics based on how you use Safe Eyes
Media Control ✓
Pause media players from the break screen

FOSSPicks

Sparkling gems and new releases from the world of Free and Open Source Software

Despite adding it to his basket within nanoseconds of availability and then spending 90 minutes struggling to pay, Graham still has to wait a year for his Steam Deck! BY GRAHAM MORRISON

DJ software

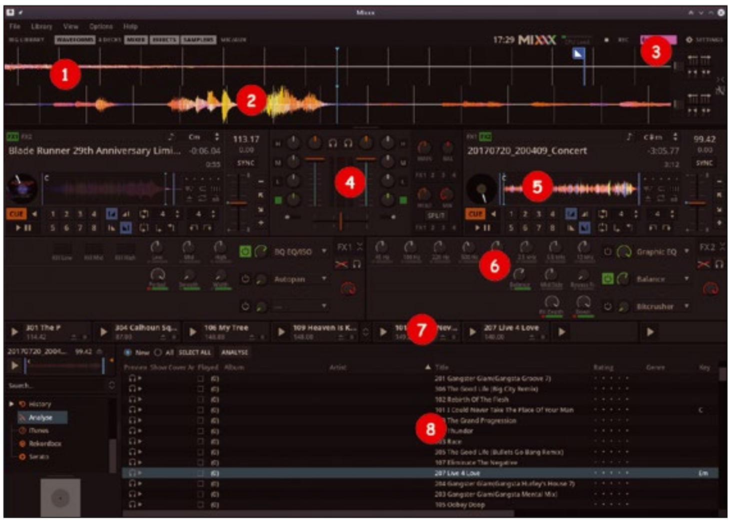
Mixxx

Mixxx started life more than 20 years ago as an application to help DJs organize and play their playlists, and it has since evolved into an incredible, fully fledged, live performance user interface for budding and professional DJs, podcasters, and radio hosts. It's a worthy competitor to the proprietary (and Linux incompatible) Native Instruments Traktor Pro, with both applications using the same user-interface paradigm. That paradigm involves two or more decks split horizontally across the main window with a bank of virtual faders and controllers in between them. The number of controls you see here depends on the number of decks you add.

A deck is akin to an old vinyl record deck or turntable, now popular again in their own right. Nightclub DJs would typically arrange two decks side-by-side with a mixer in the middle, so tracks could be switched between them with no delay. It's this configuration that Mixxx emulates, albeit for the 21st century. Instead of records, the virtual decks load one or more audio files for playback, with the virtual faders used to mix between opposite decks. Audio files can be loaded individually, dragged and dropped, or queued as a playlist. This configuration allows you to seamlessly blend between various audio tracks, whether they are breakbeats or advertising jingles, and either record the output as a single audio file or use it as the source of a live mix or broadcast.

To make transitions as seamless as possible, Mixxx is able to analyze the audio files to detect their tempo, frequency range, and even key. All of this is shown on the waveform, with the tempo highlighted as vertical bars, frequency range as color, and key as an annotation. Pitch controls can then change either the key, playback speed, or both, making it incredibly easy to match different audio files for a more integrated mix. If you still yearn to exercise those old DJ skills, you can drag your mouse across the waveforms to virtually scratch the needle across the record, producing beautifully authentic sounds. This can be taken even further by using a real turntable spinning a timecode vinyl record, which can be tracked by Mixxx inputs to accurately map your scratching to changes in playback speed. Everything can also be automated, either by the many keyboard shortcuts or via MIDI automation, where every knob and slider can be assigned a MIDI control value for external control. These can be scripted for even more control.

Modern effects can be added, chained together, and even isolated to certain audio buses or a headphone mix. You can also keep a microphone channel clear of other effects so you can announce you're running Mixxx on Arch Linux. All of this can be configured with Jack and the comprehensive settings. The latest release adds to these effects with color options for tracks, intro and outro marking, multithreaded analysis and more accurate key detection, and support for lots of physical deck controllers. Twenty years of development have enabled Mixxx to become a one-stop solution for playing, tracking, and DJing music. It's remarkable that this kind of software is being made and produced under an open source license.



1. Waveform: With two audio files loaded, their waveforms appear here; you can drag-scratch the playback position and match tempos. **2. Spectral analysis:** Waveform colors indicate the audio's frequency range, helping you to match genres and timbre. **3. Broadcast:** A mix can be broadcast live or recorded for posterity. **4. Fader:** With two audio decks on either side (by default), the fader mixes between them. **5. Audio deck:** Drag and drop audio files; cue them for playback; and play, pause, and rewind their audio. **6. Effects:** Each deck can have its own effects, such as filters and delays. **7. More decks:** Audio added here can be easily triggered or dragged into the central decks. **8. Playlist:** Mixxx can scan and analyze all of your audio files, letting you arrange and sequence them for each project.

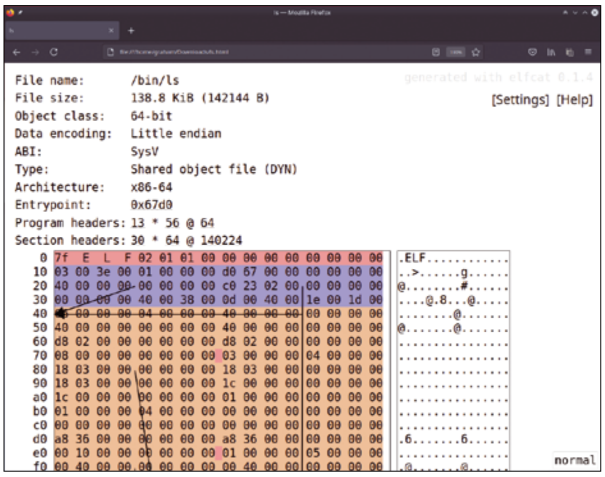
Project Website
https://mixxx.org/

Binary explorer

elfcat

`elfcat` is a tiny yet brilliant command for the terminal. Its cute name also hides its functionality; this is a command that outputs (`cat`s) the contents of a binary executable (`elf`) in a human-readable way, letting you explore the hidden secrets of your favorite applications from the command. Taking the name of a binary as its only argument, `elfcat` produces output from the binary. The raw output isn't especially human readable and not especially accessible from the command line, but it's a lot more useful than using `cat` to see a binary. The output is concatenated into an HTML file which (of course) looks best when dropped into a web browser. This is a brilliant idea because the HTML output is fully annotated to describe everything detected about the binary file. And if you wanted, you could still view it from the command line.

When loaded into a web browser, the main page shows the hex output of the raw binary of the file, alongside any ASCII decoded values, much like any hexadecimal editor. Above this is some general information about the executable, such as its size, object class, and type. The clever parts appear when you start to hover your cursor over the colored sections in the hexadecimal blocks of code. Annotations appear as text boxes on the right, informing you of what each section does and how it's linked to other blocks of code. It's still difficult to understand unless you have some knowledge of assembler, but it's a fascinating insight into what the binary contains and a lot easier than using other tools to decompile and view an executable, especially if all you want to do is find some text or see where different sections of an executable live within the raw file. Future features will include memory mapping visualization, which would be unique.



```
File name:        /bin/ls                          generated with elfcat 0.1.4
File size:        138.8 KiB (142144 B)                         [Settings] [Help]
Object class:     64-bit
Data encoding:    Little endian
ABI:              SysV
Type:             Shared object file (DYN)
Architecture:     x86-64
Entrypoint:       0x67d0
Program headers: 13 * 56 @ 64
Section headers: 30 * 64 @ 140224
 0 7f  E  L  F 02 01 01 00 00 00 00 00 00 00 00 00  .ELF............
10 03 00 3e 00 01 00 00 00 d0 67 00 00 00 00 00 00  ..>.....g.......
20 40 00 00 00 00 00 00 00 c0 23 02 00 00 00 00 00  @.......#.......
30 00 00 00 00 40 00 38 00 0d 00 40 00 1e 00 1d 00  ....@.8...@.....
40 06 00 00 00 04 00 00 00 40 00 00 00 00 00 00 00  ........@.......
50 40 00 00 00 00 00 00 00 40 00 00 00 00 00 00 00  @.......@.......
60 d8 02 00 00 00 00 00 00 d8 02 00 00 00 00 00 00  ................
70 08 00 00 00 00 00 00 00 03 00 00 00 04 00 00 00  ................
80 18 03 00 00 00 00 00 00 18 03 00 00 00 00 00 00  ................
90 18 03 00 00 00 00 00 00 1c 00 00 00 00 00 00 00  ................
a0 1c 00 00 00 00 00 00 00 01 00 00 00 00 00 00 00  ................
b0 01 00 00 00 04 00 00 00 00 00 00 00 00 00 00 00  ................
c0 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00  ................
d0 a8 36 00 00 00 00 00 00 a8 36 00 00 00 00 00 00  .6.......6......
e0 00 10 00 00 00 00 00 00 01 00 00 00 05 00 00 00  ................
f0 00 40 00 00 00 00 00 00 00 40 00 00 00 00 00 00  .@.......@......
                                                                        normal
```

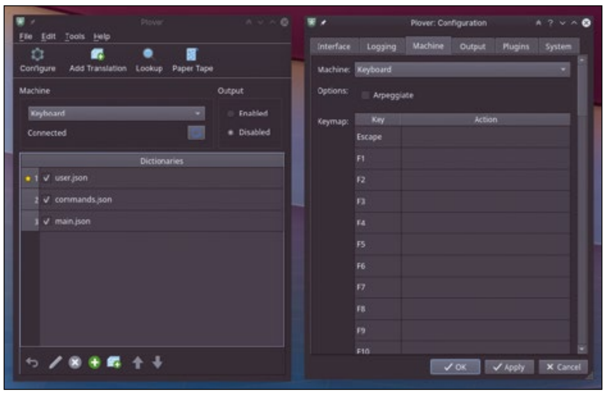
Get a quick overview and understanding of Linux executables with `elfcat`.

Project Website
https://github.com/ruslashev/elfcat

Stenotype emulator

Plover

Stenography is the ancient art of documenting something in a shortened form without losing any information, usually by replacing words and characters with more readily written symbols. That particular mode of stenography became known as shorthand – beloved by assistants everywhere. But the original stenography lives on and is now most commonly associated with court-of-law stenographers, transcribing everything from murder motives to matrimonial misdemeanors. Modern stenographers do this with a keyboard input device, called a stenotype, that looks more like an octave of a piano keyboard. The stenotype translates chords of keys into words and sentences almost instantaneously, enabling stenographers to type up to 300 words per minute, compared to the 60-80 most of us can manage.
This epic typing speed obviously has many advantages for Linux users, but stenotype machines are expensive, unconventional, and take a lifetime to master. Far better if you can experiment with stenography without the outlay, and that's exactly what Plover does. Plover translates multiple concurrent keypresses, roughly mapped to the same locations you'd find keys on a stenotype machine, into words and phrases preconfigured in editable and addable dictionaries. To do this properly, your keyboard will need to be able to support more than a few concurrent keypresses. This is known as N-key rollover (NKRO). While cheap keyboards may struggle (although some of the listed compatible keyboards are less than $30), gaming and other high-quality keyboards should be compatible. There are even some excellent DIY examples you can build, or you can start ripping key caps from your own keyboard. But even with Plover and the right hardware, learning stenography is hard. It will take a long time and plenty of practice before you even approach the input efficiency of your current setup, but the end result would definitely be worth it.



