Bash Tricks
Roll your own download manager

LINUX PRO MAGAZINE

ISSUE 260 – JULY 2022

Privacy
Keeping your secrets secret

Devuan: Defiant Debian fork with old-school init

Charts and Graphs in LibreOffice

fzf/fzy: Fuzzy search in the shell

DistroTest.net: Test a Linux distro in a browser window

10 TERRIFIC FREE TOOLS!

LINUX NEW MEDIA
The Pulse of Open Source

WWW.LINUXPROMAGAZINE.COM

High-end gaming in a compact form factor

TUXEDO Stellaris 15 Gen4

Intel Core i9-12900H
14 Cores | 20 Threads

3K WQHD IPS Display
2.560 x 1.440 pixels | 240 Hz

NVIDIA GeForce RTX 3080 Ti
TGP up to 175 W

Optomechanical keyboard
RGB | 3 mm key stroke

100% Linux

5 Year Warranty

Lifetime Support

Built in Germany

German Privacy

Local Support

TUXEDO COMPUTERS 18th ANNIVERSARY

tuxedocomputers.com

THE NEW BOSS

Dear Reader,

As this magazine goes to press, Elon Musk, who is attempting to buy Twitter, has announced that, once he ascends to the helm of the micro-blogging colossus, a certain politician who was previously banned from the platform will be let back in. I'll pop in with my usual disclaimer: We talk about tech here, not politics. It doesn't really matter who this politician is or whether you agree with the ban. The question is about whether a sensible process exists for moderating content and whether there will still be one if Musk completes his purchase.

The news of this impending change is causing a stir throughout the Internet. Musk has declared himself a "free speech absolutist" [1] and has said that he considers Twitter to be a public square, where people should be able to say whatever they want. Many are worried about what form this free speech absolutism will take. Twitter critics welcomed the arrival of the platform's moderation policies as a way of restraining the hate, bullying, misogyny, and disinformation (otherwise called *lies*) that have become associated with the platform, and they worry about what less restrictive moderation could mean for the world.

As is often the case, though, the fine print tells a story that is missing from the headlines. Like most enigmatic billionaires, Mr. Musk has a way of carefully choosing his words and saying several things at the same time. He is clearly courting the users who departed from Twitter when the banned politician started a rival platform, stating that the ban was "morally wrong and flat out stupid" [2]. But if you look to what he really has in mind for policy, the picture gets a bit cloudier. Musk said he does not believe in *permanent* bans, but he left the door open for temporary suspensions – and the removal of tweets – for speech that is "illegal or otherwise just destructive to the world." He goes on to add, "I think if there are tweets that are wrong and bad, those should be either deleted or made invisible, and a suspension, a temporary suspension is appropriate but not a permanent ban."

So what did Elon just say exactly? If you're not going to allow tweets that are "destructive to the world," you need some form of agent or entity that will determine whether a tweet is destructive or not. And once you work through the details of how to do that, it actually sounds quite a bit like moderation.

It is also interesting that Musk recently met with EU commissioner Thierry Breton and announced that he is "on the same page" with the EU on the proposed Digital Services Act [3], an expansive new law that would require social media companies to police their content more aggressively to remove hate speech and disinformation. He tucked his absolutism away for that meeting. These positions might seem incompatible, however, Musk has stated that he believes moderation should match the laws of the country for which it is intended. (To be fair, the Digital Services Act also has other features that Musk has said he agrees with, such as greater transparency and open algorithms.)

This notion that the platform should match the laws of the country seems in stark contrast to the concept of the free speech absolutism he is touting over here in the USA, and it appears to imply that he would happily comply with the rules imposed by countries such as Russia and China to censor the speech of dissidents. In other words, he won't let his personal values get in the way of business.

The need to do business could indeed temper his aspirations for a free-for-all public square. Musk, always the Twitter enthusiast, was once sued for using his Twitter freedom to call a cave diver who rescued the boys soccer team from the cave a "pedo guy" [4]. He later won the case, with his lawyers making the argument that just because you call someone a "pedo guy" doesn't mean you are calling them a pedophile – that's just the kind of stuff you say about people on Twitter when they don't like your ideas.

If he nurtures an environment where that kind of trash talk flourishes, he might find that a lot of users will just leave. The high-tech community has this way of viewing itself in epic, historical terms as the inevitable vanguard for a new civilization, but the fact is, the only thing that makes Twitter important is that everyone thinks it is important. If news sources quit quoting it, if people quit trusting it, if the market it serves now as a near monopoly breaks into smaller markets aimed at more targeted audiences, Twitter loses – and we know Elon Musk doesn't like to lose.

So maybe the new Twitter won't be so different after all? Or to quote The Who: "Meet the new boss / Same as the old boss."

Joe Casad,
Editor in Chief

Info

[1] Milmo, Dan. "How 'Free Speech Absolutist' Elon Musk Would Transform Twitter," *Guardian*, Apr. 14, 2022, *https://www.theguardian.com/technology/2022/apr/14/how-free-speech-absolutist-elon-musk-would-transform-twitter*

[2] "Elon Musk Says He Would Reverse Twitter's Trump Ban," CNN, May 10, 2022, *https://www.cnn.com/2022/05/10/tech/elon-musk-twitter-trump-ban/index.html*

[3] Elon Musk Says He's 'Very Much on the Same Page' as the EU on Social Media Laws," by Adi Robertson, The Verge, May 10, 2022, *https://www.theverge.com/2022/5/10/23065270/elon-musk-twitter-digital-services-act-dsa-european-union-eu-speech-laws-thierry-breton*

[4] Groom, Nichola and Rachel Parsons. "Tesla Boss Elon Musk Wins Defamation Trial," Reuters, Dec. 6, 2019, *https://www.reuters.com/article/us-musk-lawsuit/tesla-boss-elon-musk-wins-defamation-trial-over-his-pedo-guy-tweet-idUSKBN1YA13U*

LINUX MAGAZINE
JULY 2022

ON THE COVER

26 Devuan
When Debian opted for systemd init, a spirited group of rebels stood their ground.

32 Shell Download Manager
Manage your downloads with this handy homegrown tool.

46 DistroTest.net
Test drive a Linux distro without installing.

72 fzf/fzy
Fuzzy search lets you find the answers without getting hung up on spelling.

90 LibreOffice Charts
Light up your office memos with informative charts and graphs.

NEWS

08 News
- Danielle Foré Has an Update for elementary OS 7
- Linux New Media Launches Open Source JobHub
- Ubuntu Cinnamon 22.04 Now Available
- Pop!_OS 22.04 Has Officially Been Released
- Star Labs Unveils a New Small Format Linux PC
- MX Linux v21.1 "Wildflower" Now Available

12 Kernel News
- Taking a Header
- The Next Spectre Vulnerability

COVER STORIES

16 Portmaster
Security and anonymization play an increasingly important role on the Internet due to the endless appetite of Internet companies for personal data. Portmaster and the Safing Privacy Network will help you protect your privacy – even if you're not a security expert.

20 Anonymity on the Tor Network
The Tor Project supports a formidable collection of tools for protecting your privacy on the Internet. We'll give you some background on Tor and help you get started with the Tor Browser.

REVIEW

26 Distro Walk – Devuan
Devuan, with its promise of Init Freedom, provides users an alternative to systemd as an init process.

IN-DEPTH

28 AlekSIS
In many classroom settings, paper grade lists and the good old class register are still the rule. The AlekSIS project offers a 21st century solution for small schools that want to go digital.

30 Securing Thunderbolt
The Thunderbolt interface supports extremely fast data transfer rates, but be careful about what you plug into your port, because Thunderbolt devices access memory directly.

32 Shell Download Manager
A few lines of shell code and the Gawk scripting language make downloading files off the web a breeze.

38 High Availability vs. Backup
Some users trust their data to powerful file servers that advertise enterprise data protection, but your network-attached storage system might not be as safe as you think it is.

43 Command Line – kitty
Kitty, a terminal emulator by the creator of Calibre, promises customization and graphical acceleration at the command line.

46 DistroTest.net
With so many Linux distros to choose from, you can spend a lot of time downloading and installing before you find the right one. DistroTest.net lets you test a variety of Linux distributions from the browser without installing.

48 BlueSpice 4.1
Keep your team collaborating with BlueSpice, a wiki tool focused on professional customers.

Privacy

If you are really serious about privacy, you'll need to lean on more than your browser's no tracking button. Those who need anonymity the most depend on the Tor network – a global project offering safe surfing even in surveillance states. We also look at Portmaster, an application firewall with some useful privacy features.

IN-DEPTH

52 DIY Read-It-Later Tool
Instead of relying on a third-party read-it-later service, you can use this DIY tool to save articles from the Internet.

56 Programming Snapshot – A Go Wordle Cracker
Mike Schilli builds a command-line tool to boost his Wordle streak.

MakerSpace

60 Digital Spirit Level
The small MPU6050 sensor contains a gyroscope and an accelerometer, which means that you can build a digital spirit level with it.

64 New Tech Retrofit
Arduino was the solution for an electronic display at a local science center that was showing its age.

LINUXVOICE

69 Welcome
This month in Linux Voice.

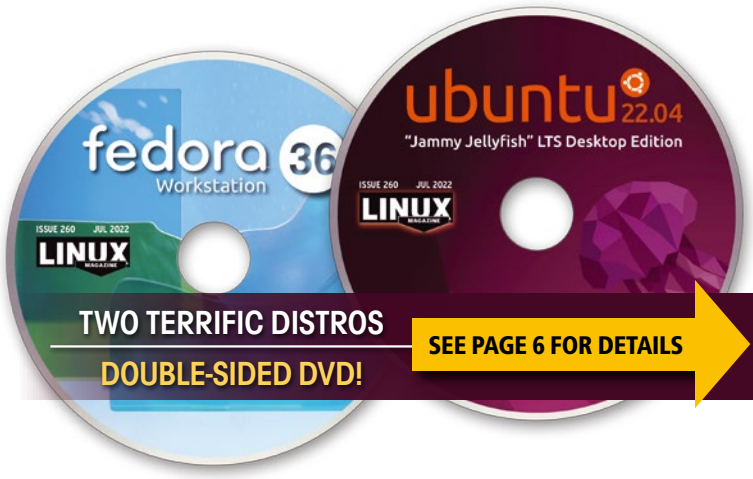
71 Doghouse – Local Internet Support
Tools such as community servers and local web traffic caches can help improve community Internet service.

72 fzf/fzy
Fuzzy finders retrieve useful results from data streams even if there are no exact matches.

78 Dim Media Manager
Dim, a relatively new open source media manager, looks to implement the appearance and feature set of the commercial Plex media center.

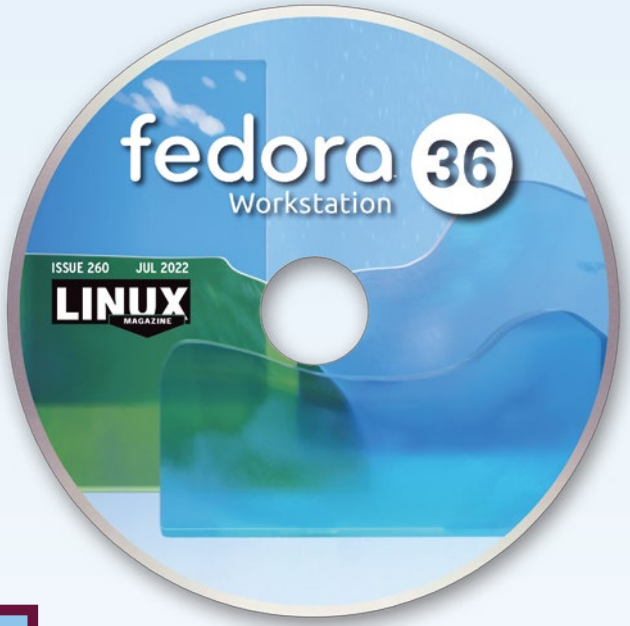
84 FOSSPicks
This month Graham looks at TerraForge3D, Navidrome, qddcswitch, Difftastic, fheroes2, and more!

90 Tutorial – LibreOffice Charts
Everybody needs charts sooner or later, and LibreOffice Calc is the easiest way to create them with free and open source software.



Ubuntu 22.04 and Fedora 36

Two Terrific Distros on a Double-Sided DVD!



Ubuntu 22.04
64-bit

Codenamed Jammy Jellyfish, Ubuntu 22.04 is the latest long-term support (LTS) release, receiving security and maintenance updates until April 2027. In the usual tradition of LTS releases, Jammy Jellyfish is full of new features and modifications.

Many of the modifications are cosmetic, such as an option for a shorter dock, a change in icon sizes in various dialogs, and more animations for those who choose them. Some seem arbitrary, such as placing icons in the desktop's lower right, while others are more versatile, such as the addition of hot corners for single-click actions or the option for a minimal installation, which is a security feature long overdue.

Unusual for an LTS release, Jammy Jellyfish is shipping with some important limitations:

- The new screenshot tool performs poorly under Wayland.
- Wayland is used by default, but not for NVIDIA cards.
- The GRUB bootloader does not automatically detect other operating systems or distributions. To correct this problem, download `os-prober` if necessary, and then add the line `GRUB_DISABLE_OS_PROBER=false` to the `/etc/default/grub` file and update GRUB.

Given the five years of support, these problems should be fixed shortly. Meanwhile, if these problems do not affect you, then this latest Ubuntu release should provide the distribution's user-friendliness for all levels of users.

Fedora 36
64-bit

Fedora 36 has been delayed several times, but it should be released by the time you read this. It will be supported until May 17, 2023.

The Fedora distribution has a long-established reputation for innovation, but Fedora 36 is a relatively modest release. As usual, the new release includes new wallpapers and themes, many of them reflecting the recent fad for dark modes. However, many of the new features are due to the Linux 5.17 kernel and the very latest Gnome 42 release, and they are mostly under the hood. Fedora's own contributions to the release are similarly largely concealed, such as enhanced support for NVIDIA drivers and Wayland. Fedora 36 can be considered a maintenance release that catches up with upstream projects and tweaks the distribution's inner workings, continuing and solidifying the modifications that have been eased into the distro over the past few years.

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

NEWS

Updates on technologies, trends, and tools

THIS MONTH'S NEWS

08
- Danielle Foré Has an Update for elementary OS 7
- Linux New Media Launches Open Source JobHub

09
- Ubuntu Cinnamon 22.04 Now Available
- Pop!_OS 22.04 Has Officially Been Released
- More Online

10
- Star Labs Unveils a New Small Format Linux PC
- MX Linux Verison 21.1 "Wildflower" Now Available

Danielle Foré Has an Update for elementary OS 7

Elementary OS founder and CEO Danielle Foré has announced updates for version 6.1 and adds that the elementary OS team is now preparing for the upcoming 7.0 release. Although there's no hard release date for 7.0, they've announced a codename (Horus) and that the team is focusing on a swift release over adding a host of new features for the initial release. That means the evolution from 6.1 to 7.0 will be subtle.

elementary

Some of the features they are planning for 7.0 include automatic app updates and power profiles. Additionally, 7.0 will debut the minimalist Music app and the migration of some components migrating to Gtk4 (which should bring much-improved performance and smoother animations).

One thing 7.0 won't do is make the switch to Wayland. Although they are preparing for the shift, elementary OS isn't ready for the jump.

Find out more about what's in store on the official elementary OS blog (*https://blog.elementary.io/*).

Linux New Media Launches Open Source JobHub

Linux New Media, along with FOSSlife and *Linux Magazine*, is pleased to announce the launch of Open Source JobHub (*https://opensourcejobhub.com/*), a job board to help people find their place in the open source ecosystem.

Aimed at developers, engineers, managers, marketers, and more, Open Source JobHub can help you find the perfect job fit. Now, more than ever, the open source tech industry is exploding, and Open Source JobHub will help you navigate the growing number of opportunities.

"Open Source JobHub not only covers jobs using open source technologies, but also other roles such as sales, marketing, and management at companies dedicated to open source. Our goal is to give the global open source community a specific platform through which to make career connections," said Brian Osborn, CEO and publisher, Linux New Media.

OpenSource | JOB HUB

Open Source JobHub now features job listings from launch partners CloudLinux, Collabora, SUSE, and TUXEDO Computers. The site will also provide resources to help job seekers build their careers with open source, and employers can quickly upload listings and reach qualified candidates.

"At Collabora, our focus has always been to promote open source technologies, whether that be our own Engineers, who are some of the most motivated and active contributors to open source, or our non-technical staff covering multiple departments. Open Source JobHub will allow the further promotion of open source by increasing awareness within the industry," said Ben Toynton, Technical Recruiter at Collabora.

Employers can post five free job listings during the launch phase. Follow Open Source JobHub on Twitter @OSJobHub (*https://twitter.com/OSJobHub*).

Ubuntu Cinnamon 22.04 Now Available

For those who prefer the Cinnamon desktop but would rather have all the fancy Ubuntu underpinnings (instead of going with Linux Mint), developer Joshua Peisach has announced the release of his Ubuntu remix, Ubuntu Cinnamon. This new version is based on Ubuntu 22.04 LTS and includes the Cinnamon 5.2 desktop. Cinnamon 5.2 adds improvements to the Menu applet (such as better keyboard navigation when using left to right languages). The Calendar applet now supports GNOME's Evolution Data Server for contacts, tasks, and calendar information.

Beyond what Cinnamon 5.2 brings to the table, Ubuntu Cinnamon 22.04 added fixes for background and screen tearing issues, Wayland set as the default when using non-NVIDIA graphics cards, updates for Active Directory (including full Group Policy support, privilege escalation, and remote script execution), and plenty of performance and reliability improvements.

Of this release, Peisach said, "This release has been two years in the making, and I am so grateful for everyone who helped – through quality assurance/testing, bug squashing, general development, and those across the different Ubuntu teams."

For more information about the release, check out Peisach's blog post (*https://ubuntucinnamon.org/ubuntu-cinnamon-remix-22-04-lts-jammy-jellyfish-released/*) and download a copy of the ISO to install on either bare metal or as a virtual image (*https://ubuntucinnamon.org/download/*).

Pop!_OS 22.04 Has Officially Been Released

System76 has officially released the latest iteration of Pop!_OS. This time around, the operating system is based on Ubuntu 22.04 and includes plenty of improvements. Although the changes found in 22.04 aren't nearly as dramatic as those in past releases, this new version still offers plenty to get excited about.

The Pop!_OS COSMIC desktop is based on GNOME 42 but has been stripped down to align with the vision System76 has with its desktop. This means the look and feel of the desktop will remain fairly consistent with what you have experienced since COSMIC was first introduced. In fact, although GNOME 42 migrated away from Gedit and GNOME Terminal, Pop!_OS is sticking with those two apps for the time being.

Two of the biggest additions to Pop!_OS 22.04 come in the form of automatic updates (which are configured in Settings > OS Recovery & Update) and Pipewire now shipping as the default audio server.