Plover includes an excellent guide to learning stenography, along with other tools, such as an arcade game, to help you practice.

Project Website
https://github.com/openstenoproject/plover

macOS filesystem

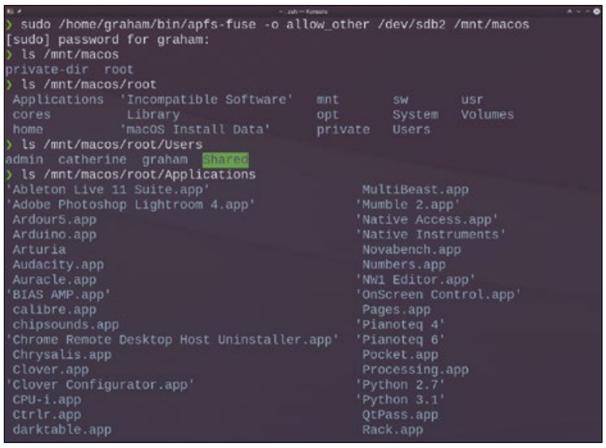
apfs-fuse

Even when you're not using other operating systems, it's always been important for Linux to be able to access other filesystems. For example, most of us read and write to Microsoft VFAT- and FAT32-formatted devices without even thinking about it, whether it's USB storage, a camera, or a shared hard drive partition. That's because FAT has been part of the kernel from almost the very beginning and subsequently built into many embedded devices. But most foreign filesystems are more esoteric. Even VFAT's successor on Windows, NTFS, doesn't enjoy the same ubiquitous compatibility, even with the open source `ntfs-3g` implementation, and it's the same for Apple-native macOS filesystems. For many years, this was HFS+, which did enjoy a certain level of compatibility on Linux systems. But then Apple went and replaced it all with APFS, which was designed for privacy and the solid state storage on both Apple's Macs and iOS devices.

APFS uses a GPT partitioning scheme with volume containers, often encrypted, that operate a little like LVM on Linux, which makes it difficult to access from Linux. But thanks to the brilliant `apfs-fuse` project, it can be done, and it works remarkably well. Like any Filesystem in USErspace (FUSE), it can be built and run locally, mounting an APFS device to a local directory with an appropriately furnished `mount` command, for example:

```
apfs-fuse -o allow_other ⮐
  /dev/sdb2 /mnt/macos
```

There are arguments for mounting specific containers and mounting a software encrypted volume, plus debug options for troubleshooting and for ignoring unknown chunks of data. There's support for macOS and iOS, extended attributes, symlinks, and direct mounting of transparent decompression for zlib and LZVN. As brilliant as all this is, it can't write files or access drives encrypted with Apple's T2 chip. Regardless of this limitation, other operating systems just can't get close to this cross-filesystem compatibility. It's great to see that this vital Linux capability is keeping up with the times.



```
❯ sudo /home/graham/bin/apfs-fuse -o allow_other /dev/sdb2 /mnt/macos
[sudo] password for graham:
❯ ls /mnt/macos
private-dir  root
❯ ls /mnt/macos/root
 Applications  'Incompatible Software'   mnt       sw       usr
 cores          Library                  opt       System   Volumes
 home          'macOS Install Data'      private   Users
❯ ls /mnt/macos/root/Users
admin  catherine  graham  Shared
❯ ls /mnt/macos/root/Applications
'Ableton Live 11 Suite.app'                        MultiBeast.app
'Adobe Photoshop Lightroom 4.app'                 'Mumble 2.app'
 Ardour5.app                                      'Native Access.app'
 Arduino.app                                      'Native Instruments'
 Arturia                                           Novabench.app
 Audacity.app                                      Numbers.app
 Auracle.app                                      'NW1 Editor.app'
'BIAS AMP.app'                                    'OnScreen Control.app'
 calibre.app                                       Pages.app
 chipsounds.app                                   'Pianoteq 4'
'Chrome Remote Desktop Host Uninstaller.app'      'Pianoteq 6'
 Chrysalis.app                                     Pocket.app
 Clover.app                                        Processing.app
'Clover Configurator.app'                         'Python 2.7'
 CPU-i.app                                        'Python 3.1'
 Ctrlr.app                                         QtPass.app
 darktable.app                                     Rack.app
```

`apfs-fuse` lets you access files on Apple-formatted storage from your trusty Linux machine.

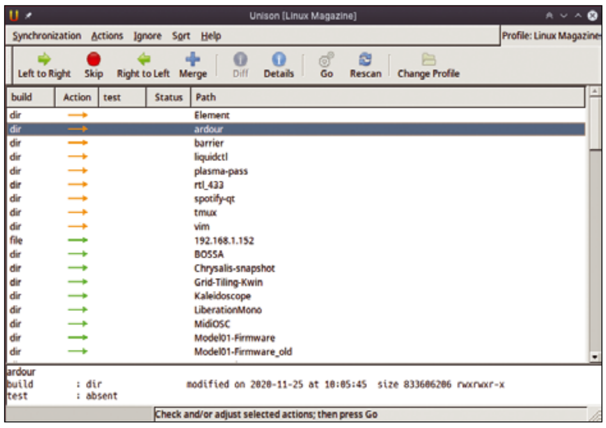
Project Website
https://github.com/sgan81/apfs-fuse

File sync

Unison

One of the most widely used system administration tools on Linux is the humble `rsync`, a command that's brilliant at replicating the contents of a directory, usually across a network, to a different location. What makes `rsync` so effective and better than a normal copy is that it can do this with certainty, thanks to hashing. `rsync` can do this by only copying the differences between an older version of a file and a newer version of a file that already exists at the destination. But what `rsync` isn't very good at is bidirectional synchronization. This is a common requirement when you're working on the contents of a synchronized directory in more than one location, such as the same project on both PC and laptop, or at home and at work. Unison has been solving this particular problem perfectly for more than 10 years.

Unison is like `rsync` for bidirectional synchronization. At its simplest, you can execute the `unison` command followed by the two locations where you wish to sync the contents. Remote directories are accessed over a direct socket or using SSH with your local SSH configuration for passwordless connections. Synchronization works best when both locations start out empty, avoiding what can be a lengthy analysis stage. After a short period of consolidation, you can begin to create files in either location to be duplicated in the other. With the Unison daemon running in the background, the two locations will remain in sync, with updates copied from both sides whenever their respective directories change. If the same file has changes at the same time on both sides and a conflict is detected, you're asked which file should take precedence, just as you might when using `git`. There's even a rather ancient GTK-based GUI, if you'd rather not use the command line. This allows you to easily see the status of your shared directories and also better navigate conflicts by visualizing changes and manually merging differences.



```
build   Action   test   Status   Path
dir     →                        Element
dir     →                        ardour
dir     →                        barrier
dir     →                        liquidctl
dir     →                        plasma-pass
dir     →                        rtl_433
dir     →                        spotify-qt
dir     →                        tmux
dir     →                        vim
file    →                        192.168.1.152
dir     →                        BOSSA
dir     →                        Chrysalis-snapshot
dir     →                        Grid-Tiling-Kwin
dir     →                        Kaleidoscope
dir     →                        LiberationMono
dir     →                        MidiOSC
dir     →                        Model01-Firmware
dir     →                        Model01-Firmware_old
ardour
build       : dir          modified on 2020-11-25 at 10:05:45  size 833606206 rwxrwxr-x
test        : absent
Check and/or adjust selected actions; then press Go
```

Unison can work transparently in the background, on the command line, and via a simple but effective GUI.

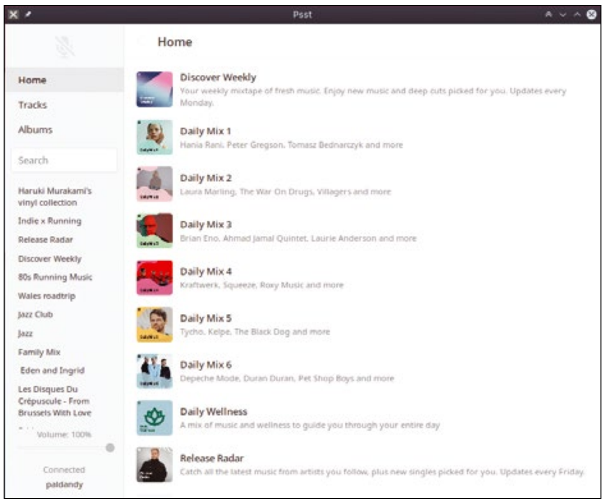
Project Website
https://github.com/bcpierce00/unison/

Spotify client

Psst

Despite the negative commentary that the official Spotify client seems to attract, it's generally a solid piece of software. It mimics the user interface of its Windows and macOS cousins. While it always lags behind Windows and macOS in new features, Spotify does allow you to explore and play your Spotify music subscription from the Linux desktop. However, the Spotify project also seldom responds to comments or feature requests, in addition to being in a perpetual state of beta development. Oh, and Spotify is not open source. This is why third-party unofficial clients are so exciting. They're able to capitalize on Spotify's lack of official development and responsiveness to create clients that better integrate with your desktop, run from the command line, or use fewer system resources because there are now a few of those resources. Using fewer system resources drives Psst development, which has resulted in a rather brilliant and low-resource graphical client that proudly boasts its speed comes "without Electron."

Psst has been developed using Rust and uses its own user interface library. On Linux, the library can use either a GTK or pure X11 back end (with Wayland under development), keeping the entire package very minimal and dependency free. The same is true of the GUI itself, which has none of the decoration you might be used to on Gnome or KDE. The window has no ornamentation and contains only the same four sections of the UI you see in the original client. There's a search pane on the left, complete with quick links to your playlists. On the right is the area used to explore new music and your own collection, while under that is the playback section with transport controls, details on the current track, and timing information. Finally, volume control is to the left. The application is still under rapid development, and there are many features missing – especially if you need to modify a playlist. Anything missing can of course be done through another client, leaving Psst to do what it's good at – low resource playback.



Home

Discover Weekly
Your weekly mixtape of fresh music. Enjoy new music and deep cuts picked for you. Updates every Monday.

Daily Mix 1
Hania Rani, Peter Gregson, Tomasz Bednarczyk and more

Daily Mix 2
Laura Marling, The War On Drugs, Villagers and more

Daily Mix 3
Brian Eno, Ahmad Jamal Quintet, Laurie Anderson and more

Daily Mix 4
Kraftwerk, Squeeze, Roxy Music and more

Daily Mix 5
Tycho, Kelpe, The Black Dog and more

Daily Mix 6
Depeche Mode, Duran Duran, Pet Shop Boys and more

Daily Wellness
A mix of music and wellness to guide you through your entire day

Release Radar
Catch all the latest music from artists you follow, plus new singles picked for you. Updates every Friday.

Avoid Spotify podcasts by using an unofficial third-party client such as Psst.

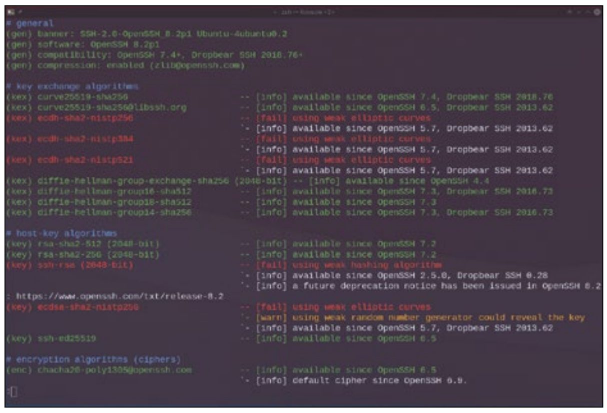
Project Website
https://github.com/jpochyla/psst

Security tool

ssh-audit

On a recent podcast episode, we ran an informal survey of what most listeners thought was their most essential open source tool. The overwhelming winner was the humble secure shell, SSH, the remote command-line client and server that many have been using for decades. SSH is super secure and easy to use, often configured by a distribution at install time, which enables even beginners to start using their machines remotely. But SSH can also be a victim of its own success. Its ubiquity makes it a common attack vector, as anyone who runs an SSH server and happens to look at the logs will attest. Your logs will be full of automated attempts to guess your username and password combinations, a situation not helped when even distributions such as older versions of Raspberry Pi OS used the same default password and username combinations.

All of this can be mitigated with a properly configured SSH server, swapping passwords for encryption keys, for instance, and changing the default ports. But unless you're an expert, it can be difficult to know whether you've exchanged one set of problems with another, which is where the aptly named `ssh-audit` can help. It's a command that can deeply check your SSH configuration by analyzing the connection rather than parsing the configuration file. This allows you to detect what is actually happening, as well as passively monitor for client and server-side vulnerabilities on incoming and outgoing connections, even when you don't control the server. The output is verbose and will detail which algorithms are being used, which should be disabled, and which removed, alongside which keys are detected. It's a glut of information but an incredibly valuable insight into the state of the connections on which we all rely.



```
# general
(gen) banner: SSH-2.0-OpenSSH_8.2p1 Ubuntu-4ubuntu0.2
(gen) software: OpenSSH 8.2p1
(gen) compatibility: OpenSSH 7.4+, Dropbear SSH 2018.76+
(gen) compression: enabled (zlib@openssh.com)

# key exchange algorithms
(kex) curve25519-sha256                     -- [info] available since OpenSSH 7.4, Dropbear SSH 2018.76
(kex) curve25519-sha256@libssh.org          -- [info] available since OpenSSH 6.5, Dropbear SSH 2013.62
(kex) ecdh-sha2-nistp256                    -- [fail] using weak elliptic curves
                                            `- [info] available since OpenSSH 5.7, Dropbear SSH 2013.62
(kex) ecdh-sha2-nistp384                    -- [fail] using weak elliptic curves
                                            `- [info] available since OpenSSH 5.7, Dropbear SSH 2013.62
(kex) ecdh-sha2-nistp521                    -- [fail] using weak elliptic curves
                                            `- [info] available since OpenSSH 5.7, Dropbear SSH 2013.62
(kex) diffie-hellman-group-exchange-sha256 (2048-bit) -- [info] available since OpenSSH 4.4
(kex) diffie-hellman-group16-sha512         -- [info] available since OpenSSH 7.3, Dropbear SSH 2016.73
(kex) diffie-hellman-group18-sha512         -- [info] available since OpenSSH 7.3
(kex) diffie-hellman-group14-sha256         -- [info] available since OpenSSH 7.3, Dropbear SSH 2016.73

# host-key algorithms
(key) rsa-sha2-512 (2048-bit)               -- [info] available since OpenSSH 7.2
(key) rsa-sha2-256 (2048-bit)               -- [info] available since OpenSSH 7.2
(key) ssh-rsa (2048-bit)                    -- [fail] using weak hashing algorithm
                                            `- [info] available since OpenSSH 2.5.0, Dropbear SSH 0.28
                                            `- [info] a future deprecation notice has been issued in OpenSSH 8.2
: https://www.openssh.com/txt/release-8.2
(key) ecdsa-sha2-nistp256                   -- [fail] using weak elliptic curves
                                            `- [warn] using weak random number generator could reveal the key
                                            `- [info] available since OpenSSH 5.7, Dropbear SSH 2013.62
(key) ssh-ed25519                           -- [info] available since OpenSSH 6.5

# encryption algorithms (ciphers)
(enc) chacha20-poly1305@openssh.com         -- [info] available since OpenSSH 6.5
                                            `- [info] default cipher since OpenSSH 6.9.
:
```

While `ssh-audit` goes into a lot of detail, you can find an excellent hardening guide on their website if you need advice on how to improve your SSH configuration.

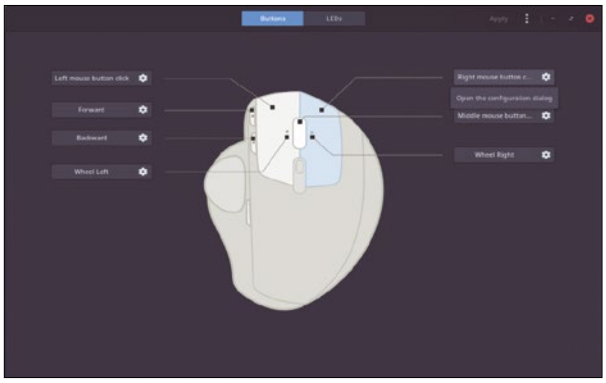
Project Website
https://github.com/jtesta/ssh-audit

Mouse configuration

Piper

This is really two FOSSPicks in one. The first part is the `ratbagd` DBus daemon, a background service that helps expose configuration options for specific input devices that would otherwise require a manufacturer-provided driver. It specializes in gaming mice, and much like OpenRGB, `ratbagd` developers have painstakingly decoded the various protocols required to unlock undocumented and proprietary features outside of their standard driver support, scaling modes, and unused buttons._ There's support for more than 50 devices, including a majority of devices from Logitech and a few from SteelSeries, among others. There is a command-line tool that can help you configure those devices, but for best results, you'll want to use our second FOSSPick for this entry, the Piper GUI.

Piper is a modern GTK-based GUI for configuring the options exposed by `ratbagd`. Its features will depend on your hardware's capabilities, but broadly, it lets you change tracking resolutions, reassign buttons and scroll wheels, and change the colors of any LEDs. You switch between the three main options using buttons at the top of the window, and the main view updates depending on which mode is selected. The main view will helpfully show you a diagrammatic representation of your mouse, complete with annotations for each of its configurable parts. If a button is assignable, for example, clicking its annotation will open up a list of actions it can be assigned, from the normal left, right, and middle mouse clicks to ratchet modes, sensitivity, and even your own keyboard shortcuts. It's a powerful tool for gaming, where a single button can now send a flurry of keys, but it's also useful on the desktop where shortcuts can be set for switching desktops, copying and pasting, and launching a terminal.



Piper is just as brilliant at configuring gaming mice as it is Logitech's more desktop-oriented trackballs.

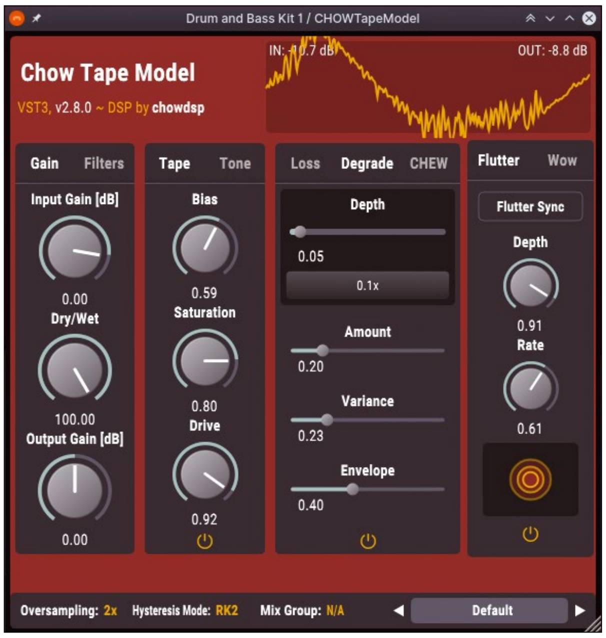
Project Website
https://github.com/libratbag/piper

Tape emulation

ChowTapeModel

ChowTapeModel is one of several exceptional audio effect plugins developed by Chowdhury DSP and released for free under an open source licence. Alongside ChowTapeModel, Chowdhury DSP offers a delay effect (ChowMatrix) and virtual guitar effects pedal (ChowCentaur). ChowTapeModel does something that would have been inconceivable not so long ago when many of us were desperately trying to leave the noise and distortion of tape recordings behind for the crystalline perfection of digital reproduction. During this transition, it became apparent that while digital audio was unmistakably more accurate, with more dynamic range, less noise, and no variance, something had been lost in translation: a certain kind of analog warmth and imperfection. ChowTapeModel tries to add this analog warmth and imperfection to your digital recordings.

The algorithm behind the effect has been carefully engineered to algorithmically model the characteristics of a real tape machine, the Sony TC-260 from the 1970s. The plugin features many controls that reflect the various configuration options for the original machine, such as tape head alignment and the speed of the tape itself. There are also controls for the amount of saturation and distortion you want to add. These equate to changes in tone and slight distortion, as well as wow and flutter to emulate slight variations in the virtual tape playback. Very subtle adjustments to these values can genuinely turn your digital recording into something that might have originally been recorded on tape, with the sound often becoming smoother, more rounded, and less separated. Different configurations sound better for different types of music, and you can save presets to quickly switch between them. It works well on rock and pop music, but it can also improve your own recordings by acting as a mastering effect to glue a mix together.



IN: -10.7 dB
OUT: -8.8 dB
Input Gain [dB] 0.00
Dry/Wet 100.00
Output Gain [dB] 0.00
Bias 0.59
Saturation 0.80
Drive 0.92
Depth 0.05
Amount 0.20
Variance 0.23
Envelope 0.40
Depth 0.91
Rate 0.61
Oversampling: 2x
Hysteresis Mode: RK2
Mix Group: N/A

This tape emulation effect works well for rock and pop, but it can also improve simple recordings such as voice for a podcast.

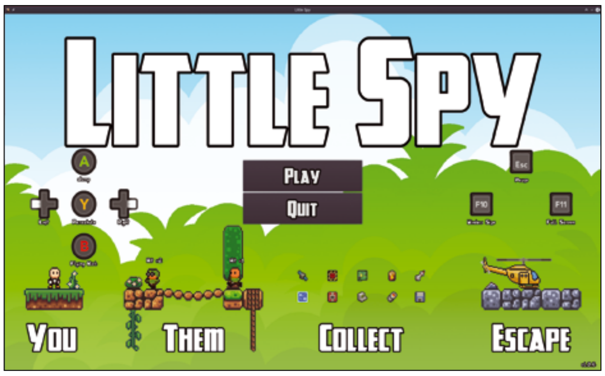
Project Website
https://chowdsp.com/

Retro platformer

Little Spy

Little Spy is a beautifully executed 2D platform game with nostalgia-laden graphics and an easy, accessible style of gameplay. It starts with a loading screen that both looks good and ably introduces the core game mechanics: Use cursor keys to control your character; press space to jump; and navigate between the floating platforms and the baddies, collecting things, before reaching the helicopter launch pad. It all feels slightly reminiscent of 8-bit Sensible Software games in the 1980s, especially when the baddies parachute in like your companions in Cannon Fodder. You can also parachute, dash, attack enemies, and double jump, all of which needs to be mastered before you can complete the game.