MORE ONLINE

Linux Magazine
www.linux-magazine.com

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

Extended File Attributes
• Jeff Layton
One way to store metadata is with the originating file in extended file attributes.

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

Incident Analysis with The Hive and Cortex
• Matthias Wübbeling
Deployed together, The Hive platform and Cortex automation tool optimize the workflow for your incident response team.

High-Performance Backup Strategies
• Jan Kappen
A sound backup strategy with appropriate hardware and software ensures you can back up and restore your data rapidly and reliably.

Harden Services with systemd
• Jens-Christoph Brendel
Systemd comes with a metric for determining the security of your system, letting you track how any service can be secured step-by-step in a sandbox.

Other highlights for the new version include a new screenshot tool, a Support option in the Settings app, kernel 5.16, GNOME Settings 41.4, Firefox 98, GNOME Terminal 3.43.9, and Gedit 41.4.

You can either download a copy of Pop!_OS from the official site (*https://pop.system76.com/*), or you can upgrade your current release using the `sudo pop-upgrade release upgrade -f` command. Just remember, if you run an upgrade, back up all of your important data before doing so.

Star Labs Unveils a New Small Format Linux PC

Star Labs has been known as a manufacturer of Linux laptops for some time. Recently, however, the company has dipped its toes into the mini PC market and its first release is impressive on paper.

The Byte MK I is a first-gen mini PC equipped with an AMD Ryzen 7 5800 Octo-Core CPU (up to a 4.4GHz clock speed), AMD Radeon graphics, up to 64GB of 3200MHz RAM, and up to 6TB SSD storage.

Besides the power the Byte MK I offers, one of the most impressive features is the number of ports you'll find, which includes two full-sized USB 2.0, two full-sized USB 3.0, two full-sized HDMI, gigabit ethernet RJ45, one USB-C (for power and expansion), a combo audio/mic jack, and a Micro SD slot. Finally, the Byte MK I ships with the Coreboot firmware and the Star Labs Coreboot Configurator utility.

The base price for the Byte MK I is $793.00 and includes 8GB of RAM and 240 GB of SSD storage. The ship date for orders is late June 2022 and you can get a five percent discount for pre-orders (while the device is still in production).

Order your Star Labs Byte MK I now (*https://us.starlabs.systems/pages/byte*).

MX Linux Verison 21.1 "Wildflower" Now Available

MX Linux is a desktop distribution that shuns systemd and opts to go with the older SysVInit initialization system. As it has in the past, MX Linux offers versions with Xfce, KDE Plasma, and MX-fluxbox. This distribution eschews flash for simplicity and offers 32-bit versions for the Xfce and Fluxbox releases and 64-bit versions for all three desktops.

This new release is based on Debian 11.3, ships with the 5.16 Advanced Hardware Support kernel and has plenty of updated applications, the usual bug fixes, and an improved installer. One pleasant surprise to be found in MX Linux 21.1 is the `mx-samba-config` tool for the simplified setup of Samba/CIFS shares.

Another interesting addition is that the developers have brought back the disk-manager utility, which automatically detects new partitions at startup, fully manages the configuration of file systems, and allows you to enable/disable write support for NTFS.

If you're already running the 21.0 version of MX Linux, you can upgrade to the .1 version through the regular update channels.

You can download MX Linux with the Xfce Desktop (*https://sourceforge.net/projects/mx-linux/files/Final/Xfce/MX-21.1_x64.iso/download*), KDE Plasma (*https://sourceforge.net/projects/mx-linux/files/Final/KDE/MX-21_KDE_x64.iso/download*), or MX-Fluxbox (*https://sourceforge.net/projects/mx-linux/files/Final/Fluxbox/MX-21.1_fluxbox_x64.iso/download*).

Read the official release notes from the MX Linux blog (*https://mxlinux.org/blog/mx-21-1-wildflower-released/*).

DrupalCon
PRAGUE2022
20-23 SEPTEMBER

DrupalCon Prague
let's meet in person again!

DRUPALCON EUROPE WILL BE HAPPENING IN THE CAPITAL CITY OF CZECH REPUBLIC, PRAGUE, BETWEEN 20 - 23 SEPTEMBER, 2022.

Visit our website https://events.drupal.org/prague2022 to get all important information.

Registration and sponsorship sales are opened!

Zack's Kernel News

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

By Zack Brown

Author

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

Taking a Header

In the Winter of 2020, Ingo Molnar decided that something simply had to be done to make everyone's life better. He reached into his ultimate sack of horrible things and pulled out the Linux kernel header hierarchy. This oozing nightmare consisted of all the header files in the kernel source tree, one depending upon the other in an endless glutinous web that could be neither untangled nor untied and that all kernel sub-projects simply glom onto, forming endless sticky layers upon which the fate of humanity truly does depend.

So Ingo untangled and untied it using determination and strange gifts. Then recently he submitted a patch, consisting of over 25 sub-trees, with over 2,200 individual commits, changing more than half of all source files in the entire kernel tree. He said, "As most kernel developers know, there's around ~10,000 main .h headers in the Linux kernel, in the include/ and arch/*/include/ hierarchies. Over the last 30+ years they have grown into a complicated & painful set of cross-dependencies we are affectionately calling 'Dependency Hell'."

He offered his patch to the world, calling it the Fast Kernel Headers project. According to his tests, it would cut kernel compile times down to as much as one fifth of what they had been. Incremental compile times – where files compiled earlier don't need to be recompiled – were even more drastically improved. The oozing web had become a delicate lace – or at least less hellish.

Ingo explained:

"When I started this project, late 2020, I expected there to be maybe 50-100 patches. I did a few crude measurements that suggested that about 20% build speed improvement could be gained by reducing header dependencies, without having a substantial runtime effect on the kernel. Seemed substantial enough to justify 50-100 commits.

"But as the number of patches increased, I saw only limited performance increases. By mid-2021 I got to over 500 commits in this tree and had to throw away my second attempt (!); the first two approaches simply didn't scale, weren't maintainable and barely offered a 4% build speedup, not worth the churn of 500 patches and not worth even announcing.

"With the third attempt I introduced the per_task() machinery which brought the necessary flexibility to reduce dependencies drastically, and it was a type-clean approach that improved maintainability. But even at 1,000 commits I barely got to a 10% build speed improvement. Again this was not something I felt comfortable pushing upstream, or even announcing. :-/

"But the numbers were pretty clear: 20% performance gains were very much possible. So I kept developing this tree, and most of the speedups started arriving after over 1,500 commits, in the fall of 2021. I was very surprised when it went beyond 20% speedup and more, then arrived at the current 78% with my reference config. There's a clear super-linear improvement property of kernel build overhead, once the number of dependencies is reduced to the bare minimum."

He went on, "the size of the 'default' headers (which with the fast-headers tree will mostly include type definitions), has been reduced by 1-2 orders of magnitude. Much of the build speed improvement is due to these reductions."

And finally, Ingo said, "so this is probably the largest single feature announcement in LKML's history. Not by choice! :-/ For this reason this tree is an RFC announcement, and I'd like to gather feedback from fellow maintainers about the structure of tree(s) before pushing for an upstream merge."

Greg Kroah-Hartman was highly impressed and offered a few technical suggestions. But he said, "I took a glance at the tree, and overall it looks like a lot of nice cleanups. Most of these can probably go through the various subsystem trees, after you split them out, for the 'major' .h cleanups."

Ingo replied:

"I absolutely plan on doing that too:

- About ~70% of the commits can be split up & parallelized through maintainer trees.

- With the exception of the untangling of sched.h, per_task and the "Optimize Headers" series, where a lot of patches are dependent on each other. These are actually needed to get any measurable benefits from this tree (!). We can do these through the scheduler tree, or through the dedicated headers tree I posted.

"The latter monolithic series is pretty much unavoidable; it's the result of 30 years of coupling a lot of kernel subsystems to task_struct via embedded structs & other complex types, that needed quite a bit of effort to untangle, and that untangling needed to happen in-order."

And Greg affirmed, "Yes, taking the majority through the maintainer trees and then doing the remaining bits in a single tree seems sane; that one tree will be easier to review as well."

Nathan Chancellor was also gobsmacked by Ingo's work. He ran some tests and saw an 18-35 percent speed improvement on his 80-core ARM64 server. Ingo replied, "Note that on ARM64 the elapsed time improvement is 'only' 18-35%, because the triple-linking of vmlinux serializes much of the of a build & ARM64 doesn't have the kallsyms-objtool feature yet." But he felt there was a lot of room for improvement on ARM architectures. Ingo added, "In the end I think the improvement could probably [be] moved into the broad 60-70% range that I see on x86."

Nathan offered a bunch of code suggestions and patches, which Ingo accepted gratefully, and the two of them had a technical discussion about remaining issues. Ingo remarked, "Your testing & patch sending efforts are much appreciated!! You'd help me most by continuing on the same path with new fast-headers releases as well, whenever you find the time. :-)"

Willy Tarreau also replied to Ingo's initial announcement, saying, "great work! I'm particularly interested in this work because I went through a similar process about 6 months ago in haproxy and saved 40-45% build time, and thought how well the same principles could apply to the kernel if anyone had felt brave enough to engage into that. I do appreciate how tedious a work it can be and do really sympathize with you on this!"

Nick Desaulniers also had some technical comments on Ingo's code, adding:

"This is a really cool series Ingo. I'm sure Arnd has seen it by now, but Arnd has been thinking about this area a lot, too. I haven't but I have played with running 'include what you use' on the kernel sources; Kconfig being the biggest impediment to that approach.

"To me, I'm most nervous about 'backsliding;' let's say this work lands, at some point probably years in the future, I assume without any form of automation that we might find ourselves at a similar point of header dependencies getting all tangled again.

"What are your thoughts on where/how/what we could automate to try to help developers in the future keep their header dependencies simpler? (Sorry if this was already answered in the cover letter.)

"It would be really useful if you were planning a talk at something like plumbers [Linux Plumbers Conference] how you go about making these changes. I really hope once others understand your workflow that we might help with some form of automation. Nice work!"

And Arnd Bergmann, nearby, also said to Ingo, "I've done some work in this area in the past, didn't quite take it enough of the way to get this far. The best I saw was 30% improvement with clang, which tends to be more sensitive than gcc towards header file bloat, as it does more detailed syntax checking before eliminating dead code."

The whole conversation grew into a large implementation discussion, with everyone chiming in. A few days later, Ingo announced version two of the patch, which had grown from "over 2,200 commits" to "over 2,300 commits" since version 1 and offered an even more impressive speed improvement over the official kernel. The implementation discussion continued, mostly between Ingo and Arnd.

This kind of lunacy happens from time to time – someone decides to tackle one of the ancient kernel nightmares, like the big kernel lock, or fixing all build-time warnings, or cleaning up the header hierarchy, and suddenly, in this sphere at least, the world is a brighter, happier place.

The Next Spectre Vulnerability

Recently a new Spectre-like security vulnerability was uncovered in a variety of CPU architectures. It's a double annoyance because, firstly, vulnerabilities must be patched, and secondly, the workaround likely involves some amount of runtime performance hit.

This time around, all of a sudden there was a gigantic blizzard of Spectre patches coming into the kernel. Among them, dear to my heart, were a bunch of documentation patches. One of these, from Tim Chen and Andi Kleen, explained the nature and risks associated with Spectre, the ways to mitigate the security problems, and how to use the sysfs files relevant to dealing with Spectre.

In the document, Tim and Andi explain, "Spectre is a class of side channel attacks that exploit branch prediction and speculative execution on modern CPUs to read memory, possibly bypassing access controls. Speculative execution side channel exploits do not modify memory but attempt to infer privileged data in the memory."

They went on to say:

"Speculative execution side channel methods affect a wide range of modern high performance processors, since most modern high speed processors use branch prediction and speculative execution.

"The following CPUs are vulnerable:

- Intel Core, Atom, Pentium, and Xeon processors
- AMD Phenom, EPYC, and Zen processors
- IBM POWER and zSeries processors
- Higher end ARM processors
- Apple CPUs
- Higher end MIPS CPUs
- Likely most other high performance CPUs. Contact your CPU vendor for details.

"Whether a processor is affected or not can be read out from the Spectre vulnerability files in sysfs."

The document went on to say:

"CPUs use speculative operations to improve performance. That may leave traces of memory accesses or computations in the processor's caches, buffers, and

branch predictors. Malicious software may be able to influence the speculative execution paths, and then use the side effects of the speculative execution in the CPUs' caches and buffers to infer privileged data touched during the speculative execution.

"Spectre variant 1 attacks take advantage of speculative execution of conditional branches, while Spectre variant 2 attacks use speculative execution of indirect branches to leak privileged memory."

They gave an example of a hostile user process attacking the kernel itself:

"The attacker passes a parameter to the kernel via a register or via a known address in memory during a syscall. Such parameter may be used later by the kernel as an index to an array or to derive a pointer for a Spectre variant 1 attack. The index or pointer is invalid, but bound checks are bypassed in the code branch taken for speculative execution. This could cause privileged memory to be accessed and leaked.

"For kernel code that has been identified where data pointers could potentially be influenced for Spectre attacks, new 'nospec' accessor macros are used to prevent speculative loading of data.

"Spectre variant 2 attacker can poison the branch target buffer (BTB) before issuing syscall to launch an attack. After entering the kernel, the kernel could use the poisoned branch target buffer on indirect jump and jump to gadget code in speculative execution.

"If an attacker tries to control the memory addresses leaked during speculative execution, he would also need to pass a parameter to the gadget, either through a register or a known address in memory. After the gadget has executed, he can measure the side effect.

"The kernel can protect itself against consuming poisoned branch target buffer entries by using return trampolines (also known as 'retpoline') for all indirect branches. Return trampolines trap speculative execution paths to prevent jumping to gadget code during speculative execution. x86 CPUs with Enhanced Indirect Branch Restricted Speculation (Enhanced IBRS) available in hardware should use the feature to mitigate Spectre variant 2 instead of retpoline. Enhanced IBRS is more efficient than retpoline."

The document also included detailed examples of a hostile user process attacking another user process, a virtualized system attacking the underlying kernel, and a virtualized system attacking another virtualized system.

There were not only documentation patches – most were actual code changes that fixed things. A few days after the patch storm began, Linus Torvalds announced Linux 5.17-rc8, saying:

"So last weekend, I thought I'd be releasing the final 5.17 today.

"That was then, this is now. Last week was somewhat messy, mostly because of embargoed patches we had pending with another variation of spectre attacks. And while the patches were mostly fine, we had the usual 'because it was hidden, all our normal testing automation didn't see it either'.

"And once the automation sees things, it tests all the insane combinations that people don't tend to actually use or test in any normal case, and so there was a (small) flurry of fixes for the fixes.

"None of this was really surprising, but I naïvely thought I'd be able to do the final release this weekend anyway.

"And honestly, I considered it. I don't think we really have any pending issues that would hold up a release, but on the other hand we also really don't have any reason _not_ to give it another week with all the proper automated testing. So that's what I'm doing, and as a result we have an -rc8 release today instead of doing a final 5.17.

"There's a number of non-spectre things in here too, of course. Among other things, people finally chased down a couple of mislaid patches that had been on the regression list, so hopefully we have those all nailed down now too.

"And obviously there's all the usual random fixes in here too. But because of the spectre thing, about half of the -rc8 patch is architecture updates.

"That said, it's still a fairly _small_ half of the patch. It was not one of the 'big disaster' hw speculation things; it was mostly extending existing mitigations and reporting.

"Anyway, let's not keep the testing _ just_ to automation – the more the merrier, and real-life loads are always more interesting than what the automation farms do. So please do give this last rc a quick try."

Several people replied about the non-Spectre patches, but there was no Spectre discussion in that particular thread.

FOSSLIFE

Open for All

News • Careers • Life in Tech
Skills • Resources

FOSSlife.org

Manage Internet uploads with Portmaster

Full Control

Security and anonymization play an increasingly important role on the Internet due to the endless appetite of Internet companies for personal data. Portmaster and the Safing Privacy Network will help you protect your privacy – even if you're not a security expert.

By Erik Bärwaldt

Intensified data grabbing is making life difficult for users on the Internet. It's not just the usual suspects like Google or Facebook who are collecting user data. Even conventional software packages have increasingly started phoning home and sending "telemetry data" to their vendors or third parties.

Users typically don't notice this data transfer and cannot track what data is being sent to whom. To stop this bad habit, a startup by the name of Safing, which has already twice received funding from the Austrian innovation incubator Netidee, has developed an application firewall called Portmaster that lets everyday users track and control the flow of data to hidden recipients [1].

Idea

Portmaster combines several privacy-related services in a single package. Included within the Portmaster application is a firewall, a system of filter lists to identify trackers and other undesirable sites, a secure DNS service, and an optional privacy service (similar to the TOR network) called the Safing Privacy Network (SPN).

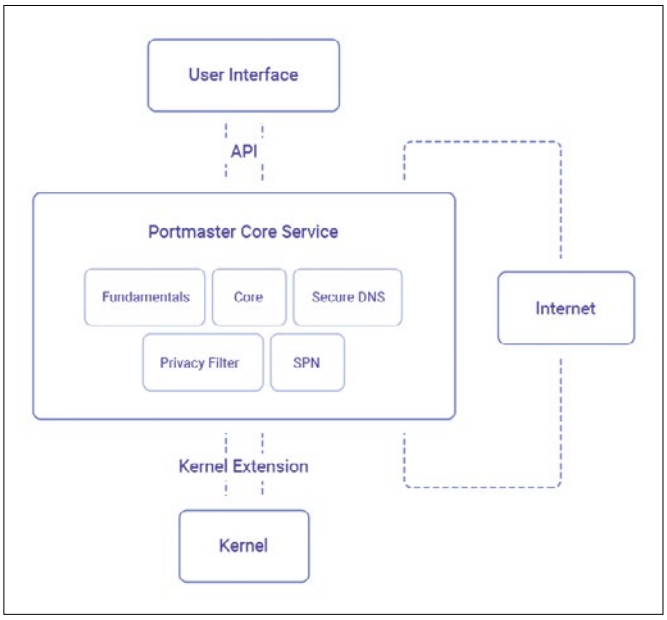


Figure 1: The Portmaster Core Service resides between the kernel and the user interface (from the Portmaster website [1]).

Perhaps the most interesting part of Portmaster is the way the developers have encapsulated all that functionality into a single user interface that you don't have to be an expert to understand and manage. The intuitive Portmaster user interface makes it easy to monitor and block network connections, set filters to automatically block trackers and adware, and configure different filter settings for different applications. Portmaster is free software hosted on GitHub [2] and provided under the GNU Affero General Public License (AGPL 3.0).

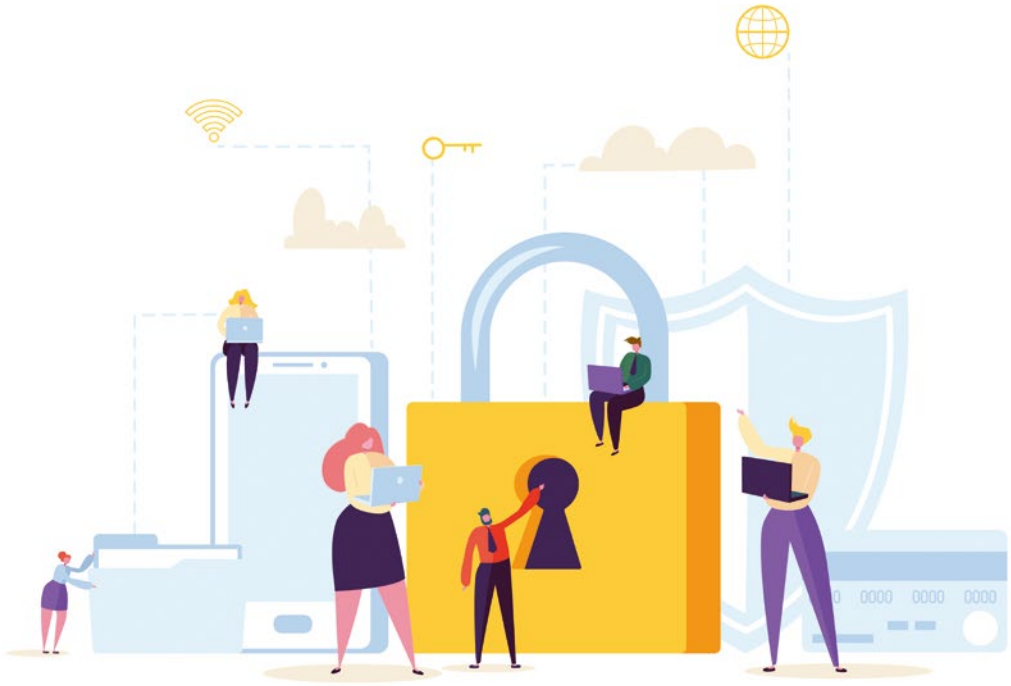
How It Works

Under the hood, what is known as a Portmaster Core Service that sits between the kernel and the user interface on one side and the kernel and the Internet on the other (Figure 1). This core service consists of several components, the most important of which are the SPN, the privacy filters, and the Secure DNS service.

The Secure DNS service uses the DNS-over-TLS (DoT) protocol, which sends DNS queries over an encrypted TLS connection. This encrypted connection stops unauthorized third parties from viewing the DNS queries. The privacy filters, which act much like a firewall, also use filter lists. The system references the filter lists to block undesirable connections.

The manufacturer is continuously developing the filter lists – lists of sites associated with malware, tracking, phishing, or other nefarious activities. The lists are maintained on a separate GitHub page (Figure 2). You can also add your own entries defining sites you wish to filter.

The SPN is an ambitious project that is still in its early stages of development. The company's long term plan appears to be to continue to give Portmaster away for free, but to sell access to SPN, which the company says will eventually obfuscate IP addresses [3] and prevent third parties from viewing data. SPN routes data packets through multiple servers on the Internet in an approach that is similar to



the TOR service. (See the article on the TOR network elsewhere in this issue.) SPN is currently in what the company describes as the alpha stage. According to the Safing website, "Treat the SPN as a VPN in your threat model for now. Please be aware that there are not enough users and servers during the alpha phase in order to protect you from VPN traffic analysis" [4]. But even if you don't decide to experiment with SPN, the intuitive user interface and background services of Portmaster are worthy of some attention.

Installation

Portmaster is available in binary package form for most popular Linux distros. A compatibility list available in the documentation shows which kernel versions and desktop environments Portmaster supports.

Most recent Linux kernels are fully compatible with Portmaster, except for version 5.6, which has a problem accessing the Netfilter queue. The widely used KDE Plasma, Gnome, Xfce, and Cinnamon desktop environments all work with Portmaster, although Budgie appears to have a problem with displaying the Portmaster icon in the taskbar.

The project's website offers installation instructions for many popular Linux distros, including information on the dependencies you need to resolve in order to achieve a speedy installation.

State-of-Art

Open the graphical front end, and you are welcomed by a state-of-art interface divided into four areas. On the far left, the program offers a vertical buttonbar at the edge of the window, which you can use to configure the application modules.

To the right is a column with the *Network Rating*. This is where you specify how restrictive Portmaster is – the *Trusted* option is the default. When the *Untrusted* and *Danger* options are selected, the tool applies more restrictive filters. In addition, the application displays status and event messages in this column.

The rightmost *Network Monitor* column lists the individual connections from the computer to the Internet. The largest column on the far right of the program window shows the details for each connection. Clicking on the desired application lets you display detailed information. The window then opens a list view with more detailed information about the individual connections (Figure 3).

Beyond the plain-vanilla status and connection display, the Portmaster interface gives users the option of editing individual

External Sources

All Portmaster's external sources are listed in the sources.yml.

Currently, the Portmaster cannot yet ingest custom sources directly. If you know about a great source, please open an issue so we can discuss adding it.
If you came here for the sources, also check out filterlists.com.

Filter Lists

These lists are managed in this repo:

- ads.txt: Services that serve ads and track their audiences.
- analytics.txt: Services that provide visitor analysis/profiling.
- fraud.txt: Services that scam people.
- malware.txt: Services that are (ab)used for attacking devices through technical means.
- phishing.txt: Services that engage in credential fishing.
- telemetry.txt: Services that collect application telemetry.
- tracking-other.txt: Services that are believed to serve ads or track users, but their exact use is unknown or not categorized.
- securedns.txt: Services that provide secure DNS resolving. Used for bypass prevention.
- securedns-ip4.txt: Same, but IPv4 addresses.
- securedns-ip6.txt: Same, but IPv6 addresses.
- p2p.txt: Services that provide STUN, TURN, ICE or similar services that expose the user's IP address and enable peer to peer networking behind NAT. Used for advanced privacy protection.
- p2p-ip4.txt: Same, but IPv4 addresses.
- p2p-ip6.txt: Same, but IPv6 addresses.

Figure 2: Information on the Portmaster filter lists is available on GitHub.

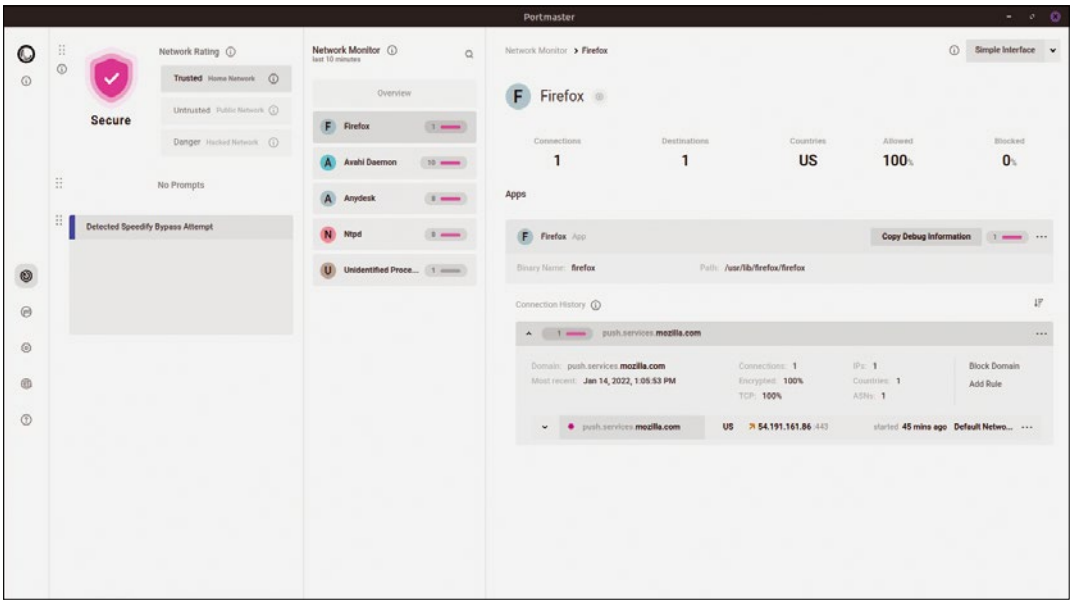


Figure 3: Portmaster's graphical interface provides information on all connection details.

connections. To do so, click on the desired application in the overview and then on the corresponding entry in the *Connection History*. Portmaster then displays all connections in the form of a table.

Use the *Add Rule* button to make further settings by defining new rules. In the subsequent dialog, you can enable or disable existing filter lists. Users with a little more networking knowledge can add their own custom rules using iptables syntax. Portmaster lets you define which data packets will pass through or be blocked (Figure 4).

You can set individual rules for all applications with Internet access, regardless of whether the application is currently running. Portmaster automatically detects all applications with network access and displays them in a list. Access this list by clicking the second button from the top on the far left of the vertical buttonbar. All apps then appear on the right, with Portmaster distinguishing between active and inactive applications (Figure 5).

Settings

To modify the global settings for Portmaster, click on the gear icon in the vertical buttonbar. The *Global Settings* dialog pops up on the right of the window, and you can use the convenient sliders to configure settings sorted into three connection categories.

For example, you can ignore DNS servers operating in your own intranet in larger environments, although Portmaster by default does not use them for name resolution until you reach the *Untrusted* level. You can also extend this setting to the *Trusted* level, network access via a secure intranet, using the slider.

SPN

Portmaster provides tunneled access to the Internet using the SPN. This network is an alternative to traditional VPN services, but it has an added technical feature. The SPN doesn't just route data packets through a tunnel between client and server; it also uses an onion structure similar to the TOR network, with the packets passing through multiple servers.

The SPN is currently still in the alpha phase, which is why Portmaster disables the service by default. Like public VPN services, the SPN requires separate access. The developers offer a simple, commercial subscription model, with an unlimited transfer quota for up to five devices at EUR9.90 (~$10) per month or EUR99 (~$103) per year.

To activate the SPN network, just click on the second button from the bottom in the vertical buttonbar on the left. A login dialog with some information now appears on the right, and you can use it to switch Portmaster to SPN mode. On top of this, in the dialog below *Global Settings*, you also need to move the slider in the *SPN* group from *Off* to *On*. After doing so, all connections are established via the SPN (Figure 6).

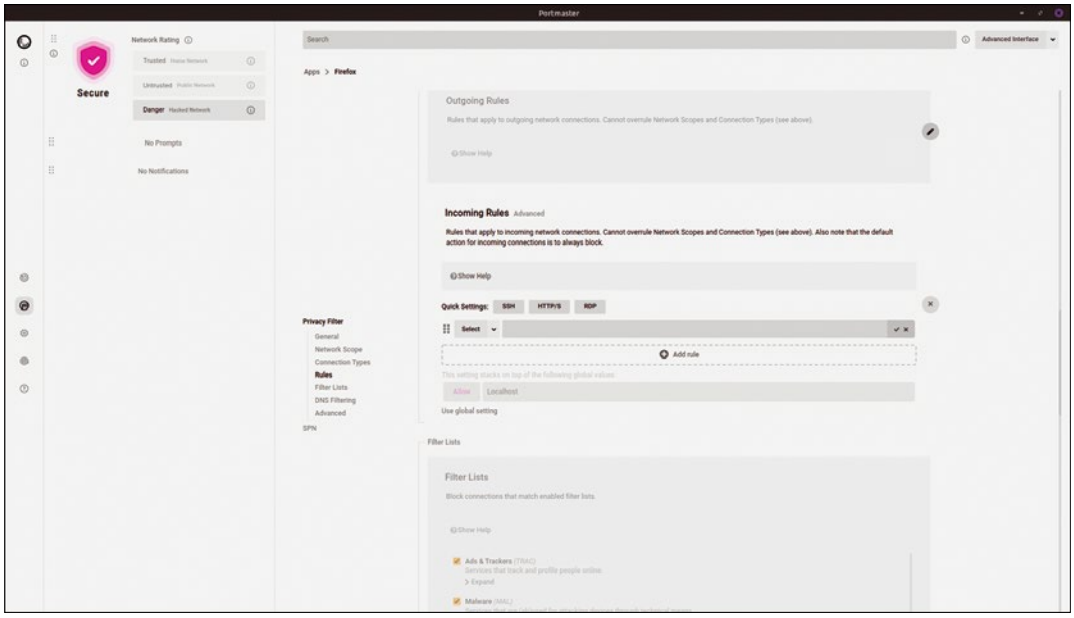


Figure 4: You can define rules for data transfer as you would for an iptables firewall.

Rapid Access

For quick access even when the app is closed, Portmaster gives you an icon in the desktop environment's system tray area. Left-clicking on the icon lets you quickly switch between the individual operating modes using the radio button. In addition, you can open the application window or disable Portmaster in the icon's context menu.

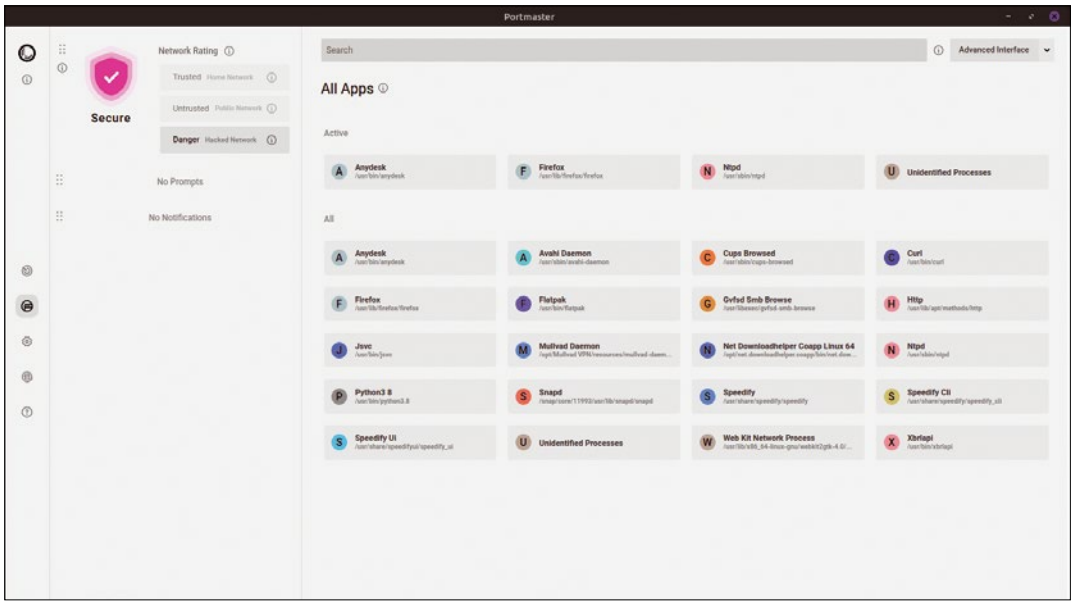


Figure 5: Even at the level of the individual programs, you can manage who the apps are allowed to send data to.

Conclusions and Outlook

Portmaster helps you put an end to dubious data transfers from your home computer to software vendors or advertising networks. The sophisticated blocking features not only use filter systems to prevent unwanted data uploads, but also prevent transfers to third parties at the application level if required. This helps you keep track of data traffic and makes it more difficult for trackers and advertising networks to spy on your surfing habits.

The Portmaster/SPN project, which is still at an early stage for such a technically demanding application, has by no means implemented all the planned features. If you're interested in trying Portmaster, you'll find Linux installation instructions at the project website [5]. The developers have also published a detailed roadmap on their website; you can check out the roadmap to find out when a function that you need, such as anonymization on the Internet using the SPN, will be integrated [6]. Portmaster is an effective tool for anyone wanting to keep their private affairs genuinely private. ■■■

Info

[1] Portmaster: *https://safing.io/portmaster/*
[2] Portmaster on GitHub: *https://github.com/safing/portmaster/*
[3] Information about the SPN: *https://safing.io/spn/*
[4] SPN alpha status: *https://docs.safing.io/spn/alpha/status*
[5] Instructions and compatibility list: *https://docs.safing.io/portmaster/install/linux*
[6] Roadmap: *https://safing.io/backlog/*

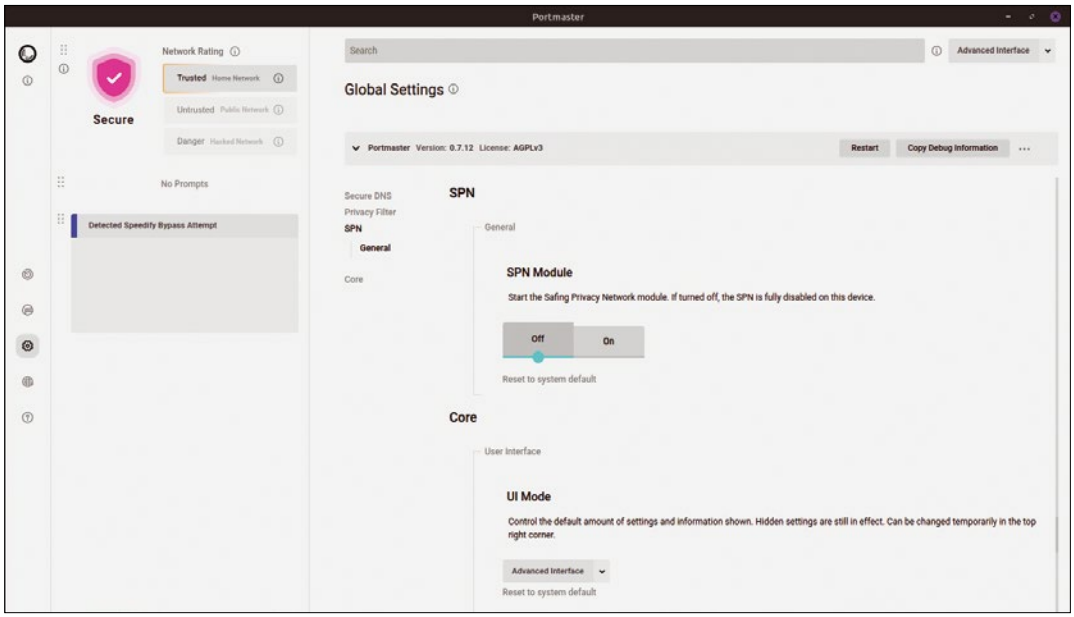


Figure 6: Portmaster also comes with the SPN, a VPN-like service that routes data through a multilayer network similar to Tor.

The Tor Network: Tools for private and secure browsing

Passing Secrets

The Tor project supports a formidable collection of tools for protecting your privacy on the Internet. We'll give you some background on Tor and help you get started with the Tor Browser.

By Bruce Byfield and Joe Casad

Computer users leave broad trails across the Internet. The websites you visit, your interactions, your purchases, your common passwords if you are careless – everything you do online – can be noted and used against you for purposes that range from the annoying to the dangerous, depending on your circumstances. Fortunately, a growing number of applications exist to restore your privacy and security, and the most mature of these is the Tor Browser [1] (Figure 1).