The game sensibly starts with a simple level to get you used to the controls, with lots of small platforms, plenty of objects to collect, and baddies to avoid. It plays a little like the ancient Bomb Jack, although the graphics here are a lot better. The game's entire production is all high quality, from the graphics and sound to the onboarding experience, and ultimately, the satisfying gameplay. Each level gets progressively more challenging and unforgiving. You need to collect a certain number of items, and the action will start to extend from one screen to several, all smoothly scrolled between as your little character jumps their way to the helicopter. There's even a story to follow – you are an agent tasked with retrieving stolen intelligence and technology. This is all a huge credit to the game's developer, Martin Wimpress, who built the game in two weeks using Godot for the My First Game Jam. And yes, that's the same Martin Wimpress who is responsible for Ubuntu MATE!



Martin Wimpress has created a brilliant 2D platformer in very little time. Here's hoping he tries something more substantial!

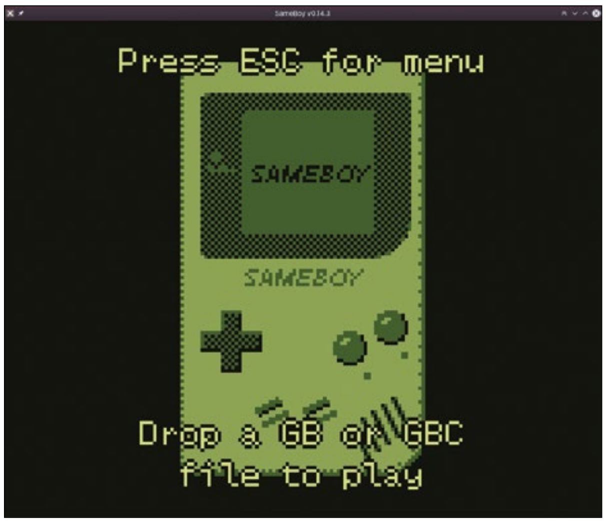
Project Website
https://github.com/wimpysworld/little-spy

Game Boy emulator

SameBoy

It's hard to believe that Nintendo's Game Boy is now over 30 years old. Like the Sony Walkman, it was a device that defined a generation, letting them play Tetris, Super Mario Land, and The Legend of Zelda: Link's Awakening wherever they wanted. It also paved the way for modern hand-held gaming, from Pokémon on a smartphone to the success of the Nintendo Switch and even Valve's hugely hyped Linux handheld, the Steam Deck. What's even more remarkable is that the CPU in the Game Boy was based on the humble Z80, with only 8KiB of internal memory expanded by the ROM cartridges and a screen resolution of 160x144 pixels. Despite this, the playability of many of its games still stands up, all these years later, which is of course why we have emulation.

SameBoy is a Game Boy emulator that's easy to use and presents all the options you'll ever need. It's also authentic, sticking with the super low resolution of the original display, which can look comically pixelated on today's screens. However, SameBoy can also emulate the later Game Boy Color console, which modernizes things a little and requires very few resources from a modern system. It's also easy to use. You simply drag and drop the game you want to play into the main window and SameBoy will react just like the original console. You can change the graphical scaling algorithms, the keyboard and joystick input assignments, and the audio quality. There's even an option to mix background interference into the audio, just like on the original, and the entire experience feels very accurate. If you've not played with a Game Boy for a couple of decades, SameBoy is the perfect excuse to give it another try.



SameBoy features a rollback mode, letting you replay the last several seconds to avoid making a mistake.

Project Website
https://sameboy.github.io/

GARDEN TRICKS
USE LONG-RANGE RADIO TO MONITOR YOUR PLANTS
BUILD AN IRC BOT
LINUX MAGAZINE
ISSUE 239 – OCTOBER 2020
BUILD AN IRC BOT
Customize a GTK3 Theme
DISPATCHER SCRIPTS
Automate NetworkManager for seamless location changes
20TH ANNIVERSARY ISSUE!
ALL 239 ISSUES ON A SEARCHABLE DISC!
HUGE SAVINGS! £29.90 VALUE!
20 YEARS ARCHIVE DVD
ISSUE 240 – NOVEMBER 2020
20 YEARS
IT'S ALIVE!
Breathe life into your coding experiments by creating your own simple OS
Graphing the Pandemic
GO PROGRAMMING
Search for files on Google Drive
3D PRINTING
ISSUE 242 – JANUARY 2021
3D PRINTING
From first steps to a finished creation
MOFO Linux
Steer around censorship with this privacy-focused distro
Nyttig
Share files and stream Internet radio from a Rasp Pi
Netdata
Zero configuration monitoring tool
Nerf Dart Game
Score points for your next neighborhood party
Costs and Benefits
Choose the best alternative with the
LINUXVOICE
Stacer: Clean up your system
maddog: Morality and licensing
JessyInk: Presentations in Inkscape
Mistborn
All-in-one tool for protecting your
JAM.PY
BUILD A WEB FRONT END FOR YOUR DATABASE
SECURE YOUR SYSTEM
ISSUE 241 – DECEMBER 2020
SECURE YOUR SYSTEM
Expert tips for protecting your Linux
Clonezilla SE
Mass cloning without the mess
Managing Servers with Cockpit
RebornOS
Arch for the slightly less geeky
Christmas Tinkering
Build a holiday music box
Git Utilities
Cool tools for extending Git version control
Fun with Go
Display song line by line
FIREFOX
Enabling hardware acceleration
Exploring the iNet Wireless Daemon
ISSUE 243 – FEBRUARY 2021
Intel's wireless daemon ushers in a new world for Linux networks
MS Visual Studio on Linux
You can even develop code for a Rasp Pi
ExifTool
Analyze file metadata
SAFE MESSAGING
PASS YOUR PUBLIC KEY USING DNS
STREAM PROCESSING
ISSUE 244 – MARCH 2021
STREAM PROCESSING
New coding techniques for the data revolution
Make It Mandatory
Requiring two-factor authentication at login
2020: The Year in Review
config Deep Dive
Exploring the powerful system that configures the Linux kernel
Raspberry Pi 4 on 8GB RAM
Do more and go faster with the souped-up 8GB version
Gnuplot
Draw illuminating graphs with only a few lines of code
Checksecurity
Lock down your system with automated security checks
IOBROKER
Home automation with a Rasp Pi
CHOOSE A SHELL
ISSUE 245 – APRIL 2021
CHOOSE A SHELL
We compare Bash, Z
ELK Stack
Keep watch over your log
Rendering Images as T
Boop
Keep it local with this sc
Xubuntu on a De
Refurbish old lapto
LINUXV
The rise of the small Interne
Font Manager: Keep your ty
maddog: The benefits of a F system
WWW.LINUX-MAGAZ
MakerSpace
Video Calls on a Rasp Pi
ISSUE 246 – MAY 2021
Faster Startup
Tune your system for speedier boot
MakerSpace
Hands-On Guide to the GP
ISSUE 247 – JUNE 2021
Post-Quant Encrypti
Will privacy survive quantum computers?
JSON Deep Dive
LINUXVOICE
maddog explores
MakerSpace
Retro Gaming with Lakka
ISSUE 248 – JULY 2021
Brain Tools
Stay sharp with these free tools for study and play
COVID Crunch
Explore pandemic data with Gawk and Bash
Z-Wave
Raspberry Pi as a home assistant
LINUXVOICE
Voice and video chats with Tox
Zotero personal research assistant
10 FREE SOFTWARE GEMS!
WWW.LINUX-MAGAZINE.COM
FREE DVD
Double-Sided DVD INSIDE!

A kanban-based workflow management system

Visual Aid

Wekan lets self-organzing teams manage a project's workflow by tracking task ownership and progress visually.

BY MARCO FIORETTI

Kanban, the Japanese word for signboard, is a workflow management system for organizing and tracking work. Kanban boards show team members the status of each task at any given time, including ownership and task order, with the aim of limiting the number of tasks "in progress" at any given time.

First developed in the 1980s to improve the efficiency of Toyota's assembly lines [1], kanban has proven its usefulness as a general-purpose organization tool. Today, people use kanban for everything from simple personal to-do lists to coordinating teams that produce complex goods or services.

The original kanban boards were physical boards attached to factory or office walls. While many people still use physical kanban boards (even if it's just on paper), a variety of kanban board software tools exist (see also the "Scrum vs. Kanban" box). Before using any type of kanban system, I recommend reading why some people still choose physical kanban boards [3].

In this tutorial, I take a look at Wekan [4], an open source, MIT-licensed, web-based kanban system. Written with the Meteor Javascript framework [5] and using a MongoDB database, Wekan runs on any web browser. I'll show you how to install and use Wekan using general kanban principles.

Installation

You can download the Wekan source code, as well as Snap packages for several Linux distributions, from the Wekan website [4]. Alternatively, you could use Wekan as an application inside the open source Sandstorm platform for self-hosting web apps [6].

Although the process is not complicated, installing Wekan and all its dependencies from scratch on a dedicated server can be a lengthy process with a lot of details. I don't have the space to cover installing Wekan from source here, but an online tutorial written for CentOS 7 can point you in the right direction [7]. Though not completely up to date, the CentOS tutorial is very detailed and an excellent addition to the official Wekan documentation, regardless of your distribution.

If you don't need to install from source code, the Snap packages are much easier to use. On Ubuntu, you can run

```
sudo snap install wekan
sudo snap set wekan port="3500"
sudo snap enable wekan
```

at the prompt to install Wekan v5.37.0 and tell Wekan to answer connection requests on IP port number 3500. After installation, if you want to connect to Wekan from the same host, you must point your browser to the location *http://localhost:$IP_PORT*, where `$IP_PORT` is the parameter value you set in the second command above. I chose port 3500, but any port will do as long as it is not already used by another service on the same host. The final command above tells Linux to start Wekan automatically at bootup.

Scrum vs. Kanban

In the past few decades, the need to organize work with computers, possibly online, has resulted in a wide range of organizational tools. You may be familiar with Scrum [2], a work organization method that looks similar to kanban. Both tools have the same high-level purpose. There are, however, several important differences between Scrum and kanban boards. To begin with, while both methods are meant to support self-organizing work teams, Scrum uses formal roles, whereas kanban doesn't use them at all. Furthermore, a kanban board is always "active," as long as the project is produced or maintained. Scrum boards, instead, are created only when needed, to perform specific tasks in "sprints" that have precise, often quite pressing deadlines.

For this tutorial, I ran Wekan on my main Linux computer. If you need your installation to be accessible from the Internet, Wekan will also need to know the URL it should answer to, and you will have to open the corresponding IP port in the server firewall (please consult your firewall's documentation for instructions). The URL can be set with just one more `snap` command:

```
sudo snap set wekan root-url=⮐
  "https://wekan.example.com"
```

Initial Configuration

When you first connect to your Wekan installation, you must create at least one user account. Next, you need to configure several system-wide features, including email configuration, which allows Wekan to notify users whenever something happens associated with their accounts (see the Wekan wiki [8] for instructions).