Tor (short for The Onion Router) is a modified version of Firefox designed to hide your trails on the Internet. Tor obscures your electronic trail by routing your interactions through several servers and encrypting your actions each step of the way. Tor's network of servers is decentralized, making your communications even harder to track down. Over the last two decades, several features have been added to the basic browser, providing a defense in depth against privacy and security intruders.

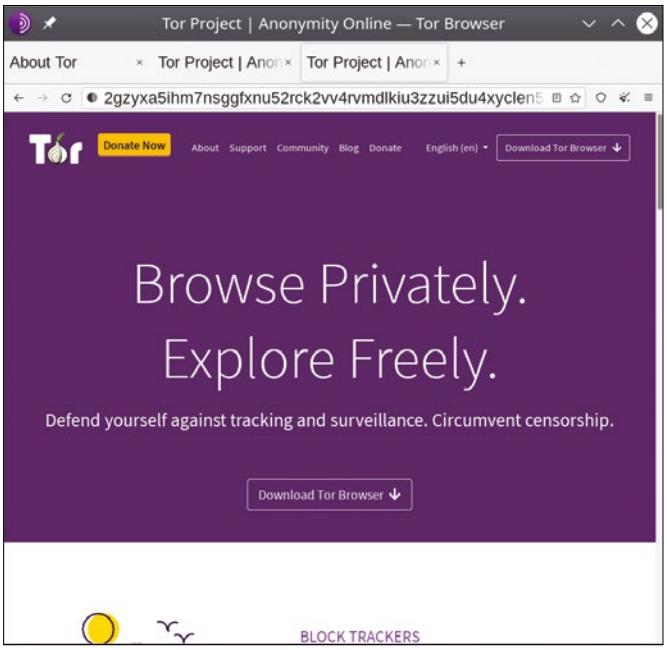
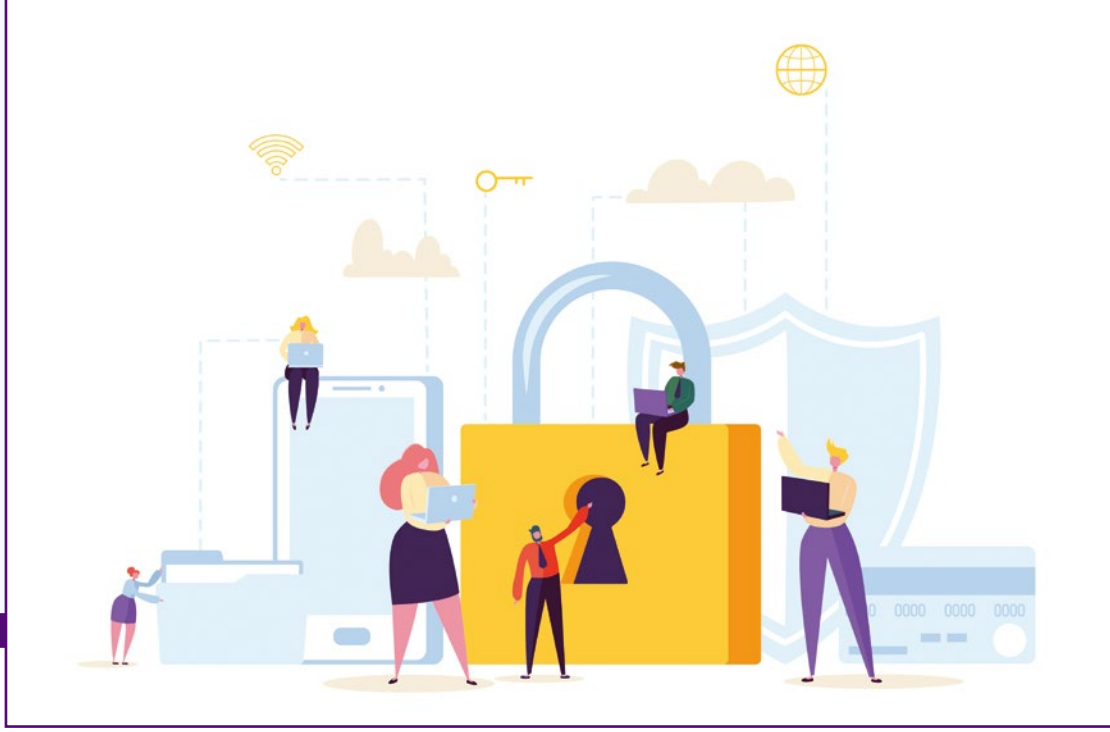


Figure 1: The Tor Browser is one of the simplest and most effective ways to preserve your online privacy.

The concept of onion routing was originally developed by the US Navy in the 1990s as a way of securing communication over the Internet. The Tor project was launched in 2002. In 2004, the Navy released the code under a free license, and the Electronic Frontier Foundation (EFF) became an early financial supporter of the project. In today's world, users in the great western democracies think of *privacy* as protection from ad trackers and big data aggregation, but the original vision for Tor was very much tied to the idea of providing safe communication for dissidents in authoritarian countries. The Tor project is proud of its contribution and support for the Arab Spring movement in 2010, and Tor has also supported several high-profile whistle blowers, including the famous Eric Snowden. Other humanitarian groups have backed the Tor project for its potential for bringing free speech to users in repressive countries, including Human Rights Watch and the US government's Bureau of Democracy, Human Rights, and Labor.

The privacy offered by Tor is a powerful thing, and it should come as no surprise that criminals have also used Tor to hide their activities. Ransomware attacks often use the Tor network for ransom payments, and the so-called dark web, with its illicit sales of drugs and stolen credit card numbers, makes extensive use of Tor to let its users operate anonymously. The Tor developers acknowledge that some criminals have used their technology for illegal ends, but they insist that "The majority of our users use Tor in a responsible way."

Some Tor users choose to do so out of principle – the belief that everybody has the right to control access to their own data. Others want to avoid the nuisance of being dunned by retailers or enlisted in a cause, or targeted by identity thieves. Still others want to bypass censorship restrictions placed on their browsing by their companies or countries. Others are victims of abuse in hiding, or whistle blowers revealing corporate or government corruption. The Tor project site has a page detailing the growing list of legitimate reasons for using Tor [2] that establishes that privacy and security are increasingly mainstream concerns – and they are concerns that most operating systems do not adequately address. Tor's purpose is to correct these omissions and to give its users greater peace of mind.



Routing in Tor

The Tor network is a collection of servers running software that allows them to participate in *onion routing*. Onion routing has existed longer than the Tor network and is a more general term – the Tor network has a particular set of protocols based on the onion routing concept.

The details are quite complex, but the basic idea is that the Tor browser client system (called the *originator*) routes a message through a series of Tor relays (see Figure 2). The message is encrypted in layers, so that each relay along the circuit only knows which node sent it the message and where to forward it next. The relay doesn't know the contents of the message or who originated the message. The exit node at the end of the circuit knows the final destination of the packet but doesn't know the source. The entry node at the beginning of the circuit knows the source but not the destination.

As you can see from Figure 2, the key to making this process work is assembling a message that allows each node along the path to know only what it needs to know and nothing more. The Tor client builds this message in layers (Figure 3), which are peeled off one step at a time like the layers of an onion (hence the name *onion routing*).

To build this many-layered message (called the *onion*), the Tor client must obtain a symmetric session key from each of the nodes along the circuit. These keys are obtained through a series of public key (Diffie-Hellman) connections among the nodes of the circuit that forward the encrypted session keys back to the client.

The client encrypts the message it intends to send to the destination server in the session key for Node 3. This encrypted message is then encrypted with the session key for Node 2, and this message is encrypted with the key for Node 1. When Node 1 receives the message, it knows the sender, but it doesn't know if the sender is the originator or just another link in the chain. Node 1 decrypts the outer message with the Node 1 session key and learns it is supposed to send the message to Node 2, although it can't see the contents or the eventual destination. Node 2, in turn, decrypts the message with the Node 2 session key and learns it should send the message to Node 3. Node 3 decrypts the final layer and learns that it must send the message to the destination server. Node 3 does not know who originated the message, but it knows it received the message from Node 2 and remembers that fact, so it will know how to forward the reply back from the server.

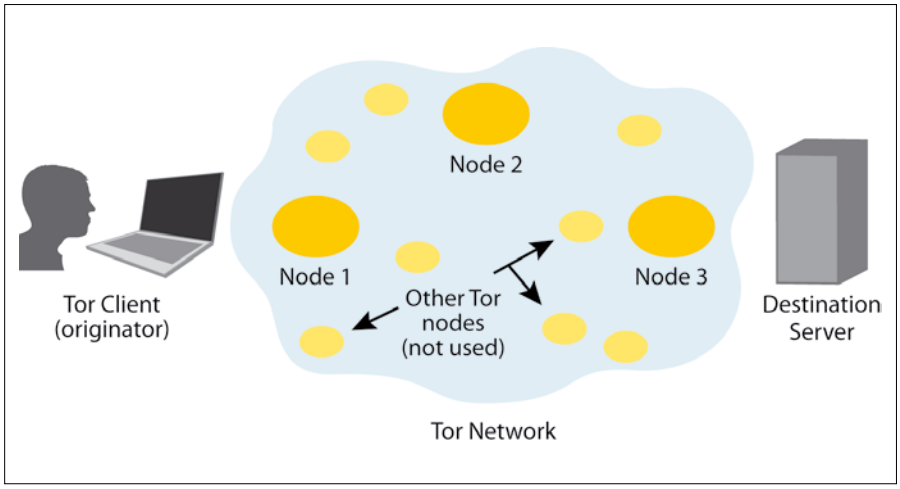


Figure 2: Tor routes the message through a series of routers. No link in the chain knows both the source and the destination of the message.

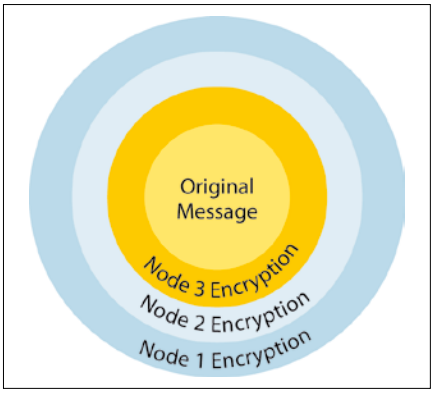


Figure 3: The message is enclosed in encrypted layers. Each node can only decrypt one layer, which reveals information on where to send the packet next.

In the response from the server back to the client, a similar process occurs in reverse. Each node adds a layer of encryption, obscuring the source of the response, and the client at the end of the chain unwraps all the layers using the previously obtained session keys.

As you can see, this process requires many steps and computations, so the Tor network is significantly slower than ordinary Internet traffic.

Installing Tor

Tor releases are available in most distributions. However, except in distributions with rolling releases of frequent security updates, the version is likely not to be the latest and might therefore not be secure. In the past, Ubuntu in particular has lagged so far behind the latest version that the Tor Project permanently warns against using it. Instead, Ubuntu and other Debian derivatives can set up the Debian repository maintained by the Tor Project [3].

Any distribution can download Tor directly from the project site [4]. In addition to Linux, downloads are available for Android, macOS, Windows, and source code, as well as 36 languages other than English (Figure 4). The uncompress download will have the name `tor-browser_LANGUAGE-LOCALE` – for instance, `tor-browser_en-US` for the American English version. Descend the directories and click on *start-tor-browser.desktop*. You can register Tor as part of the desktop environment with the command:

```
./start-tor-browser.desktop --register-app
```

To make Tor accessible to the entire system, uncompress or move the download to `/opt`, changing permissions as needed.

Next, run `start-tor-browser.desktop`, which opens the Tor Launcher (Figure 5). Configure any network settings, and, if you choose, elect to have Tor connect automatically in the future. Clicking the *Connect* button for the first time will connect in 15-30 seconds, but it will take only a few seconds later. Start by looking at the online help in the upper-left corner, where, among other things, you can adjust the default privacy and security settings (Figure 6), many of which will be familiar if you have previously used Firefox. As always, your choices are apt to be a balance between security and privacy on the one hand and convenience on the other hand. Finding the settings you can tolerate may be an ongoing experiment. What you have is a modified Firefox browser, but remember not to add any extensions not specifically supported by the Tor Project, since they can potentially compromise security and privacy. As convenient as Firefox extensions can be, there are simply too many to vet. Similarly, do not change any of the default configuration settings, which have been carefully configured.

Once connected, the Tor browser can be used like any other version of Firefox. You can configure the browser to automatically use hidden `.onion` sites where available. The browser uses DuckDuckGo to anonymize searches, including searches on Google. For the latest protection, you should also check frequently for updates.

Features Specific to Tor

While you are using the Tor Browser, you should be aware of its unique features, located in the upper-right corner of the window. Right next to the field for entering URLs is a button for adjusting the Security Level, using the same window available from the Tor Launcher's online help. The broom icon next to it restarts Tor, giving you a new temporary identity. The third button is the standard Firefox menu, modified for Tor. Under *Add-Ons and Themes*, you will find the add-ons for use with Tor. HTTPS Everywhere (Figure 7) is a collaboration between the Tor Project and the EFF that, if possible, forces sites to display using the encrypted HTTPS protocol rather than the unencrypted HTTP. A second add-on, NoScript (Figure 8), allows you to set how and when JavaScript runs, both generally and on individual pages. Both add-ons are essential parts of the Tor Browser's functionality. Currently, they are the only add-ons you should use with the Tor Browser.

Note that if you are looking at old documentation, you may come across a reference to *TorButton*. *TorButton* is an obsolete feature that allowed the Tor Browser to be turned off, leaving only the features of Firefox. Although convenient for some users,

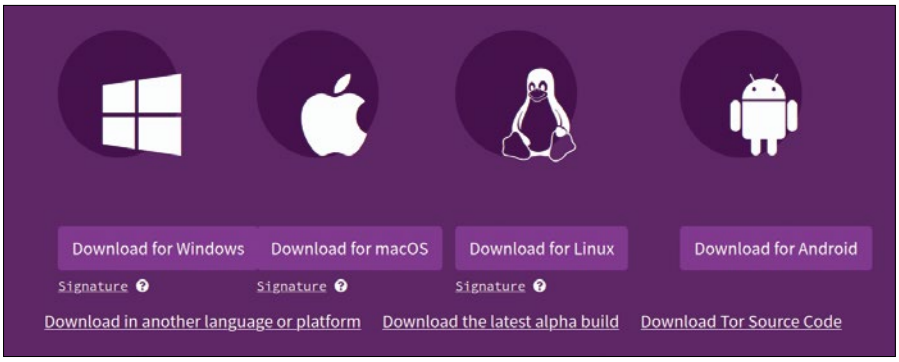


Figure 4: The Tor Browser supports multiple operating systems.

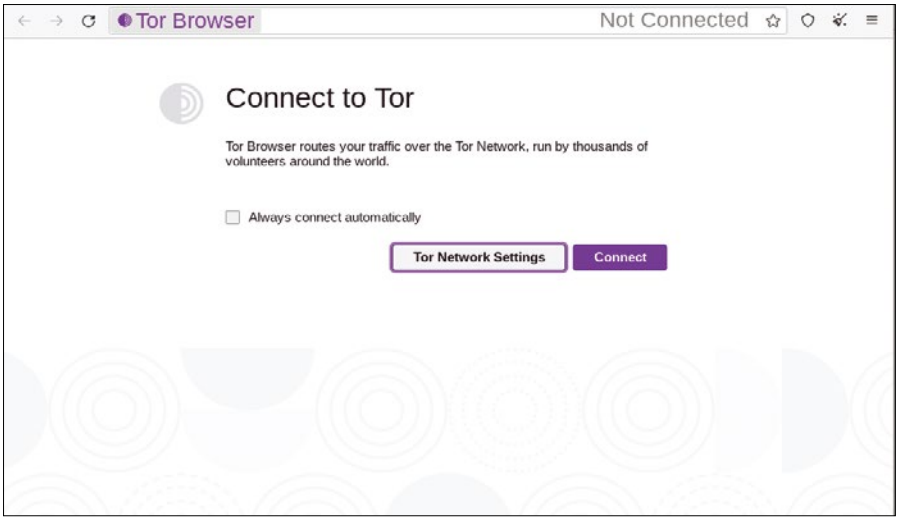


Figure 5: The launcher prepares the Tor Browser for use.

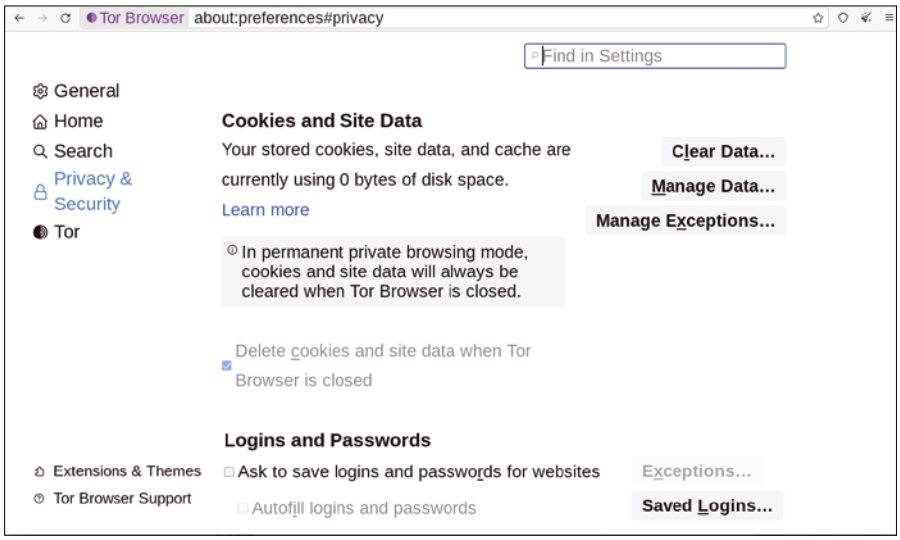


Figure 6: The Tor Browser uses many of the existing Firefox settings.

TorButton was discontinued because the Tor project is too small to keep up with Firefox's steady stream of releases. Should you somehow find a version of Tor that includes *TorButton*, you have an obsolete release and should upgrade immediately.

Notice, too, that because your identity changes each time you start the Tor Browser, you will see a notice about cookies each time you open a page that uses them (Figure 9). Your choices will not matter, because Tor deletes all collected cookies when it closes.

The Tails Distribution

Used as described, Tor will generally allow for anonymous browsing. However, Tor is tightly focused on its stated purpose and does not audit the system it is installed on. Should the system have privacy or security weaknesses, Tor will not protect you from them. For this reason, if you want a greater guarantee of privacy or security, consider using the Tails (The Amnesic Incognito Live System) distribution, one of the Tor's projects major sub-projects, and the only one endorsed by Edward Snowden. As the full name suggests, Tails allows you to browse anonymously and without storing any permanent information.

Tails is a distribution designed to run the Tor Browser from an exterior drive. Because Tails is a separate operating system that is secure in itself, it is unaffected by vulnerabilities of any installed operating system. Once shut down, it also leaves no trace. Moreover, because it is portable, Tails can be used on any computer simply by adjusting the BIOS to boot from an exterior drive. In addition, Tail's detailed documentation is an ideal place to learn privacy and security issues.

Relays

As you can tell from the design of the Tor network, the relay nodes are a precious commodity that are at the heart of the system. The more relay nodes you have available for forwarding traffic, the lower the system load and thus the faster the traffic. More nodes also makes the system more resilient to outage and makes it harder to compromise.

You can operate the Tor Browser without being part of the relay system. However, the Tor network is always looking for volunteer relay operators. The Tor project has a web page with resources for relay operators. You'll need some basic system administration skills to run a relay, as well as sufficient bandwidth. According to the Tor project, a non-exit relay should be able to handle at least 7,000 concurrent connections, which is too much for most consumer-grade routers. According to the Tor website, "Fast exit relays (> = 100 Mbit/s) usually have to handle a lot more concurrent connections (> 100k)."

Hidden Services

The Tor network allows Tor clients and relays to offer *hidden services* that are visible only through Tor. In other words, you can run your own web server, SSH server, or other service without revealing your IP address. Tor hidden services have the pseudo top-level domain `.onion`.

The *dark web* is actually a collection of web servers using Tor's hidden service feature. Dissidents in repressive countries

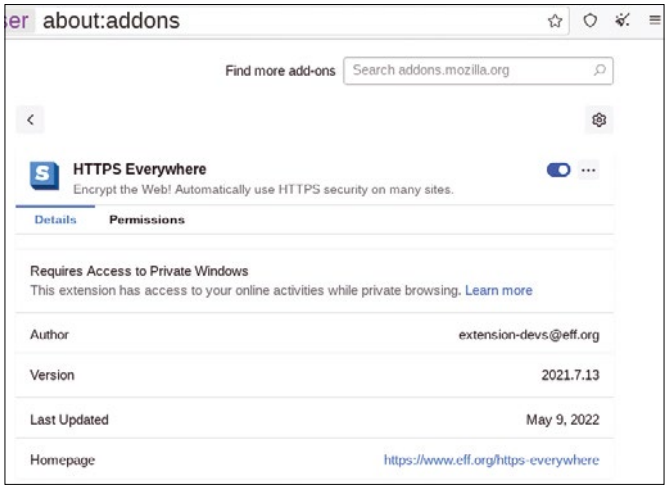


Figure 7: HTTPS Everywhere is an add-on to default to encrypted HTTPS whenever possible.

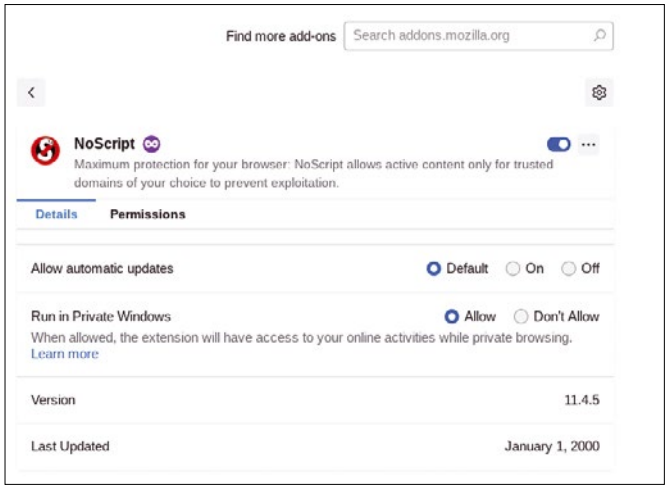


Figure 8: With NoScript, you can control the use of JavaScript on individual pages.

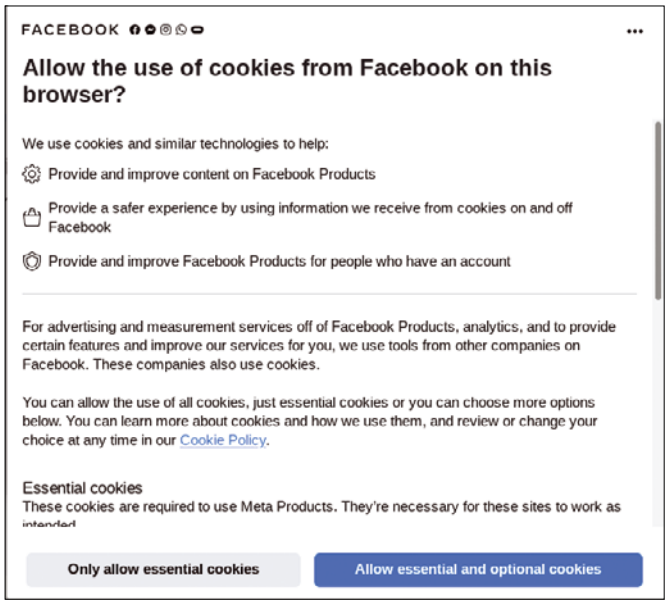


Figure 9: Because you are browsing anonymously, be prepared to see plenty of cookie notices.

use hidden web servers to operate blogs and information sources free of government surveillance. See the Tor project website for more on configuring and launching a hidden service.

Limitations

The Tor browser is a powerful tool for anonymity on the web, but it still has some limitations that users should be aware of. Two obvious ways to compromise the Tor network are:

- Break the encryption (not so easy if the encryption is done properly with up-to-date techniques).
- Exploit a flaw in one of the system components. For example, the Tor browser is based on Firefox, which occasionally has vulnerabilities that require patches and updates.

Beyond these potential attacks are a range of other concerns that are less obvious if you're not an expert. In fact, security researchers make a point of looking for flaws in the Tor network, and the Tor project actually welcomes it, because transparency is one of their values, and if there is a way in, they would rather know about it than have it go unreported.

Governments have also been busy with trying to break into the Tor network, and they are a little less willing than academics to publish the results on how they do it. Several high-profile crackdowns on elements of the dark web have occurred through the years. In 2014, for instance, a coalition of government agencies took control of several Tor relay nodes, resulting in the takedown of several black market sites, including Silk Road 2.0. No one at the time, including the Tor developers, knew exactly how the agencies accomplished this attack, which became known as Operation Onymous, but the Tor developers published a blog post with some ideas about how it might have happened, suggesting things like:

- Operational security – stolen passwords and rogue users (the same way other servers on the Internet are attacked)
- Flaws in the web application that could have led to an SQL injection or other similar attacks.
- Bitcoin de-anonymization – according to experts, the Bitcoin transaction process can reveal information that could potentially de-anonymize a Tor user
- Attacks on the Tor network itself

Tor network attacks can take various forms. In the report following Operation Onymous, the Tor developers state "Some months ago, someone was launching non-targeted deanonymization attacks on the live Tor network. People suspect that those attacks were carried out by CERT researchers. While the bug was fixed and the fix quickly deployed in the network, it's possible that as part of their attack, they managed to deanonymize some of those hidden services" [5].

Much of the research on breaking the anonymity of the Tor network has focused on the entry and exit nodes. You might not know what happens inside the network, but if you watch the exit node, you can still gain insights on the traffic. The Tor project itself warns that it cannot protect users against a so-called *end-to-end timing attack*. "If your attacker can watch the traffic coming out of your computer, and also the traffic arriving at your chosen destination, he can use statistical analysis to discover that they are part of the same circuit" [6].

These elaborate attacks that require constant monitoring of all possible entry and exit nodes, infiltration of Tor relay systems, or statistical studies of random nodes across the Internet require enormous resources, and if you're just using Tor because you don't want to get tracked by an ad tracker, you're probably safe. However, if you are truly using Tor because you are trying to avoid capture by a well-funded state actor, it is best to at least be aware of these limitations.

Despite the occasional sting like Operation Onymous, the Tor network continues to provide anonymity for the vast majority of its users.

Conclusion

The Tor developers view privacy as a basic human right that should be available to everyone and must be protected on principle. The project places a high premium on the safety of its users. They acknowledge that some Tor users might access the platform for illegal activities, but they maintain that most users have a legitimate purpose. They argue that Tor is a tool, like any other tool, that can be employed for good or evil. A criminal could use a car, or a gun, or the Internet itself to commit a crime, but people generally acknowledge that these tools also have legitimate uses.

If you are a privacy-conscious user, and you are weary of depending on browser plugins and your clunky browser security settings to protect you from surveillance, this might be the time to explore the Tor Browser. In an era in which privacy and security are a forever increasing concern, Tor is an important option for defending yourself and your data. ■■■

Info

[1] Tor Browser: *https://www.torproject.org*
[2] Reasons for using Tor: *https://2019.www.torproject.org/about/torusers.html.en*
[3] Ubuntu and Debian with Tor: *https://support.torproject.org/apt/*
[4] Tor Downloads: *https://www.torproject.org/download/*
[5] Attacks on the Tor network: *https://blog.torproject.org/thoughts-and-concerns-about-operation-onymous/*
[6] About Tor page: *https://2019.www.torproject.org/about/overview.html.en#stayinganonymous*

STORAGE DEVELOPER CONFERENCE

SDC 22

BY Developers FOR Developers

- **Blockchain and Storage**
- **Computational Storage**
- **DNA Data Storage**
- **Persistent Memory**
- **Storage Architecture, and more**

September 12-15, 2022
Fremont Marriott Silicon Valley

Don’t miss this event for the best in technical discussions and education on the latest storage technologies and standards.

A SNIA® Event

www.storagedeveloper.org

The fight for Init Freedom

DEVUAN

Devuan, with its promise of Init Freedom, provides users an alternative to systemd as an init process. *By Bruce Byfield*

Long-time Linux users may remember a time when Debian was viewed as a collection of anarchists, with radical ideas about voting and decision-making. At times, Debian was even the lone dissenter among distributions about decisions made by the Free Software Foundation. However, over the years, Debian has developed its own hierarchy along the way to becoming the source for some two-thirds of active distributions. Today, the Debian derivative most reminiscent of early Debian is Devuan [1], which forked from Debian in 2014 over how decisions were made and the technical connotations of using systemd.

Recently, two Devuan developers – fsmithred, who builds the live images and helps with support, and golinux, the community manager – took the time to recall Devuan's past and why their issues are still relevant today. Because Devuan lacks a formal hierarchy, they emphasize that their remarks are "unofficially official."

In 2014, major Linux distributions were transitioning from SysVinit to systemd as an init process – init being the first process to start on a system and the one that manages other processes. Ubuntu had started using Upstart a decade earlier with little controversy. By contrast, systemd was controversial from its earliest days. To start with, systemd is much more than an init system. Rather, as contributor dasein described on the Debian User Forums, "calling systemd an init system is like calling an automobile a cup holder" [2]. That is, while systemd includes the functions of an init system, dasein said systemd is also "an effort to recreate large portions of existing userspace (including login, job scheduling, and networking, just to name a few) inside a single process traditionally reserved for the sole purpose of starting *nix userspace. (Just in case it isn't clear, there is a huge difference between starting userspace (init) and being userspace (systemd).)"

From this perspective, not only is systemd overkill, but it is a violation of the basic tenet of Unix development that an application does only one thing and does it very well. As Christopher Berry stated in the Linux Kernel Archive, this philosophy is what makes Linux "a collection of simple modular components that could be plugged together at will to do real work" [3] – an operating system that is both flexible and accessible. Just as importantly, a modular structure allows the pieces to be assembled in different ways, so that

Photo by Fuu J on Unsplash

each distribution can be unique. By contrast, systemd imposes a structure on all Linux systems that reduces variety – which is convenient in some ways, but needlessly limiting in time-honored ways.

As expected, the Debian mail forums debated these perspectives extensively. Unsurprisingly, the discussion culminated in a General Resolution among Debian users, with many Debian office users favoring systemd. The winning option was to use systemd, but at the same time, a more general resolution to favor systemd placed last – a decided ambiguity. Although rarely stated in so many words, much of the dissatisfaction implied that the decision to use systemd was imposed by the Debian hierarchy upon the general membership.

Whether this implication was valid is besides the point. Many believed it was. On November 24, 2014, the Devuan fork was announced. The intention was "to produce a reliable and minimalist base distribution that stays away from the homogenization and lock-in promoted by systemd" [4].

Introducing Init Freedom

Rather than being seen as simply an anti-systemd project, Devuan calls its position Init Freedom (Figure 1). The name invokes Richard Stallman's four essential freedoms, although the idea itself might seem less basic. Devuan's Init Freedom page [5] simply defines the idea as being "about restoring a sane approach to PID1 [init] that respects portability, diversity, and freedom of choice," assuming that the value of these goals is self-evident.

In practice, Init Freedom means supporting a choice of init freedoms. Although systemd advocates often maintain that supporting multiple init systems would make packaging more difficult, from its first release, Devuan has continually added init alternatives without any apparent difficulties. Today, in addition to the default SysVinit, Devuan lists six alternatives: OpenRC, runit, sinit, s6, 66-devuan, and Shepherd, and it is open to considering others. Fsmithred suggests that most people simply use sysVinit, although OpenRC and runit, which use SysVinit scripts, are also available. Scripts are also being developed for runit and to extend usability of other shipped alternatives. For those interested in learning more, discussion can be found on the Dev1 Galaxy Forum [6] and on Devuan's IRC channel [7].

Figure 1: Devuan supports a choice of init alternatives.

In addition, the Init Freedom page lists other Linux distributions that offer systemd-free alternatives, as well as other Unices such as the BSD Family. DistroWatch also offers a search filter for distros without systemd – currently, 97, a total far higher than many might suspect, although it includes only a handful of major distributions, such as MX Linux, Alpine Linux, and KNOPPIX [8]. Devuan keeps in close touch with these distributions, especially on the Dev1 Galaxy Forum.

Fsmithred adds that, "We rely heavily on Debian. Most of the packages in Devuan are unchanged from Debian. We only fork packages that require systemd. There is collaboration between Devuan and Debian on a few packages, and we occasionally send bug fixes or patches upstream to Debian for packages we do not fork."

Beyond Init Freedom

Devuan is usually referred to in terms of Init Freedom – often with the obviously mistaken assumption that it is a lost cause. However, Devuan also features Docker images and community-developed ARM packages. Chimaera, the latest Devuan release, also includes an option to not install PulseAudio to enable simultaneous speech synthesis in both a graphical and console session. In addition, at least one Devuan-derivative exists, Maemo Leste, whose goal is "to provide a free and open source Maemo experience on mobile phones and tablets like the Nokia N900, Motorola Droid 4, Motorola Bionic, PinePhone, PineTab, Allwinner tablets, and more" [9]. Although Devuan might be a niche distribution, clearly it is a thriving one.

But can Devuan's cause become mainstream? It's not impossible. Linux is in an era in which large parts of it are being written. If PulseAudio can be replaced with PipeWire – which is currently happening – then systemd's obsolescence is not impossible, either. Meanwhile, for those who disagree with the majority, Devuan provides a workable alternative, while keeping the early spirit of Debian alive. ■■■

Info

[1] Devuan: *https://www.devuan.org*

[2] "Combating revisionist history" by dasein, Debian User Forums, February 25, 2015: *https://forums.debian.net/viewtopic.php?t=120652*

[3] "Open letter to the Linux World" by Christopher Berry, Linux Kernel Archive, August 12, 2014: *http://lkml.iu.edu//hypermail/linux/kernel/1408.1/02496.html*

[4] Devuan announcement: *https://www.devuan.org/os/announce/*

[5] Init Freedom: *https://www.devuan.org/os/init-freedom*

[6] Dev1 Galaxy Forum: *https://dev1galaxy.org/search.php?search_id=1415868092*

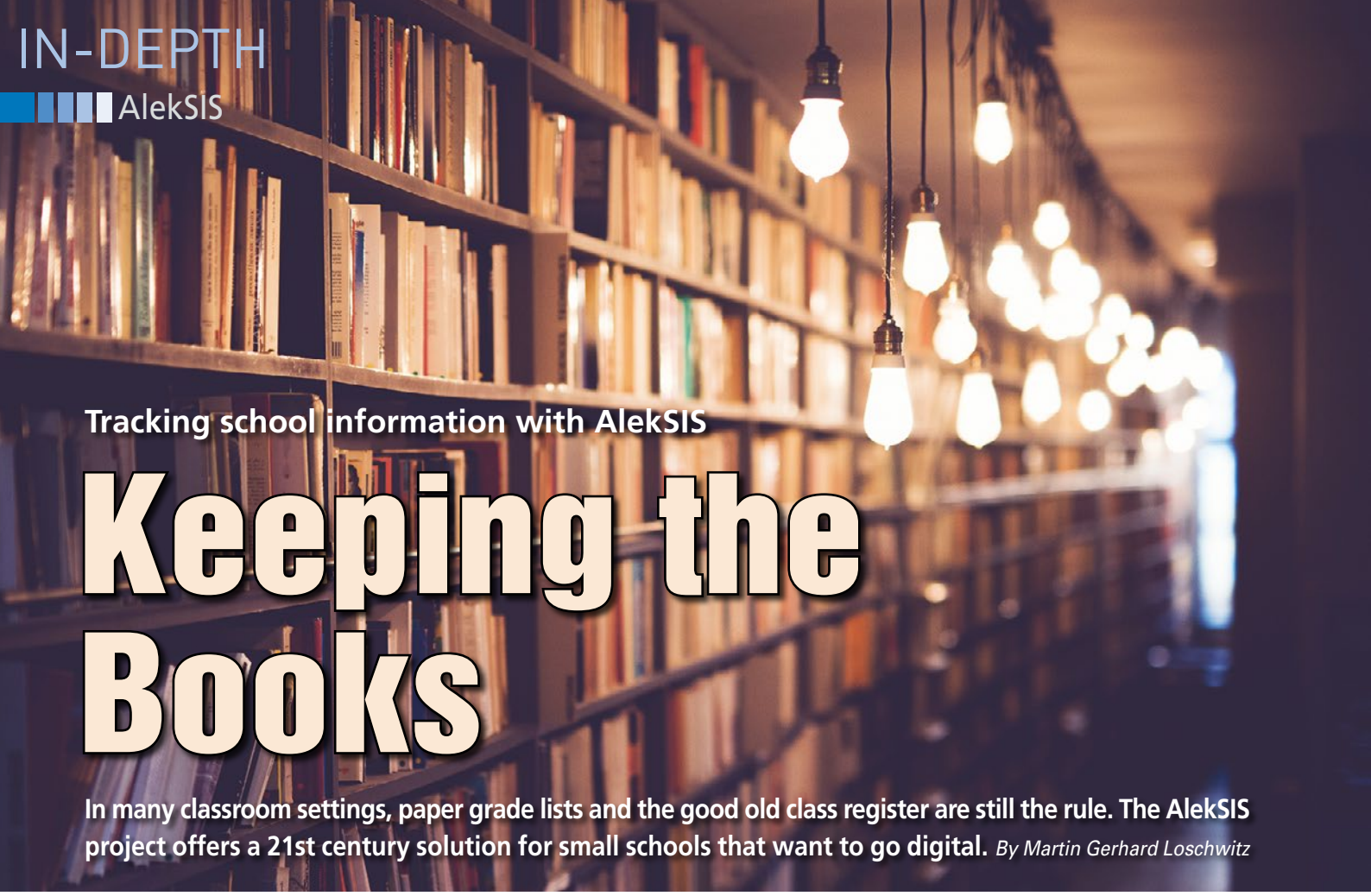
[7] Devuan's IRC channel: *http://reisenweber.net/irclogs/libera/_devuan/*

[8] Distros without systemd: *https://distrowatch.com/search.php?ostype=All&category=All&origin=All&basedon=All&notbasedon=None&desktop=All&architecture=All&package=All&rolling=All&isosize=All&netinstall=All&language=All&defaultinit=Not+systemd&status=Active#simple*

[9] Maemo Leste: *https://maemo-leste.github.io/*

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/*.