After configuring email, you may want to customize parameters found in the Admin Panel's *Registration*, *Accounts*, and *Layout* tabs (Figure 1). You can allow (or forbid) self-registration. Existing users can also be given permission to change their personal Wekan account's primary data (i.e, username and email). In the *Layout* tab, you can define a clickable logo with a custom URL, as well as a few other cosmetic details.

Potentially very interesting, *Global Webhooks* lets you define site-wide hooks that automatically notify external applications of something that happened inside Wekan or load data from external applications. You can also define board-specific webhooks. Unfortunately, at the time of writing, this feature is not adequately documented to be easily usable without spending significant time studying the code or hunting for answers in the Wekan support forum. However, webhooks may not be needed at all by the majority of projects that use Wekan.

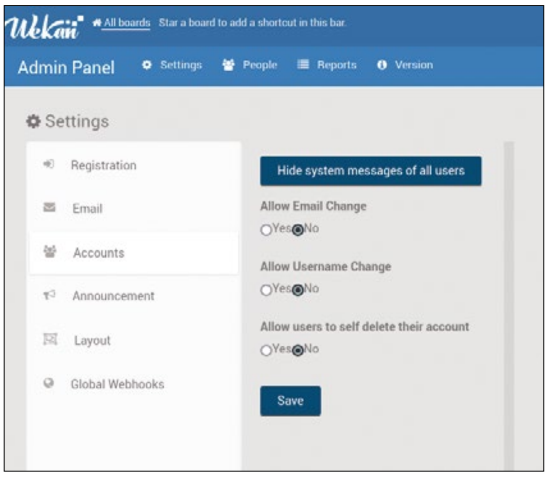
A single Wekan installation supports multiple users who can be grouped into separate organizations, each composed of different teams (Figure 2). Figure 3 shows configuration options for an organization, and Figure 4 shows the team options.

Columns, Cards, and WIPs

Regardless of the type of project, all kanban boards should support, at a minimum, three types of components: columns, cards, and Work In Progress (WIP) limits.

A column, or a list, in Wekan lingo, defines a specific activity or phase of the entire workflow. For instance, a restaurant may have columns that indicate each client's current status: a list for clients waiting for their orders and another list for clients being actively served. The simplest possible workflow in most cases probably contains only three lists: To-Do, In Progress, and Finished. However, every project can create as many lists as needed and name them in whatever way makes their function clear to the whole team.

A card contains a single task managed on a kanban board. Each card moves through the columns in the workflow as a given tasks progresses toward completion. In most kanban systems, including Wekan, you can share the workload by explicitly assigning team members to specific cards. To make things even clearer, Wekan lets you attach colored labels to cards, allowing you to group or sort and filter the cards by title, creation date, or due date.

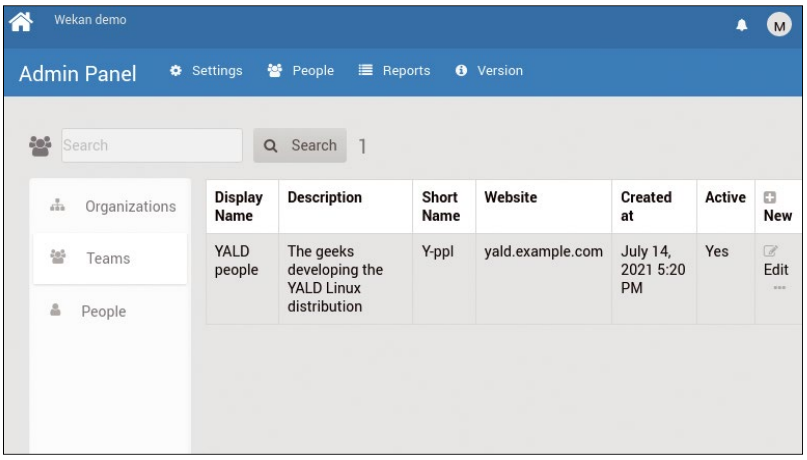


Allow Email Change
Yes [✓ No]

Allow Username Change
Yes [✓ No]

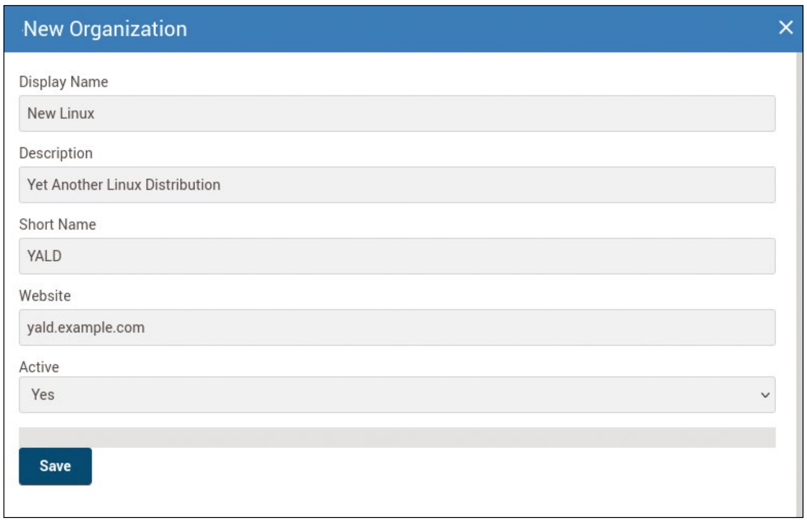
Allow users to self delete their account
Yes [✓ No]

Figure 1: The Wekan Admin Panel settings are simple but functional.



Display Name	Description	Short Name	Website	Created at	Active	New
YALD people	The geeks developing the YALD Linux distribution	Y-ppl	yald.example.com	July 14, 2021 5:20 PM	Yes	Edit

Figure 2: One kanban to rule them all: Wekan supports independent organizations consisting of teams and individual users.



Display Name
New Linux

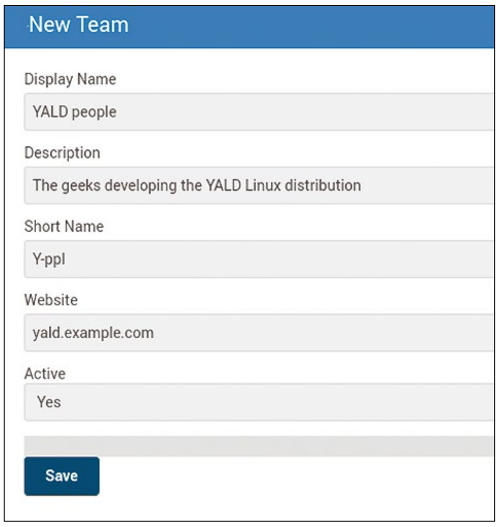
Description
Yet Another Linux Distribution

Short Name
YALD

Website
yald.example.com

Active
Yes

Figure 3: The New Organization dialog.


New Team

Display Name: YALD people

Description: The geeks developing the YALD Linux distribution

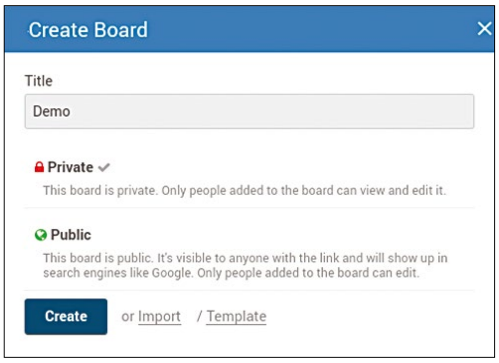
Short Name: Y-ppl

Website: yald.example.com

Active: Yes

Save

Figure 4: The New Team dialog.


Create Board

Title: Demo

Private ✓
This board is private. Only people added to the board can view and edit it.

Public
This board is public. It's visible to anyone with the link and will show up in search engines like Google. Only people added to the board can edit.

Create or Import / Template

Figure 5: Wekan supports both public and private kanban boards.

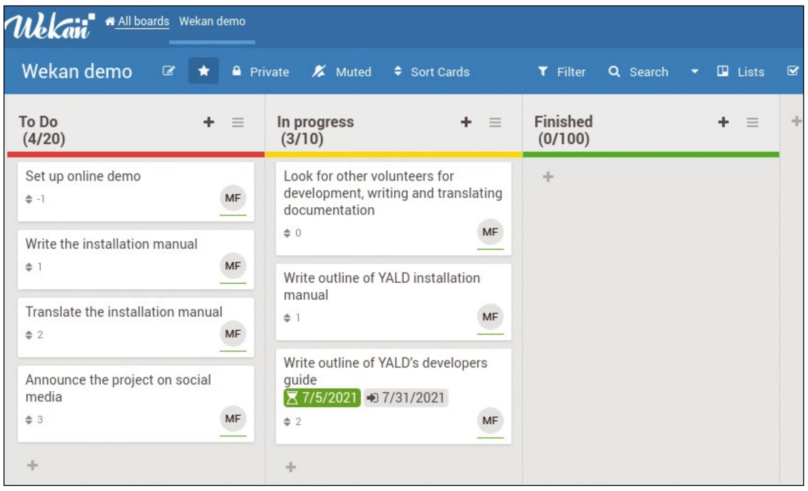
The WIP limit defines the maximum number of cards that can be in one column at any given time, in other words, the maximum number of tasks that the team allows itself to have in that particular state at any given moment. When the number of cards in a column exceeds the WIP limit, anyone responsible for any of the cards must focus to complete those tasks and move them to the next column before other cards begin to pile up in that column. In my opinion, the WIP limit is the kanban method's core strength. Make the WIP limit an adequate value for your team and project, which of course only you can estimate, and life will be good as long as everyone respects the limit. If the WIP limit is ignored, problems could arise. A WIP limit set too low results in unrealistic expectations and unnecessary stress. In contrast, a value set too high, even in just one column, can slow down the entire workflow represented on the board.

In Wekan, you can create and use several boards simultaneously (each containing a separate project) and set each of them as public or private (Figure 5). Once you have created a board, click on the plus sign at the bottom to add as many lists (columns) as you want to the board. After attaching a list to the board, you can (and should) assign a separate color to the list, but most importantly, you should define a list's WIP limit. Each list can have a different WIP value, set as either a hard or soft limit. A soft limit means the limit can be exceeded, but a warning will be given. After you have created all your lists, you can rearrange them by dragging and dropping them inside the board. You can also rename a list by selecting it and then clicking again on its name.

View Modes

Each Wekan board has three View Modes: Lists, Calendar View, and Swimlanes. The default mode, Lists (shown in Figure 6), shows all the lists side by side, each with its own cards. Calendar View, which I'll cover later, shows the start and due dates for all active cards.