Tracking school information with AlekSIS

Keeping the Books

In many classroom settings, paper grade lists and the good old class register are still the rule. The AlekSIS project offers a 21st century solution for small schools that want to go digital. *By Martin Gerhard Loschwitz*

AlekSIS, a free school information system, aims to drag schools into the digital age. Currently in a functional state and ready for production use, AlekSIS [1] is a direct competitor of other tools with similar school-management features. However, because it is free software, AlekSIS helps you steer clear of the dreaded vendor lock-in. AlekSIS's creators have a declared goal of blowing the commercial solutions out of the water.

Designed by teachers in Germany, AlekSIS was originally developed for German high school settings, but the basic features, such as schedules, student and staff profiles, class rosters, and attendance tracking, are universal features of classrooms everywhere. AlekSIS offers a user interface in English and complete English versions of the AlekSIS Handbook and website.

Most large school districts in the English-speaking world are already using software for tracking students and classroom business, but many small private schools and charter schools, as well as training facilities and other independent institutions, are still tracking their classroom business on paper or using rudimentary tools that lack integration. In these settings, AlekSIS offers a sensible alternative.

Author

Martin Gerhard Loschwitz is a freelance journalist focusing primarily on topics such as OpenStack, Kubernetes, and Ceph.

Web-Based Access

Because some classrooms today still do not have a computer, the AlekSIS team knew from the outset that the software would have to be easy to use on mobile devices such as smartphones. As a result, AlekSIS's central interface is based on the principle of web-based access, with the web interface adapting to the displaying device. As a result, AlekSIS works just as well on an iPhone as it does on an Android tablet, a laptop, or a desktop computer.

This capability, known as a Progressive Web Application (PWA), offers more than is apparent at first glance. Because AlekSIS exclusively relies on a browser, it accommodates the capabilities of many operating systems to generate ad hoc dynamic mini-apps with their own icons from websites. If you want to have your own timetable (Figure 1) as a widget on your smartphone's main screen, AlekSIS is the way to go.

User-Friendly

AlekSIS places a high bar for clarity and ease of use. Because students from grade five up need to be able to use the interface just as easily as their teachers, the AlekSIS interface has been deliberately kept very simple. The menu is located on the left edge of the screen and gives users direct access to all of the software's functions. AlekSIS does not come with color schemes. If you need them, you can add them via the configuration. You can also adapt AlekSIS's appearance to match existing specifications (e.g., to store a specific logo in the software). Adhering to the form follows function principle for all the GUI adaptations, AlekSIS offers an easy-to-use, intuitive interface that meets modern standards in terms of user interface/user experience design.

Personnel and Room Management

Anyone who has ever had to deal with the administrative inner workings of a school will quickly realize where AlekSIS is headed. The solution aims to enable extensive digitalization of schools. Managing

Photo by Janko Ferlič on Unsplash

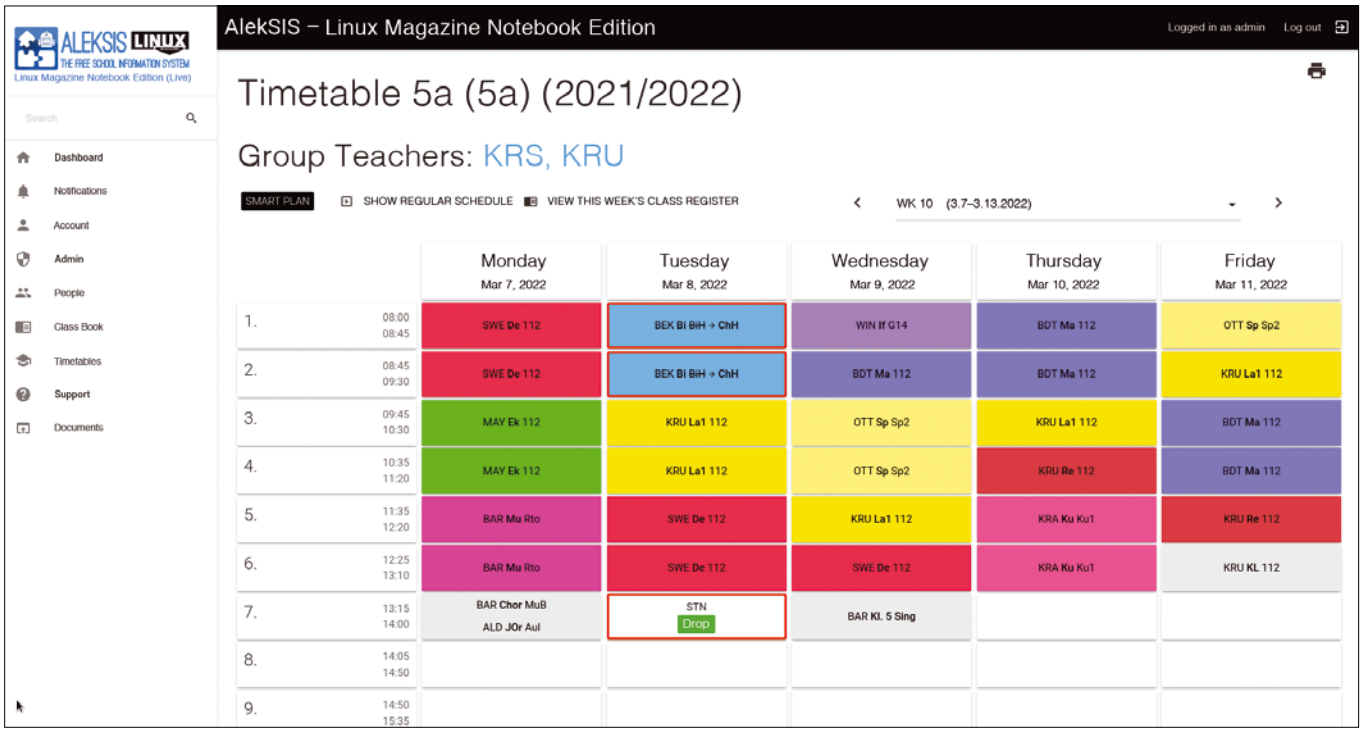


Figure 1: One of the most practical AlekSIS functions dynamically displays timetables for students and teachers based on available data. ©Dominik George, TecKids

teachers and students with, at least limited, self-service options is the first step.

AlekSIS lets you create a teacher database with each staff member's information using a large number of different parameters for each teacher. You can enter the subjects taught by each teacher, as well as access a calendar that lets you block periods in a weekly schedule when the teacher is not available. In addition, teachers can manage their absences in AlekSIS.

If required, AlekSIS can be linked to an existing LDAP or Active Directory, from which it then sources the user data. Otherwise, the solution maintains a local user database.

Just as it manages people, AlekSIS manages rooms and courses or classes. Rooms and classes need to be maintained centrally by the admin office, but AlekSIS cleverly combines the information it has already collected. After you create a class, you can assign students and teachers for individual lessons to it. In turn, a classroom management element can be linked with a teaching unit once it has been created by the admin office. Bringing the classroom and personnel elements together allows comprehensive mapping of the day-to-day schedule of everyone at the school.

Fielding Emergencies

In the hectic atmosphere of everyday life, AlekSIS also actively assists the school admin staff. For instance, you can solve relatively easily the case of a sick teacher with AlekSIS. Because the program offers a comprehensive overview of teacher schedules, an administrator can quickly see which potential substitute is currently available.

Digital Class Register

AlekSIS makes it far more convenient to check classroom attendance. Because each student in AlekSIS is entered as a separate administrative object, you can call up the class roster to centrally manage the student's absences. Adding a note stating whether or not an absence is excused simply becomes part of the entry. In addition, AlekSIS provides a separate field for late arrivals, which also become part of the student's record.

Grades

Thus far, AlekSIS does not implement centralized, digital grade management. The theoretical foundations for adding grades to the student profile are certainly in place, but the feature itself is currently missing. If AlekSIS continues to evolve, you can expect that grades will eventually be added. The privacy requirements for tracking grades might vary depending on the jurisdiction, and not every teacher will switch to fully digital grading. However, if AlekSIS succeeds in integrating grades with other school information, this will be a huge step in the right direction.

Conclusion

All in all, AlekSIS already offers a powerful solution for the digitalization of schools, despite being at a relatively early stage. Thanks to the software being completely open source, schools that rely on AlekSIS do not have to worry about being tied to a single commercial provider.

AlekSIS is not the first software to embrace the digitalization of schools, but many commercial alternatives are based on the idea of software as a service (SaaS), a comprehensive offering with support from the vendor. While this may be more convenient for schools initially, it also ties them to the vendor and its prices in the long term.

If you are looking for a powerful front end for the digital administration of a school that is easy to use and implement, keep a watchful eye on AlekSIS. ■■■

Info

[1] AlekSIS: *https://aleksis.org*

Locking down the Thunderbolt interface

Lightning Protection

The Thunderbolt interface supports extremely fast data transfer rates, but be careful about what you plug into your port, because Thunderbolt devices access memory directly. We'll show you some Linux tools for locking down your Thunderbolt interface. *By Thorsten Scherf*

The Thunderbolt interface is an interface used for connecting peripheral devices to many modern computers. Thunderbolt connections (with the familiar lightning arrow symbol – see Figure 1) support fast transfer of audio, video, and other data over a single cable and can also charge devices connected through the same interface.

The Thunderbolt specification was developed by Intel in collaboration with Apple. Many users might think of Thunderbolt in the context of Apple hardware. Apple actually started shipping MacBook Pro models with the interface back in 2011 (see the box entitled "Thunderbolt Through the Years"). Thunderbolt has become a common feature on MacBook computers, as well as many other Intel-based systems.

Through the years, however, the power and speed of Thunderbolt has led to some security issues. Like other technologies that communicate with a system via PCI Express (PCIe), Firewire, or similar protocols, Thunderbolt supports direct access to system memory. Directly accessing memory enables fast data transfer rates, but it also poses a security risk, because many different components access memory at the same time, which creates the potential for a DMA attack. (A DMA attack involves unauthorized access to the system memory in order to read arbitrary data.)

Security concerns have led to a new approach with recent Thunderbolt versions. Some of the basic security features available in Thunderbolt 3 have been enhanced for version 4. Thunderbolt now uses the Intel Virtualization Technology for Directed I/O (VT-d) to provide protection against DMA attacks.

The Linux kernel supports Thunderbolt out of the box, but to use Thunderbolt 3's security features, you need to

Photo by Micah Tindell on Unsplash

Figure 1: Thunderbolt ports marked with the lightning logo.
CC By-SA 4.0 Wikipedia

have kernel version 4.13 or later. If you want to use I/O virtualization in combination with Thunderbolt 4, you'll need at least kernel 4.21. This article offers some tips on how to secure your Thunderbolt ports in Linux.

Security in Thunderbolt 4

Current Intel processors have an Input/Output Memory Management Unit (IOMMU) that supports I/O data virtualization. Each device connected via Thunderbolt 4 can therefore be assigned a private memory area, effectively preventing one device from accessing the data or memory area of any other. This feature is also known as direct memory access remapping (DMA-r). Further information on this can be found in an Intel white paper [1] on the subject of DMA protection with IOMMU.

To use I/O virtualization, you also need to enable the IOMMU option in the system BIOS. You can then use the Linux `dmesg` utility in a terminal to check whether the option is actually active (Listing 1).

Use the sysfs filesystem to configure the Linux kernel's Thunderbolt subsystem. A description of the individual configuration options is included in the Linux kernel documentation [2]. For example, to find out whether your Thunderbolt devices each use their own virtualized memory area, you just need to read the file called:

```
/sys/bus/thunderbolt/devices/↵
domain0/iommu_dma_protection
```

If the file contains a value of `1`, DMA protection based on VT-d is active. If the value is `0`, the IOMMU option in the BIOS might not be active, you might have an old kernel installed, or you are not using Thunderbolt 4 hardware. If you are not using Thunderbolt hardware, you can still fall back on the security features available with version 3.

Five Possible Security Levels

The Thunderbolt specification supports five different security levels: `none`, `dponly`, `user`, `secure`, and `usbonly`. You can set the desired level for the Thunderbolt interface in your system's BIOS. To find out which level is currently active, ask the Linux sysfs filesystem by typing:

```
cat /sys/bus/thunderbolt/↵
devices/domain0/security
```

If you want to use the `user` or `secure` security level, you first need to authorize a device to establish a channel between the device and your system. The `secure` level also creates a shared key that the device must use to authenticate itself against the system each time. This key is stored on the device itself and also in the sysfs filesystem.

You can authorize a device and store a key manually on the sysfs filesystem or use the bolt [3] tool instead. The bolt tool consists of a service (`boltd`) that interacts with the sysfs filesystem and makes the Thunderbolt devices registered there available to other applications via the D-Bus message bus. Use `boltctl` to manage your Thunderbolt devices. Calling `boltctl` only shows you the registered devices. You can then authorize a device using the following command:

```
boltctl authorize <UUID>
```

If the security level is set to `secure`, a key for the device is also generated and distributed. The software is available in most Linux distributions, but you can also download it from the GitHub repository [3].

Conclusions

Thunderbolt is great for quickly transferring large volumes of data: The downside is that the interface allows direct access to a system's memory. If you support Thunderbolt devices, additional security measures are very much recommended if you want to stop unauthorized devices from accessing your system and reading sensitive data. Starting in Thunderbolt 3, you can use security levels to ensure that a device needs to be manually authorized first before it can talk to your system. The current Thunderbolt 4 assigns a private memory area to each device, which stops the device from accessing the entire system memory through I/O virtualization. ■■■

Thunderbolt Through the Years

The first version of Thunderbolt had a maximum transfer rate of 10Gbps, with two bidirectional channels for data transfer. Two years later, version 2 arrived with a maximum speed of 20Gbps.

In 2015, version 3 replaced the Mini Display Port connector used in previous versions with a rotatable USB-C connector. Version 3 also saw the introduction of four PCIe wires for data transfer to enable transfer rates of 40Gbps. Version 3 could supply power to devices with consumption of up to 100 watts.

Today's devices use version 4 of the Thunderbolt technology. Version 4 does not support faster transfer rates, but it does offer some other interesting new features. For example, the USB4 standard is now used. Support for USB hubs in alternate mode is also very interesting; it lets you connect an 8K display, or optionally two 4K displays, several external hard disk drives, and a power supply to your computer with only one cable.

Listing 1: IOMMU Option in the BIOS

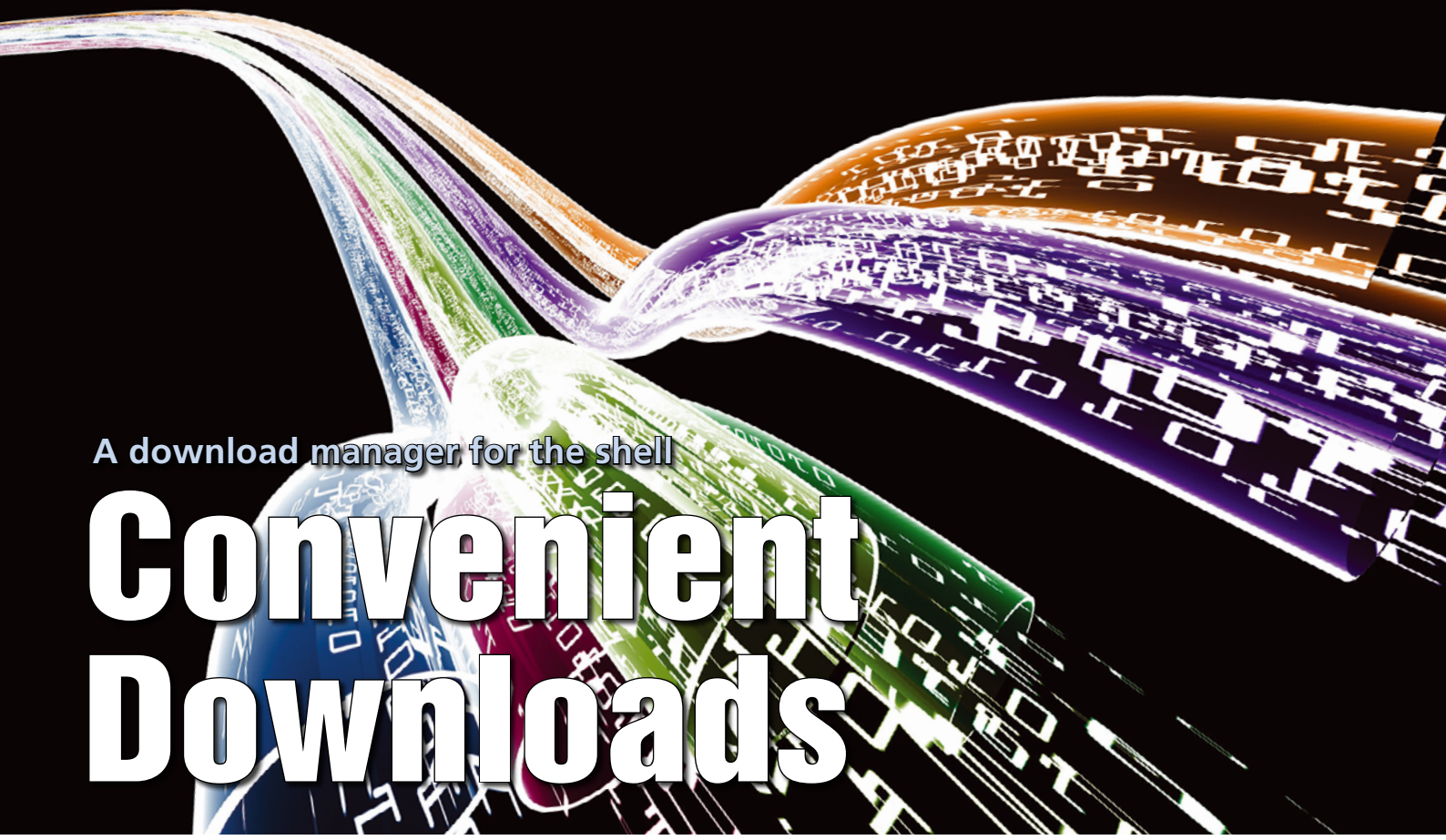
```
0.000000] DMAR: IOMMU enabled
0.301602] DMAR: Host address width 39
0.301603] DMAR: DRHD base:
  0x000000fed90000 flags: 0x0
0.301612] DMAR: dmar0: reg_base_addr
  fed90000 ver 1:0 cap 1c0000c40660462
  ecap 19e2ff0505e
0.301616] DMAR: DRHD base:
  0x000000fed91000 flags: 0x1
0.301621] DMAR: dmar1: reg_base_addr
  fed91000 ver 1:0 cap d2008c40660462
  ecap f050da
0.301624] DMAR: RMRR base:
  0x000000a869a000 end:
  0x000000a86b9fff
0.301626] DMAR: RMRR base:
  0x000000ab000000 end:
  0x000000af7fffff
0.301627] DMAR: RMRR base:
  0x000000a86cb000 end:
  0x000000a874afff
```

Info

[1] A Tour Beyond BIOS: Using IOMMU for DMA Protection in UEFI Firmware: *https://www.intel.com/content/dam/develop/external/us/en/documents/intel-whitepaper-using-iommu-for-dma-protection-in-uefi.pdf*

[2] Linux kernel Thunderbolt documentation: *https://www.kernel.org/doc/Documentation/ABI/testing/sysfs-bus-thunderbolt*

[3] bolt: *https://github.com/gicmo/bolt/*



A download manager for the shell

Convenient Downloads

A few lines of shell code and the Gawk scripting language make downloading files off the web a breeze. *By Goran Mladenovic*

Almost everyone downloads data from the Internet. In most cases, you use the the web browser as the interface. Usually, websites offer files for download that you can click on individually in the browser. By default, the files usually end up in your `Downloads/` folder. If you need to download a large number of files, manually selecting the storage location can quickly test your patience. Wouldn't it be easier if you had a tool to handle this job?