The Swimlanes view may be unnecessary for single individuals or simple projects, but this view is essential in almost all other cases. *Kanbanize.com* defines swimlanes as "horizontal lines that split a kanban board into sections, [in order to] visually separate different work types on the same board and organize homogenous tasks together" [9]. In other words, if your organization has different teams that work mostly autonomously, with the same workflow but without continuous direct interactions, you can still place them on the same board by putting

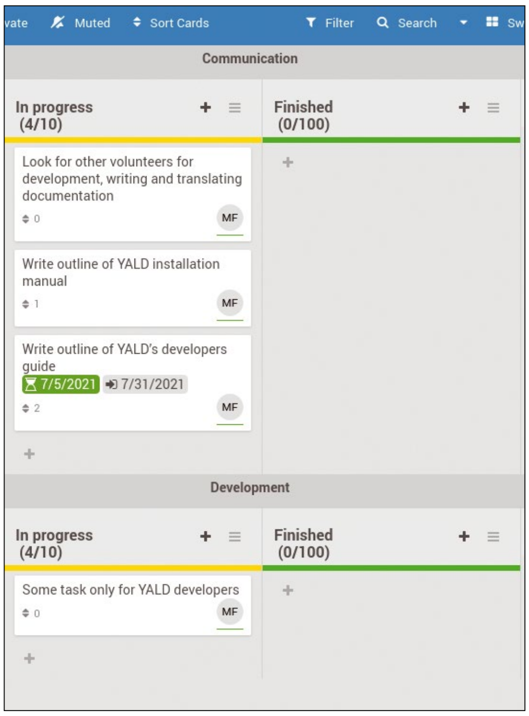


To Do (4/20)
- Set up online demo — -1 — MF
- Write the installation manual — 1 — MF
- Translate the installation manual — 2 — MF
- Announce the project on social media — 3 — MF

In progress (3/10)
- Look for other volunteers for development, writing and translating documentation — 0 — MF
- Write outline of YALD installation manual — 1 — MF
- Write outline of YALD's developers guide — 7/5/2021 7/31/2021 — 2 — MF

Finished (0/100)

Figure 6: The default Lists view in Wekan: Each project phase has its own list (column) with individual tasks (cards), along with a WIP limit displayed below the list name.



Communication

In progress (4/10)
- Look for other volunteers for development, writing and translating documentation — 0 — MF
- Write outline of YALD installation manual — 1 — MF
- Write outline of YALD's developers guide — 7/5/2021 7/31/2021 — 2 — MF

Finished (0/100)

Development

In progress (4/10)
- Some task only for YALD developers — 0 — MF

Finished (0/100)

Figure 7: Wekan swimlanes (in this example, ***Communication*** and ***Development***) keep the work of independent teams separate but still visible at a glance.

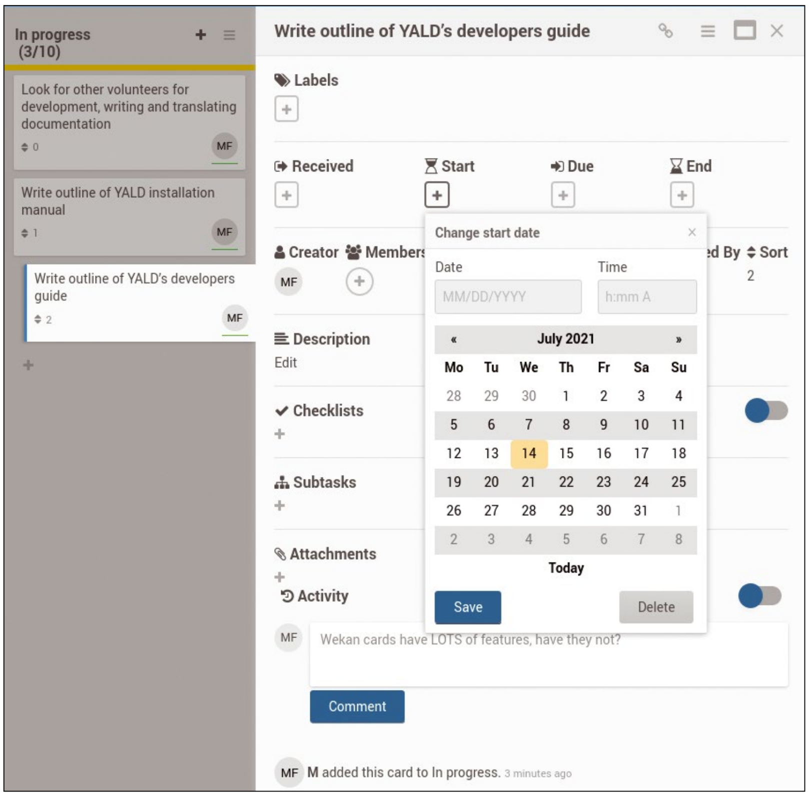
each team in a separate swimlane. In that way, their activities (i.e., the workload and responsibilities for each member of each team) remain as visually separated as possible. Figure 7 shows how this looks in practice inside Wekan, with swimlanes for communication and development activities.

Cards, at Last!

Wekan offers so many configuration options for cards, that at first glance the options may seem overwhelming or unnecessary (Figure 8). The *Labels* option lets users group and filter cards. The *Received* option lets you assign a start, due, and end date for each card, which you can see in the Calendar View mentioned earlier (Figure 9). You can also add comments to each card at the bottom.

Most crucial, the *Description* option lets you specify without ambiguities the nature of the task and when it can be considered finished. Both *Checklists*, which lets you attach one or more checklists to each card, and *Activity*, which lets you measure how much time is spent on each task, are useful but probably not essential.

A couple of options should be used with caution. The *Subtasks* option, which lets you assign subtasks to a task, should be monitored closely for one simple reason: If a team finds itself relying too much on subtasks, always attaching several to each card, it may be a strong sign that the overall workflow (the number of lists and swimlanes on the kanban board) needs a serious redesign. In addition, the *Attachments* option, which lets you attach any type of file to a card, can lead to confusion if an important document is only attached to one Wekan card and then forgotten. You can also run into problems if attachments are duplicated in a document management system, such as the SeedDMS [10].



Look for other volunteers for development, writing and translating documentation

Write outline of YALD installation manual

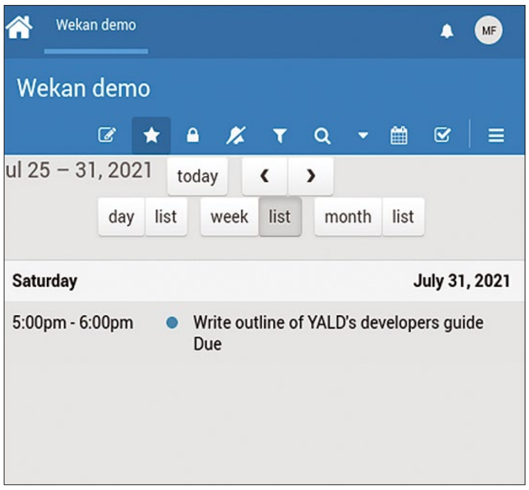
Write outline of YALD's developers guide

Wekan cards have LOTS of features, have they not?

MF M added this card to In progress. 3 minutes ago

Figure 8: Wekan cards have enough configuration options to make the most demanding project managers happy.

In the card's titlebar, you'll find a link icon and a hamburger menu. The link icon lets you link a card to other cards (similar features are available to link boards or lists) to highlight the dependencies



Saturday July 31, 2021

5:00pm - 6:00pm Write outline of YALD's developers guide Due

Figure 9: While Calendar View may be the least used view mode in Wekan, it will show you the start, end, and due dates for all active tasks.

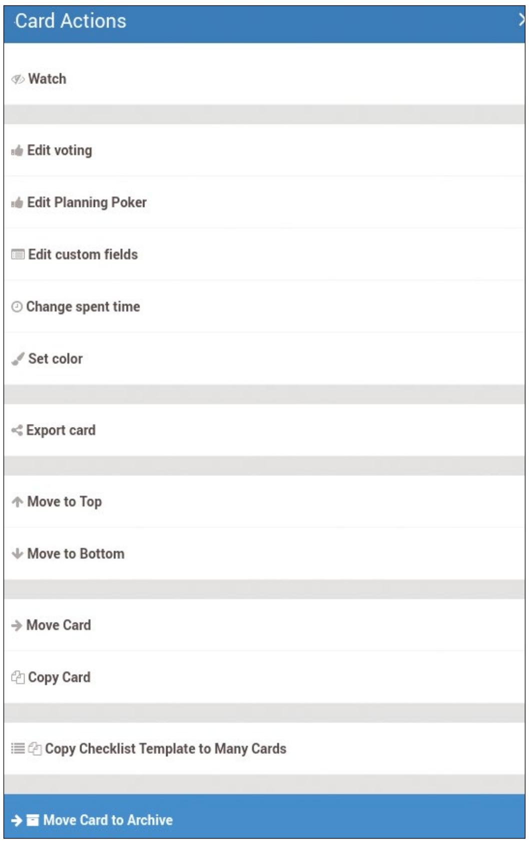


Figure 10: The Wekan Card Actions menu allows you to further customize your project tasks.

across different operations and teams. If you click on the hamburger menu in the card's titlebar, the Card Actions dialog opens (shown in Figure 10). Here you can, among other things, edit custom fields, set color, move a card around the board, use a card as a template, and eventually archive the cards.

Other Useful Features

Wekan can also generate usage reports. The most important reports are the ones that show broken (unlinked or inconsistent) cards and "orphaned" files (uploaded files that are no longer attached to existing cards) on a given board.

If you have boards from other kanban software or in plain text CSV format, you can import these boards into Wekan. To do so, click on the plus sign in the top menubar, and then click *Import* and follow the on-screen instructions.

Conclusions

Kanban boards have been around for decades without substantial changes to their original form. While they may not be the best tool for every team or project, kanban boards deliver what they promise: immediate visibility of a project's process, the responsibilities for each task, and any bottlenecks in the workflow. Kanban boards present this information in a way that is particularly well suited to self-organized teams without a boss directing tasks.

Wekan effectively implements kanban principles for workflow management. However, Wekan's effectiveness relies on all users following the rules. If all team members reliably enter all tasks, update the task statuses, and so on, Wekan will be helpful. Otherwise, it may create (or hide until it's too late) problems and frustration. If you and your team decide to use Wekan as your assistant, you must really be faithful to Wekan's kanban principles.