In this article, I will show you how to program a convenient download manager using Bash and a scripting language like Gawk. As an added benefit, you can add features that aren't available in similar programs, such as the ability to monitor the disk capacity or download files to the right folders based on their extensions.

Planning

Programming a download manager definitely requires good planning and a sensible strategy. Obviously, the download manager should download files, but it should also warn you when the hard disk threatens to overflow due to the download volume. Because most files have an extension (e.g., `.jpg`, `.iso`, or `.epub`), the download manager also can sort your files into appropriately named folders based on the extensions.

To build your own download manager, you will need two command-line programs that do not come preinstalled on modern Linux distributions by default: `xclip` and `lynx`. The `xclip` command-line tool monitors the clipboard, while `lynx` functions as a web browser for the terminal. Lynx also comes with several command-line options, making it usable as a link spider that offers many link display options.

During planning, the basic thing you need to consider is how to pass the download page's URL and links into the terminal. `xclip` and `lynx` can help you do this. This project also includes functions that handle specific subtasks, such as capturing and selecting the links. Even if you don't know in the beginning the content of these functions, I recommend creating a script that includes empty functions as placeholders for the time being (see Listing 1).

The framework in Listing 1 serves as a basis for building the individual functions one by one. Focus on a single function first rather than on the big picture. For each function, you need to consider, separately, what tasks the function handles, how many and what kind of parameters the function needs, and whether there are any return values.

For convenience, you can outsource parts of the script and then include the source code using dot or source notation such as

```
. outsourcedFunction
```

Listing 1: Script Framework Without Functions

```
#!/bin/bash
function capture () { :; }
function splitter () { :; }
function rename () { :; }
function download () { :; }
function menu () { :; }
```

Listing 2: Basic Variables

```
VERBOSE=true

LYNX="$(which lynx)"
XCLIP="$(which xclip)"

download_directory=~/Downloads7

filetypes=(jpg jpeg png tiff gif bmp swf svg)
filetypes+=(mp4 mp3 mpg mpeg vob m2p ts mov avi wmf asf mkv webm 3gp flv)
filetypes+=(gzip zip tar gz tar.gz 7zip)
filetypes+=(pdf doc xlsx odt ods epub txt mobi azw azw3)
filetypes+=(iso dmg exe deb rpm)
filetypes+=(java kt py sh zsh)
filetypes+=($(echo ${filetypes[@]} | sed -r 's/.+/\U&/'))

free=$(df /home | gawk 'NR == 2{print $4}')
```

Listing 3: Outputting Error Messages

```
function warn () {
  if ${VERBOSE}; then
    echo "WW: ${1}";
  fi;
}

function err () {
  if ${VERBOSE}; then
    echo "EE: ${1}";
  fi;
}

if [ -z ${LYNX} ]; then
  err "Lynx not available on the system."
  err "Cancel."
  exit 1
fi

if [ -z ${XCLIP} ]; then
  err "Xclip not available on the system."
  err "Cancel."
  exit 1
fi

[ ! -e ${download_directory} ] && mkdir -p
  ${download_directory}
```

By making these functions as abstract as possible and keeping them independent of the script, I was able to include a function for renaming files in other scripts without needing to modify them.

In abstract terms, a function takes either no parameters, one parameter, or multiple parameters and eventually returns something, regardless if you use the function in this script or in a completely different context.

Fundamentals

Listing 2 shows a couple of basic things you need to do for the download manager script: declare basic variables that will determine the program flow later on and store file types in an array.

Because you can download a variety of different file types off the web, it is up to you whether or not you add any file types to the array and what structure you choose. However, I recommend keeping some kind of order, for example, by putting the graphic files in one line in your script and video or other file types in another line. The last line records the available storage space in a variable.

Listing 3 contains two functions that your download manager will need to output messages if the `VERBOSE` variable is set to `true`. Also, notice the two `if` statements that check for the presence of `lynx` and `xclip`. If either tool is missing, the script outputs an error message and terminates with `exit 1`. If the download directory does not exist, the script creates it in the last line. If overly verbose warnings or error messages bother you, set `VERBOSE` to `false`.

Functions

It makes sense to follow the same approach for functions as you did for the variables. You should design functions so that they can coexist independently in this and any other scripts.

If it is not immediately apparent from a function how many parameters it takes and what it returns, you need to add a comment describing that. The `capture` function takes care of capturing the links from the web page and storing them. Listing 4 first creates three arrays that the functions fill with values. The first array stores and processes the URLs.

Because I want the script to arrange the downloads by size, `wget` uses the

Listing 4: Functions

```
01 declare -a download_links
02 declare -a indexed_downloads
03 declare -a indexed_indexes
04
05 function capture () {
06   lynx_options="-dump -listonly -nonumbers" # further potential options
07   lynx_command="lynx $lynx_options $url"    # -hiddenlinks=[option], -image_links
08   grep_searchstring="http.+($(sed 's/ /|/g' <<<${filetypes[@]}))$"
09   grep_command="grep -Eoi $grep_searchstring"
10   download_links=(`$lynx_command | $grep_command`)
11   for x in ${download_links[@]}; do
12     file_size=$(wget --spider $x 2>&1 | gawk -F " " '/Length/{print $2}')
13     while true; do
14       [ -z ${indexed_downloads[$file_size]} ] &&
15       indexed_downloads[$file_size]=$x &&
16       break || (( file_size++ ))
17     done
18   done
19   indexed_indexes=(${!indexed_downloads[@]})
20 }
```

`--spider` option (in line 12) to discover the size. Then the `indexed_downloads` array captures the downloadable file, using the file size as the index and the name of the download itself as the value (line 15). This avoids the typical indexes (`0`, `1`, `2`, `3`, and so on) for the array, and instead the file size gives you, say, `233`, `1004`, `780`, and so on, which Bash prints in ascending order of size when listing all indexes. This also happens in line 19, where the `indexed_indexes` array stores the file sizes.

Later, you will see that the potential downloads appear in ascending order of size. Occasionally, two files are exactly the same size. However, the `while` loop in lines 13 to 17 fields this problem. To make this index suitable for the download, the script increases the displayed size by one (virtual) byte in this case.

Short and Painless

Listing 5 shows a function that sensibly splits up the download selection. You make the selection after listing the downloads by typing `1,2,3,4-10`, for example. In this case, the script takes the downloads from `1` to `10`, which are then sent to an array in the form of `(1:*2 3:*4 ...)`.

Listing 5: Splitter Function

```
function splitter () {
  sed 's/,/\n/g' <<<$* | sed -r '/-/ s/([0-9]+)-([0-9]+)/seq \1 \2/e' | sort -nu
}
```

Listing 6: Rename Function

```
01 function rename () {
02   filename=$1
03   other_filenames=`echo ${@:2}`
04   while grep -q -F "${filename}" <<<${other_filenames}; do
05     filename=$(sed -r 's/(.+)(\.)(.+)/\1_\2\3/' <<<${filename})
06   done
07   echo ${filename}
08 }
```

Listing 7: Debug Function

```
$ cat <(sed -r -n '/function (warn|rename)/,/^}/p' downloader_optimized2.bash) > debug
```

Listing 8: Download Function

```
function download () {
  name=$(basename $1)
  suffix=$(cut -f 2 -d "." <<<${name})
  [ ! -e ${download_directory}/${suffix} ] && mkdir ${download_directory}/${suffix}
  cd ${download_directory}/${suffix} && files_in_directory=$(ls)
  future_name=$(rename $name $files_in_directory)
  wget -O $future_name $1
}
```

Listing 9: Menu Function

```
01 function menu () {
02   for index in ${!indexed_downloads[@]}; do
03     local base_name=$(basename ${indexed_downloads[$index]})
04     local size=${index}
05     echo "${size} ${base_name}"
06   done | gawk --assign free=${free} -F " " -f cutter.awk
07 }
```

The `splitter` function strips the string to remove the commas and creates a sorted, space-separated string of numbers from ranges such as `4-10`. You can use these selection numbers later on to find the downloads contained in the `indexed_downloads` array.

Keep in mind that the count for Bash arrays always starts at `0`. For example, to select file number `5`, you need to find it by querying `indexed_indexes[4]`. You can use the same index with the `indexed_downloads` array to retrieve the associated value.

Listing 6 shows the function that renames the files at download time. You need to pass in the base name of the URL as a parameter. At this point, the script is already in the right directory for the file extension (e.g., `jpg/`). The parameters starting in line 3 find other files that are already in the directory before the download starts. Bash uses the command from line 4 to check if there are files with the same name. If so, the script inserts an underscore (_) between the name and the dot that separates it from the file extension. This will also tell you how many times the file has been renamed. If the name contains one underscore, the file was renamed once; if it contains two, the file was renamed twice; and so on.

At this point, you should debug the functions in detail by isolating them. For example, using `sed` and the command from Listing 7, you could write the `warn` and `rename` functions to a separate file, where you would then subsequently debug them with, for example,

```
bash -x debug
```

by renaming the function within the file and calling it with the associated parameters.

Listing 8 shows the download function, which first filters out the base name from the download link. To do this, it deletes all path information, as well as `http://...` or `https://...`, until only the actual file name remains. The script then finds the file extension and, if it does not already exist, creates a directory with this name. Then it changes to the directory and starts the download after running the `rename` function.

Listing 9 generates a menu that lists the available downloads. This function

starts a loop that iterates across the `indexed_downloads` array and outputs the size and the base name from the array index one line at a time. At the end of the loop in line 6, everything is piped to `gawk`.

Thanks to the `-f cutter.awk` option, `gawk` knows which AWK file to use as the program text. The call has an additional `--assign free=${free}` option, which ensures that the `gawk` script is aware of the free disk space previously determined in Bash. `gawk` then examines the file size and the base name one line at a time and evaluates both line by line.

Formatted Displays

The `gawk` script, `cutter.awk`, in Listing 10 starts with two functions I defined myself. The first function, `cutter`, truncates long basic names for the display by cutting them into two parts and dropping three dots into the middle. The second function, `separating_line`, generates separating lines in the display to improve clarity for download pages with a large number of links.

The `BEGIN` block defines some basics as well as header formats. In addition, it shows the free space on the hard disk, or in your home directory, in the line following the header.

Finally, the command block without a pattern specification in lines 21 to 27 iterates through the individual lines, displaying the download size (in kilobytes, megabytes, and gigabytes) in the first field and the basic name in the second. This can be useful if you are downloading smaller files, such as wallpapers, ebooks, or MP3 files – output in gigabytes only, for example, would not make much sense here. This command block also computes the total size of all downloads and finally outputs it via the `END` block (Figure 1).

Control

Finally, Listing 11 shows the main function that controls the entire program flow. As soon as you copy a URL from the browser by pressing Ctrl + C, `xclip` accesses it (line 1).

If the graphical interface uses multiple clipboards (like FVWM), you need to use the `-selection primary` or `secondary` option to explicitly specify which clipboard `xclip` should use.

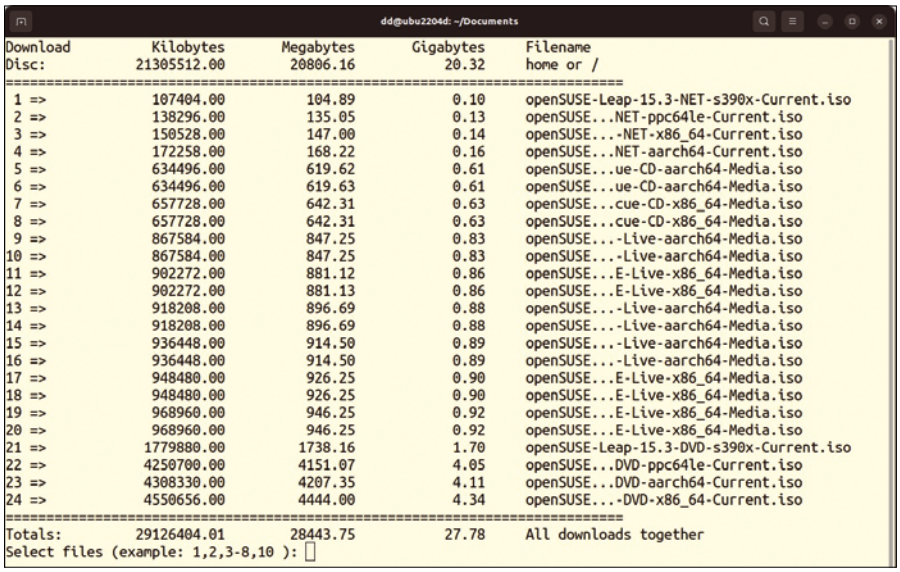


Figure 1: The formatted output shows the files provided by a website. To select a file, enter a number from the left.

Listing 10: cutter.gawk

```
function cutter( word ){
  l = length(word)
  part1 = substr(word,1,8)
  part2 = substr(word,l-22)
  return part1"..."part2
}

function separating_line ( lesser_equal ) {
  for ( p = 0; p <= lesser_equal ; p++){
    printf "%s" (p == lesser_equal ? "\n" : "") ,"="
  }
}

BEGIN {
  i = 1
  printf "%8s %18s      %10s   %13s     %s\n", "Download", "Kilobytes", "Megabytes",
                                              "Gigabytes", "Filename"
  printf "%-5s %21.2f      %10.2f   %13.2f     %s\n", "Disc:", free, free/1024,
                                                      free/(1024*1024),"home or /"
  separating_line(75)
}

{
  if ( length($2) > 40 ) {
    $2 = cutter($2)
  }
  printf "%2i => %21.2f   %13.2f   %13.2f     %s\n", i++, $1/1024, $1/(1024*1024),
                                                     $1/(1024*1024*1024), $2
  total += $1
}

END {
  separating_line(75)
  printf "Totals: %19.2f   %13.2f   %13.2f     All downloads together\n",
                                                 total/1024, total/(1024*1024),
                                                 total/(1024*1024*1024)
}
```

After capturing the clipboard, the script checks whether the URL was also filled with the clipboard contents. If the URL has a length of zero, the program cancels the operation.

Lines 11 to 16 then check whether any download links have been captured on the page and whether they can be output. If there are no files to download, the script also terminates at this point.

Finally, a `for` loop processes all the downloads, checking their sizes and comparing them against the free disk space. If the size of the selected downloads exceeds the free disk space size, the script terminates.

Because all you need to do is copy the URL of the desired site for downloading files from the browser's address bar for the script to start automatically, this script works with all web browsers. The script retrieves the URL from the clipboard, evaluates the page for files that can be downloaded, and shows them to you sorted by size. You can then conveniently choose which of them you want to download.

Listing 11: Main Function

```
01 url=$(xclip -o 2>/dev/null)
02
03 if [ -z $url ]; then
04   warn "No URL present."
05   warn "Use [Ctrl]+[C] to copy the URL from the browser to the clipboard."
06   exit 1
07 fi
08
09 capture
10
11 if [ ${#download_links[@]} -gt 0 ]; then
12   menu
13   read -p "Select files (Example: 1,2,3-8,10 ): " selection
14 else
15   warn "NO DOWNLOADS PRESENT" && exit 1
16 fi
17
18 declare -i total_size_dowloads=0
19
20 for selection in $(splitter $selection); do
21   total_size_dowloads+=${indexed_indexes[((select - 1))]}
22   current_download=${indexed_downloads[${indexed_indexes[((select - 1))]}]}
23   if [[ ${frei}-5000 -lt ${total_size_dowloads}/1024 ]]; then
24     warn "Not enough free disc space."
25     warn "Canceling ${Current_download}."
26     exit 1
27   else
28     download $current_download
29   fi
30 done
```

Conclusions

Generally speaking, this custom download manager offers a simple and reliable approach for conveniently downloading files off the web. You just need to copy the download page's URL to the clipboard and then select the desired files in the shell.