To find out how Wekan (and kanban principles in general) can benefit your team, give Wekan a try for a week or two on a test project. Wekan is a cool tool, as long as you use it appropriately. ■■■

Info

[1] Kanban: *https://en.wikipedia.org/wiki/Kanban*

[2] Scrum: *www.scrum.org/resources/what-is-scrum*

[3] Thoughts on paper-based kanban boards: *www.jrothman.com/mpd/project-management/2017/02/why-i-use-a-paper-kanban-board/*

[4] Wekan: *https://wekan.github.io/*

[5] Meteor: *www.meteor.com*

[6] Sandstorm: *https://sandstorm.io/*

[7] Installing Wekan from sources on CentOS 7: *www.howtoforge.com/tutorial/centos-wekan-installation/*

[8] Configuring email: *https://github.com/wekan/wekan-snap/issues/107*

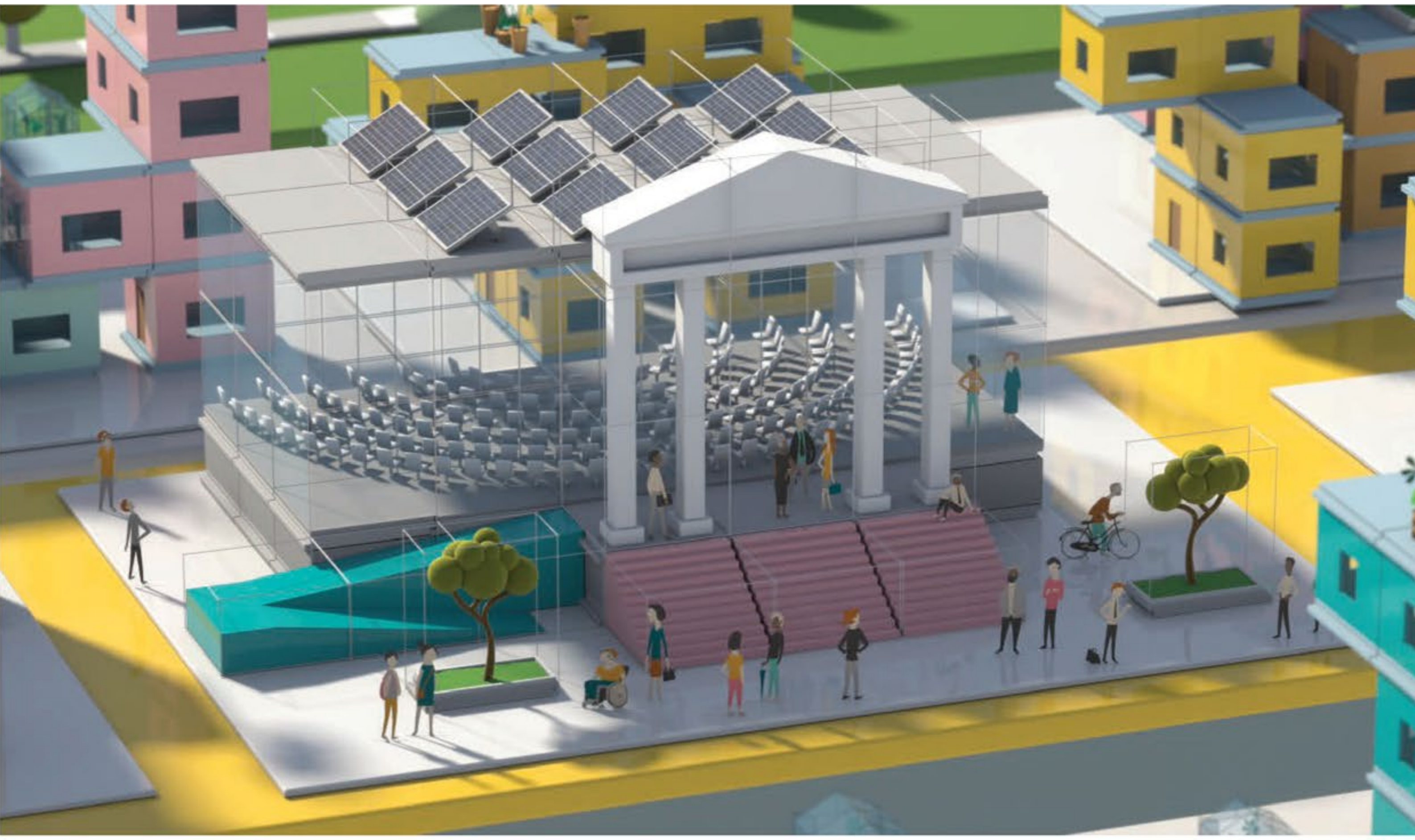
[9] Kanban swimlanes: *https://kanbanize.com/kanban-resources/kanban-software/kanban-swimlanes*

[10] "Keep Your Documents Organized with SeedDMS" by Marco Fioretti, *Linux Magazine*, issue 249, August 2021, pp. 88-93

The Author

Marco Fioretti (*http://mfioretti.com*) is a freelance author, trainer, and researcher based in Rome, Italy. He has been working with free/open source software since 1995 and on open digital standards since 2005. Marco also is a Board Member of the Free Knowledge Institute (*http://freeknowledge.eu*) and blogs about digital rights at *https://stop.zona-m.net*.

Public Money Public Code

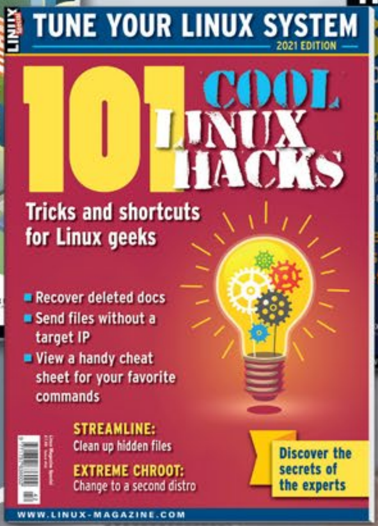


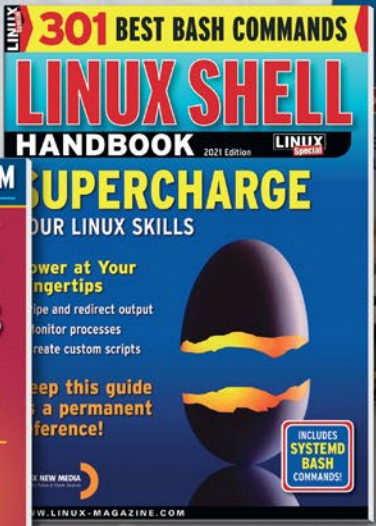
Modernising Public Infrastructure with Free Software

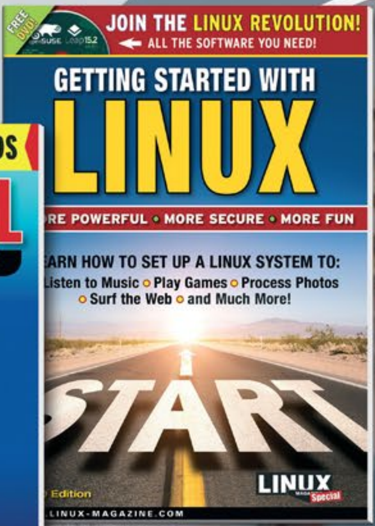
fsfe Free Software Foundation Europe

Learn More: https://publiccode.eu/

FREE DVD! LibreOffice Full Version
Become a LibreOffice Expert!
2020 Edition
LibreOffice
Dive deep into the world's greatest free office suite
Write Your Own LO Macros
Digital Signatures
Edit and Save MS Office Files
Create Professional:
Replace MS Office and Google Docs!
Includes full versions for Windows, macOS, and Linux
WWW.LINUX-MAGAZINE.COM


TUNE YOUR LINUX SYSTEM
2021 EDITION
101 COOL LINUX HACKS
Tricks and shortcuts for Linux geeks
Recover deleted docs
Send files without a target IP
View a handy cheat sheet for your favorite commands
STREAMLINE: Clean up hidden files
EXTREME CHROOT: Change to a second distro
Discover the secrets of the experts
WWW.LINUX-MAGAZINE.COM


301 BEST BASH COMMANDS
LINUX SHELL HANDBOOK
2021 Edition
SUPERCHARGE YOUR LINUX SKILLS
INCLUDES SYSTEMD BASH COMMANDS!


JOIN THE LINUX REVOLUTION!
ALL THE SOFTWARE YOU NEED!
GETTING STARTED WITH LINUX
MORE POWERFUL • MORE SECURE • MORE FUN
START

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.

#250/September 2021

Inside the Kernel

The only real way to celebrate the 30th anniversary of Linux is to write about Linux itself – not the agglomeration of software we know as a Linux distro, but the real Linux – the beating heart in the center of it all: the Linux kernel.

On the DVD: AlmaLinux Minimal 8.4 and SystemRescueCd 8.03

#249/August 2021

Turn Your Android into a Linux PC

UserLAnd lets you run Linux applications on your Android phone – all without replacing Android OS.

On the DVD: openSUSE Leap 15.3 and Kubuntu 21.04 Desktop

#248/July 2021

Brain Tools

Sometimes you want the computer to think for you, and sometimes you want the computer to make you think. This month we present a selection of free Linux tools for learning and thinking.

On the DVD: Ubuntu 21.04 and Fedora 34 Workstation

#247/June 2021

Post-Quantum Encryption

Quantum computers are still at the experimental stage, but mathematicians have already discovered some quantum-based algorithms that will demolish the best of our current encryption methods. What better time to look for quantum encryption alternatives?

On the DVD: Knoppix 9.1 and ZORIN OS 15.3 Core

#246/May 2021

Faster Startup

Weary of waiting for a login window? Your driver-drenched Linux distro was configured for all systems, not for your system. This month we show you how to optimize your system for faster startup.

On the DVD: Manjaro KDE Plasma 20.2.1 and Clonezilla Live 2.7.1

#245/April 2021

Choose a Shell

You're never stuck with the same old command shell – unless you want to be. This month we review some of the leading alternatives.

On the DVD: Arch Linux 2021.02.01 and MX Linux mx-19.03

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 her

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 calenda 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.

DrupalCon Europe 2021

Date: October 4-7, 2021

Location: Virtual Event

Website: *https://events.drupal.org/*

DrupalCon Europe 2021 brings together the European Community (and beyond) for 4 days of sessions and workshops. This year, regional camps will be hosted online at the main event. A single DrupalCon ticket gives you access to all events 4-7 October 2021.

All Things Open

Date: October 17-19, 2021

Location: Raleigh, NC and Virtual

Website: *https://www.allthingsopen.org/*

All Things Open is a polyglot technology conference focusing on the tools, processes, and people that make open source possible. This hybrid event will feature three days of in-person and virtual programming.

Events

DeveloperWeek Global: Cloud	September 14-15	Virtual Event	https://www.developerweek.com/global/conference/cloud/
KVM Forum	Septermber 15-16	Virtual Event	https://events.linuxfoundation.org/kvm-forum/
Practical Open Source Information (POSI)	September 16	Virtual Event	https://eventyay.com/e/e7dfbfc4
EuroBSDCon	September 17-19	Virtual Event	https://2021.eurobsdcon.org/
OSDNConf 2021	September 18	Kiev, Ukraine	https://osdn.org.ua/
ApacheCon 2021	September 21-23	Virtual Event	https://apachecon.com/acah2021/
Open Mainframe Summit	September 22-23	Virtual Event	https://events.linuxfoundation.org/
OSPOCon	September 27-29	Seattle, WA and Virtual	https://events.linuxfoundation.org/
Open Source Summit	September 27-30	Seattle, WA and Virtual	https://events.linuxfoundation.org/
Embedded Linux Conference	September 27-30	Seattle, WA and Virtual	https://events.linuxfoundation.org/
Linux Security Summit	September 29-Oct 1	Seattle, WA and Virtual	https://events.linuxfoundation.org/
Open Source Strategy Forum	October 4-5	London, UK and Virtual	https://events.linuxfoundation.org/
Open Source Automation Days	October 4-6	Munich, Germany	https://osad-munich.org/
DrupalCon Europe 2021	October 4-7	Virtual Event	https://events.drupal.org/europe2021
JAX London Hybrid	October 4-7	London, UK and Online	https://jaxlondon.com/
OSPOCon Europe	October 6	London, UK	https://events.linuxfoundation.org/
Cloud Native eBF Day North America	October 11	Los Angeles, CA	https://events.linuxfoundation.org/
O3DECon	October 11-12	Los Angeles, CA	https://events.linuxfoundation.org/
Open Networking & Edge Summit	October 11-12	Los Angeles, CA and Virtual	https://events.linuxfoundation.org/
Kubernetes on EDGE DAY	October 11-12	Los Angeles, CA and Virtual	https://events.linuxfoundation.org/
Cloud Native Security Conference North America	October 12	Los Angeles, CA and Virtual	https://events.linuxfoundation.org/
All Things Open	October 18-19	Raleigh, NC and Virtual	https://www.allthingsopen.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.*

Authors

Erik Bärwaldt	56
Zack Brown	12
Bruce Byfield	6, 22, 46, 60
Joe Casad	3
Mark Crutch	71
Adam Dix	73
Markus Feilner	40
Marco Fioretti	88
Markus Frei	16
Jon "maddog" Hall	72
Frank Hofmann	30
Charly Kühnast	54
Christoph Langner	78
DeeAnn Little	40
Vincent Mealing	71
Leah Metcalfe	68
Pete Metcalfe	68
Graham Morrison	80
Mike Saunders	26
Mike Schilli	36
Michael Spohn	62
Ferdinand Thommes	50
Jack Wallen	8

Contact Info

Editor in Chief
Joe Casad, jcasad@linux-magazine.com

Copy Editors
Amy Pettle, Aubrey Vaughn

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
Dena Friesen

Cover Image
© Olga Yastremska & Laurent Davoust, 123RF.com

Advertising
Brian Osborn, bosborn@linuxnewmedia.com
phone +49 8093 7679420

Marketing Communications
Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
4840 Bob Billings Parkway, Ste 104
Lawrence, KS 66049 USA

Publisher
Brian Osborn

Customer Service / Subscription
For USA and Canada:
Email: cs@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.

Distributed by Seymour Distribution Ltd, United Kingdom

LINUX PRO MAGAZINE (ISSN 1752-9050) is published monthly by Linux New Media USA, LLC, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA. Periodicals Postage paid at Lawrence, KS and additional mailing offices. Ride-Along Enclosed. POSTMASTER: Please send address changes to Linux Pro Magazine, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA.

Published monthly in Europe as Linux Magazine (ISSN 1471-5678) by: Sparkhaus Media GmbH, Bialasstr. 1a, 85625 Glonn, Germany.

Approximate

UK / Europe	Oct 02
USA / Canada	Oct 29
Australia	Nov 29

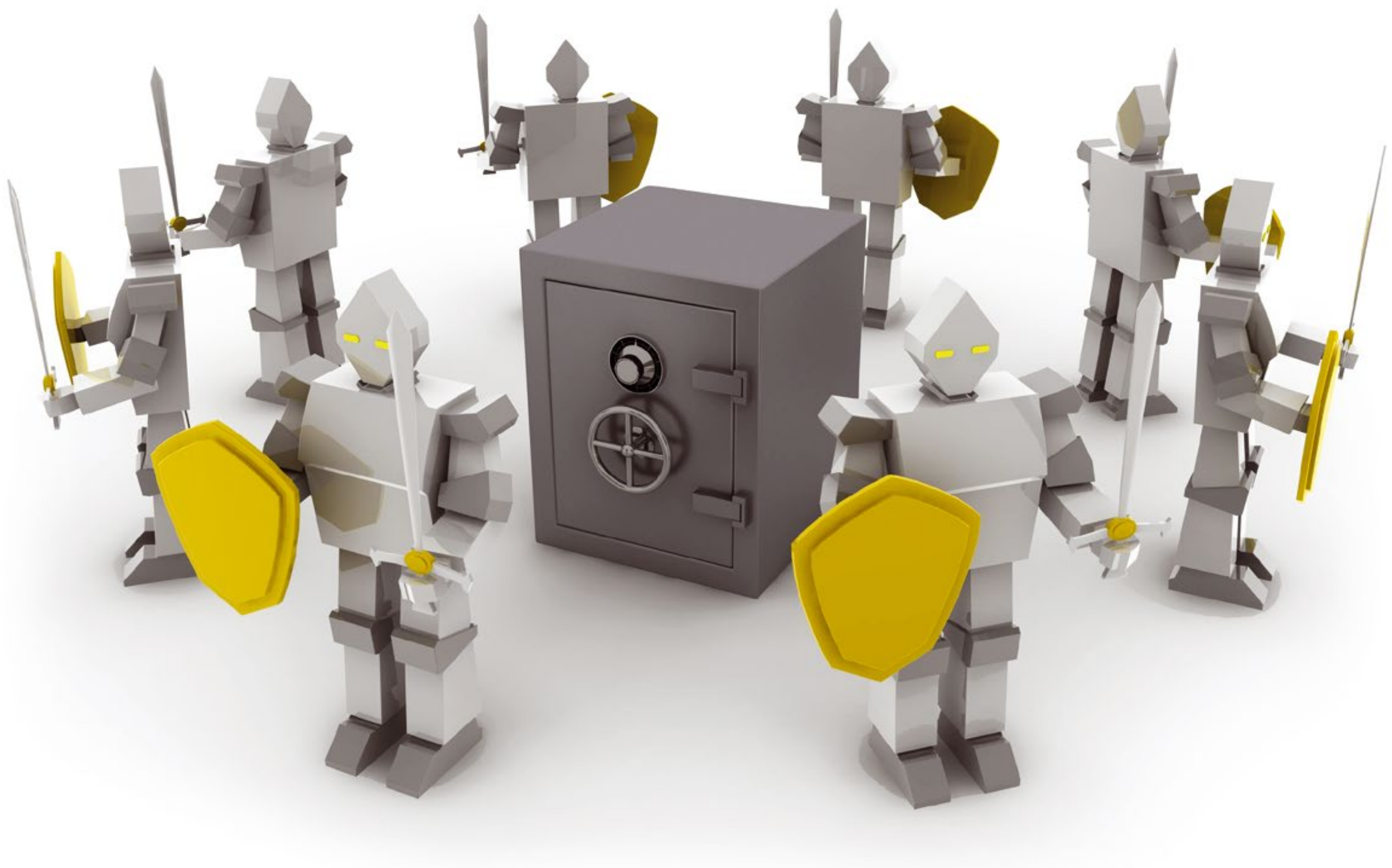
On Sale Date

Please note: On sale dates are approximate and may be delayed because of logistical issues.

Issue 252 / November 2021

The New Security

New technologies have changed the way we protect our personal computers. Next month we study some innovations in PC security, including the Nitrokey smart stick, the OpenPGP smartcard, and Doas, a lighter and simpler version of Sudo.



Preview Newsletter

The Linux Magazine Preview is a monthly email newsletter that gives you a sneak peek at the next issue, including links to articles posted online.

Sign up at: ***https://bit.ly/Linux-Update***

Image © tarokichi, 123RF.com

Sharpen your view!

TUXEDO InfinityBook Pro 14

Intel Core i7-1165G7
Intel Iris Xe Graphics

3K Omnia Display
16:10 | 2880 x 1800 Pixels

Deep Grey
magnesium chassis
1,5 cm thin | 1 kg

Thunderbolt 4
Full featured USB-C 4.0

100%
Linux

5
Year
Warranty

Lifetime
Support

Built in
Germany

German
Privacy

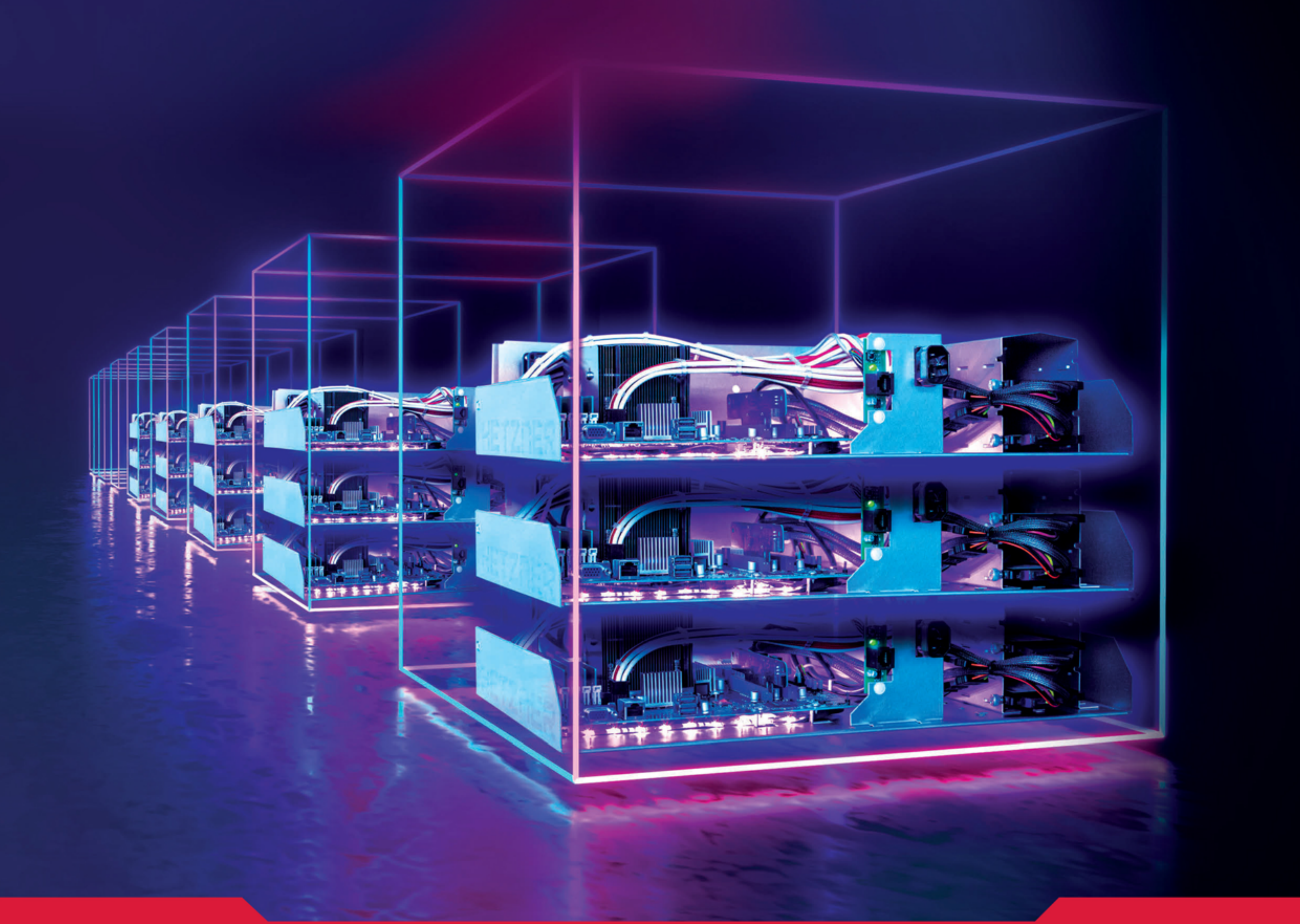
Local
Support

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 $47

AMD RYZEN 5

monthly from **$41**

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 $71

AMD RYZEN 7

monthly from **$65**

All prices exclude VAT and are subject to the terms and conditions of Hetzner Online GmbH. Prices are subject to change. All rights reserved by the respective manufacturers.

www.hetzner.com