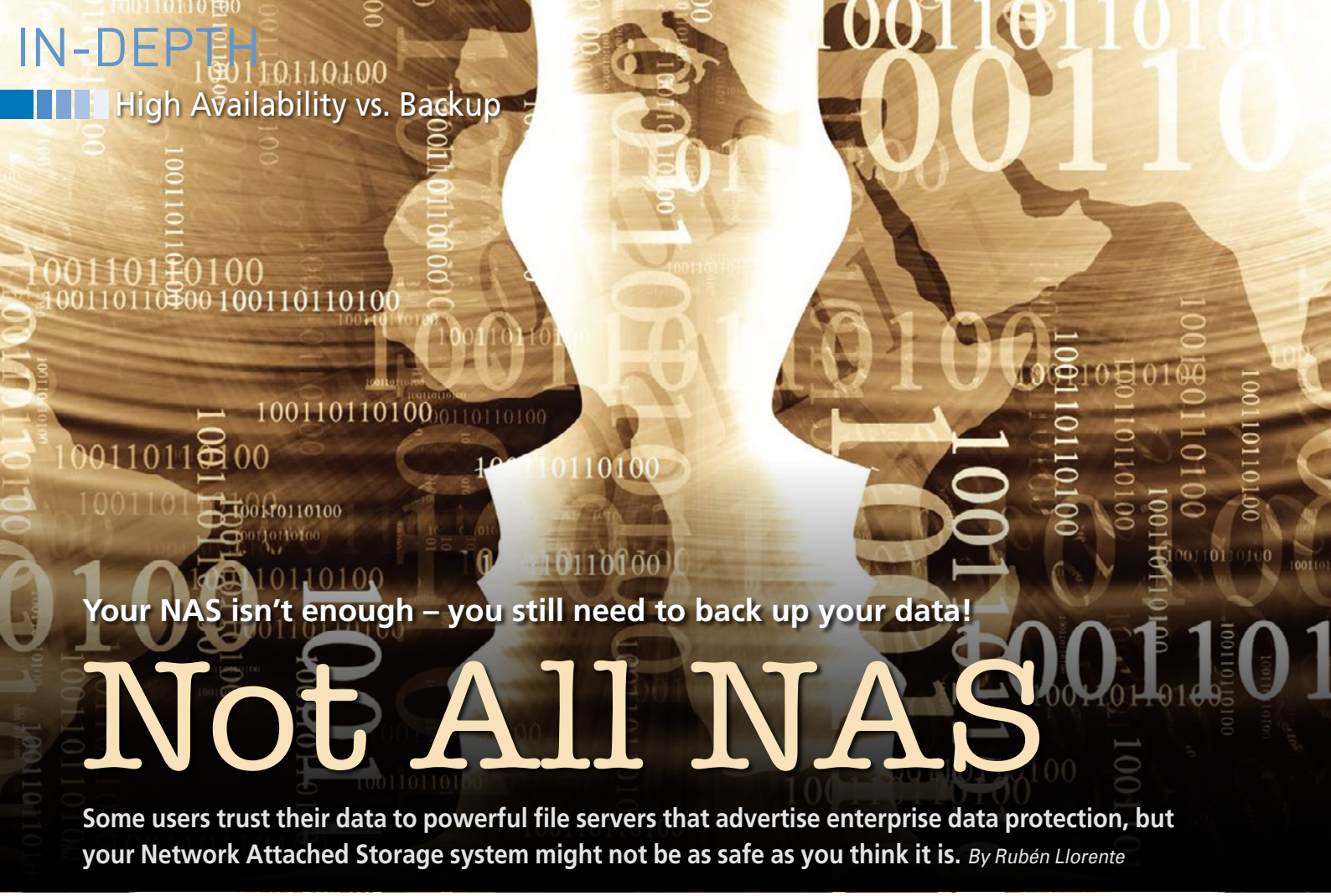
Because I designed the scripts to use a very structured approach, and the functions included here mostly work on their own, they can also be used in other shell programs. The scripts for this article (`download_optimized2.bash`) along with `cutter.awk` are available for download at [1]. ■■■

Info

[1] Download manager scripts: *ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/260/*

Author

Goran Mladenovic is a hobby developer and inventor, who believes programming is a passion.



Your NAS isn't enough – you still need to back up your data!

Not All NAS

Some users trust their data to powerful file servers that advertise enterprise data protection, but your Network Attached Storage system might not be as safe as you think it is. *By Rubén Llorente*

There is a point in the life of a compulsive data hoarder when a regular computer is not enough to contain a burgeoning file collection. Upon the relentless expansion of a massive data compilation, the first step a home user takes to extend the storage capacity is to purchase an external USB hard drive. The hard drive will buy the user some time, but eventually this solution will fall short. A data hoarder who is dedicated enough will eventually have to invest in a Network Attached Storage (NAS) server.

A NAS is a dedicated server optimized to store large amounts of information. NAS servers are commonly available as commercial appliances, but many power users prefer to build their own from spare parts. Serious NAS servers are scalable and allowed to increase their capacity by adding hard drives as needed. Better yet, they often offer enterprise features that come in very handy, and they promise mitigations to the most common threats against the long term survival of your files.

Figure 1: The TrueNas Mini E is a popular NAS appliance. It features 8GB of ECC RAM (which is upgradeable to 16GB) and four hot-swap bays for hard drives.

NAS vendors often advertise fault tolerance and profess the immunity of their systems from disaster, which causes users to treat this sort of storage as bulletproof, dumping their data and then skipping the step of making backups. But rarely do these consumer-grade storage systems provide a complete solution. This article describes some of the things that can go wrong – and why you still need to perform backups to ensure that your data is safe.

The Features of a Quality NAS

A wide range of NAS options are available for home users. These options vary in quality from desktop toys to quasi-enterprise systems trying to pass as domestic appliances (Figure 1).

With the exception of the low end ones, NAS boxes are designed with the purpose of offering the highest possible availability. In this context, a *high availability* machine is one that can keep serving its users under adverse conditions. Such a server needs to be able to keep functioning if a hard drive fails, if the power grid blacks out, or if its power supply malfunctions.

Servers mitigate hard drive failures by the use of Redundant Array of Independent Disks (RAID). A RAID group is just a set of hard drives that are recognized as a single virtual drive by the

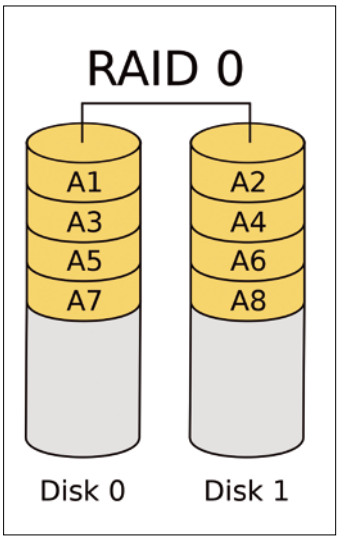


Figure 2: RAID 0 distributes the data across the drives of the array. This configuration is good for performance, but losing a single drive destroys the whole array.

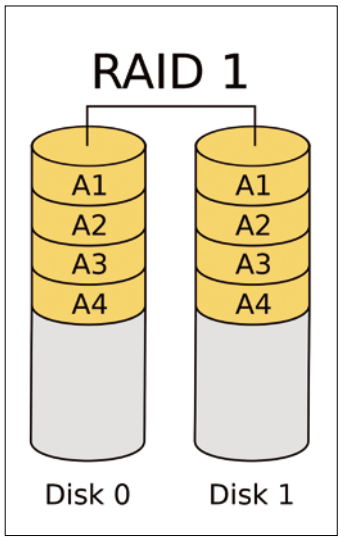


Figure 3: RAID 1 ensures that the data is mirrored from one drive to the other. As long as there is a functioning drive, the array will keep working, but this configuration is not cost effective.

Popular RAID Levels

RAIDs can be built in multiple ways, depending on the purpose they serve. The most popular traditional RAID levels are:

- *RAID 0* stripes data across all the drives in the set for increased performance (Figure 2). The total size of the RAID is that of the sum of the sizes of every individual drive. A disk failure kills the array, making it a dangerous RAID level to use. RAID 0 has better read and write throughput than a single hard drive of the same size as the array, because the workload is evenly distributed over the individual drives in the RAID.
- *RAID 1* mirrors the data across all the drives in the array (Figure 3). Since every drive has a full copy of all the data, a RAID 1 can keep working as long as one of its drives is still operational. RAID 1 is good for keeping a proper uptime, but it is not very cost effective, because, at the very least, it takes twice as many drives for the same storage capacity.
- *RAID 5* is among the most popular in small deployments. This form of RAID is known as *disk striping with parity*. The disks are striped (as with RAID 0), but an additional drive provides a parity bit, ensuring that the array can keep working if one of the drives fails (Figure 4). *RAID 6* does pretty much the same thing, except it can keep working after two hard drive failures.
- *RAID 10* is a combination of RAID 0 and RAID 1. Drives are deployed in couples in which each unit mirrors the other. Then all the pairs are placed in a RAID 0 (Figure 5). RAID 10 can keep functioning as long as at least one drive in each pair is in working order.

operating system. (See the box entitled "Popular RAID Levels" for more information on some common RAID scenarios.) In a domestic NAS context, these drives will most often be grouped in the so called RAID 5 level. RAID 5 distributes the data within the array evenly across every device, with some extra parity components. Should one of the drives fail, the server will keep functioning in a degraded state by keeping the remaining drives running and using the parity data to reconstruct lost information.

A server can survive blackouts by the use of an Uninterrupted Power Supply (UPS), which is just a fancy term for a battery that kicks in when the power grid goes down (Figure 6). A modern UPS can communicate with the server over USB or Ethernet in order to let the operating system know how much power is left in the battery, which is useful to force the machine to shutdown in an orderly way when the supply is about to run dry.

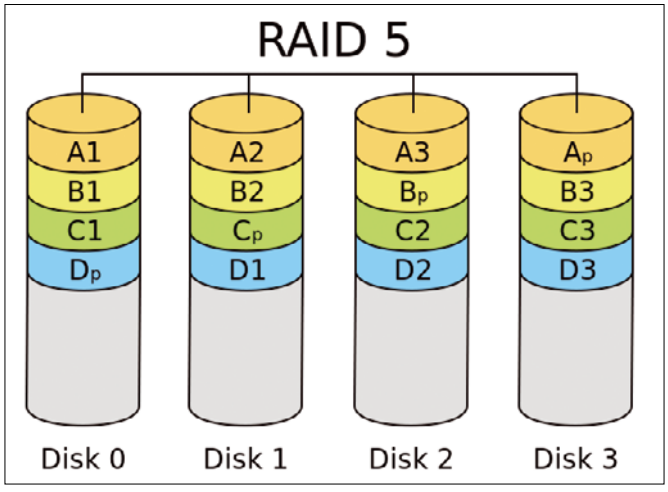


Figure 4: In a RAID 5 configuration, data is distributed evenly across all the drives of the array, alongside a small amount of parity information, in such a way that the server hosting the array may keep functioning if one of the drives fails.

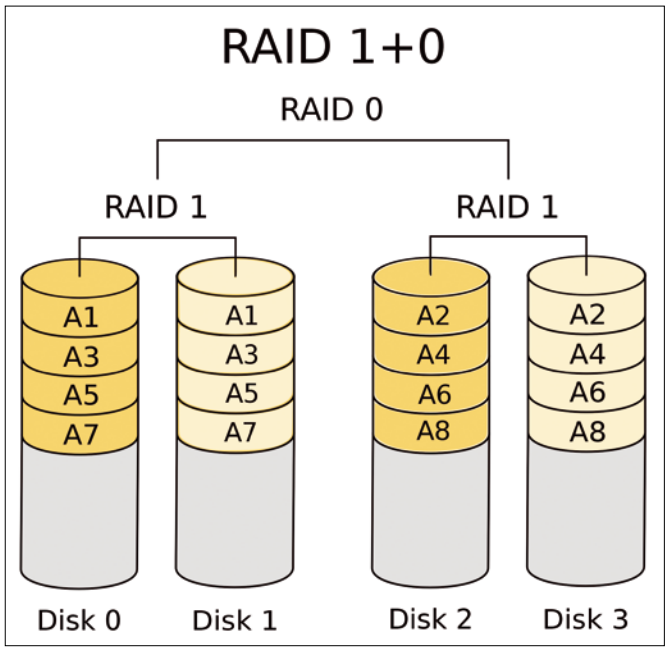


Figure 5: RAID 10 places RAID 1 pairs within a RAID 0. This configuration is very fault tolerant but also very expensive.

About ECC

Good NAS hardware will often feature Error Correction Code (ECC) RAM. ECC RAM is capable of checking itself for consistency against random errors in memory, which are more frequent than it seems [1]. RAM errors are considered dangerous for the survival of a dataset and the continued operation of a server. A botched bit in RAM could cause the operating system to malfunction or cause a file to get corrupted. ECC is intended to reduce the risk of such an event and keep the system running after a memory error.

A theory holds that a bit error in RAM could cause a chain reaction, resulting in massive data corruption within a ZFS filesystem. It is therefore argued that the only safe way of running a ZFS server is with ECC RAM, and that doing otherwise is borderline suicidal.

ZFS uses no pre-mount consistency checker and lacks filesystem repair tools at the time of this writing. ZFS was conceived as a self-healing filesystem, capable of repairing data corruption on the go. Should ZFS try to read a data block that has been corrupted by, let's say, a hard drive defect, the filesystem would be able to identify the issue and attempt to repair it on the fly from parity data. Such self-healing features do, in theory, eliminate the need for recovery tools. The FreeNAS project (now TrueNAS) used to warn that a botched memory operation could cause permanent damage to the filesystem, and since there are no recovery tools available, data could end up being unrecoverable [2].

Figure 6: File servers are often paired with an Uninterrupted Power Supply system, such as this CyberPower unit. This device will prevent an unclean shutdown in case of blackout.

However, opinions differ on whether ZFS is more susceptible to failure than other filesystems. Matthew Ahrens, cofounder of Sun's ZFS project, argues that using ZFS with non-ECC RAM is about as risky as running a regular filesystem without it [3], arguing that ECC RAM is not necessary but is highly recommended.

RAID Issues

A good NAS promises excellent uptime and looks indestructible on the surface. It would seem like files should be able to survive indefinitely in such a server. After all, if a NAS is capable of withstanding a hard drive failure (the most common hardware malfunction [4]), there is not much incentive for spending the big amount of money required to set another server up and keeping a backup of the original one.

The problem is that there is only so much a file server can do to protect your data, especially outside of an enterprise environment. Quality server hardware is designed to guarantee good uptime in the face of trouble, but not necessarily the integrity of your information. There are a number of reasons why a NAS may still fail.

If a hard drive fails within a NAS' RAID 5 set, the whole array will work at a degraded level. From the user viewpoint, the array is still operational, but it has ceased to offer fault tolerance. Should another drive fail before a new one is added and the array is rebuilt, the information contained in the array will be lost. Many a RAID array has failed due to owner procrastination – or due to the long wait time waiting for the attention of an overworked sys admin.

But tardy repair is just one of the reasons why some experts are wary of depending on RAID. A casual search on the Internet will find countless opinions regarding the unsuitability of RAID 5 for modern file servers [5]. Storage media is not perfect and may suffer random read failures. Hard Drives are reliable enough for most purposes [6], but every now and then they will throw an Unrecoverable Read Error (URE). UREs are errors which take place when the hard drive tries to access a block of data and fails to do so. Modern drives are estimated to suffer an URE for every 10^14 bits read on average, which means errors are rare.

The bigger a disk array, the higher the chance that a defective sector exists somewhere. The argument of RAID 5 detractors is that disk arrays are becoming so big that the probability of triggering a URE is becoming too high to be acceptable. This is so because the more bits are managed by the RAID, the more likely it is that at least one block of information is problematic.

If a RAID 5 loses a drive to hardware failure, a new drive can be plugged in, and the RAID 5 may be rebuilt from the data existing in the remaining disks. However, if any of the remaining disks throws a URE during this process, the consequences may range from losing the data existing in that sector to being unable to rebuild the whole RAID (depending on the quality of the RAID controller and drives).

Experience suggests that the fear of being unable to rebuild big arrays is blown out of proportion. Nevertheless, it is important to remember that RAID 5 is a tool for guaranteeing uptime rather than the integrity of your files.

There are RAID levels with better fault tolerance than RAID 5 (such as RAID 6 or RAID 10) but using these alternative RAID levels in a small system is comparatively expensive.

Nearly as bad as this is the fact that many RAID controllers are proprietary and don't offer a good migration path. If you are using a proprietary solution and want to move your hard drives from an old server – maybe because the old one finally bit the dust! – you might discover that your data is unreadable in its destination machine.

On the other hand, software issues might destroy your files just as quickly as a hardware level malfunction, and

using an enterprise-grade server won't do much for you if you are hit by a bug. For example, QNAP's NAS appliances were massively affected by a vulnerability that caused many users to be preyed on by the DeadBolt ransomware [7][8].

Power Failure

Modern filesystems are moderately resistant to power failure, but even the mighty ZFS could suffer from a blackout [9]. A UPS will help, but beware of cheap units: Many budget domestic UPS are not prepared to handle continuous operation and will wear out, eventually bringing down the NAS with them. According to a Ponemon Institute's 2016 survey, UPS failure is the top cause of unplanned data center outages [10]. What this means in practice is that blackout protection reduces the risk of suffering data loss from power loss, but it does not remove the threat entirely.

In enterprise scenarios, administrators are aware that trying to make a NAS bulletproof is not enough to guarantee true high availability. In practice, the enterprise uses Storage Area Networks (SAN) or distributed filesystems such as Ceph [11]. Such tools are deployed in computer clusters, in such a way that if a server goes down, the rest of the cluster remains operational.

The minimal (and, for serious purposes, insufficient) storage cluster that can be deployed is described in Figure 7. This is known as a Primary-Replica topology, in which the primary performs services for the clients. The replica's contents are periodically synchronized with the primary's. Should the primary go down, the load balancer will promote the replica and turn it into the new primary (Figure 8).

The Cloud Option

Real life high-availability systems are not something you are likely to be able to run at home: typically they feature redundant load balancers and might require some Border Gateway Protocol (BGP) magic thrown in. Even the naive and simple method I just described multiplies the cost of the storage by more than two, because it requires a redundant server and a load balancer (at which point you are likely to need a server rack in a server room).

It is therefore not a surprise that many users, especially small businesses, turn to professional storage vendors, who offer cloud storage for a fee and take care of ensuring the storage systems are perpetually available. Professional storage vendors might also be very cost effective. For example, cloud storage might cost you around $1,500 in four years, which is less than what you are likely to spend on a good NAS. As I assume a NAS is likely to need an upgrade around the fourth year, the cloud option is not entirely unreasonable. Sadly, storage vendors come with their own issues: Uploading your data to them can take much longer than uploading it to a local server, and some vendor environments might present privacy concerns.

Humans and Software

Even if you were to assume that your chosen storage solution is completely indestructible, it would still not

eliminate the need for a proper backup system. If you manually delete a file by mistake, or if you lost the file to a software bug or malware, it makes no difference whether it was stored on a regular laptop, a high-end NAS, or a cloud storage provider. Experience shows that human mistakes force you to restore from backups much more often than hardware failures. Certain storage vendors know this and keep a historical registry of every file uploaded to them, so you can retrieve an old version of a file if you discover you have uploaded a corrupt version or deleted something important by accident. Interestingly, the vendor is actually running a backup policy for you.

Conclusion

A high-availability system is designed to serve its users even if issues such as hardware failure or power loss affect it. A side effect from a high-availability setup is that information that would have been lost from a failure in a non-redundant system may survive if it is managed by a storage cluster or even a high-end domestic NAS.

On the other hand, high-end storage systems can only protect your data so much. As shown in this article, solutions designed to keep a storage system running in the face of adversity might fail to guarantee the integrity of the data. After all, their primary concern is to maintain the continuity of the service, not to protect the information stored inside.

For this reason, it is advisable to maintain proper backup for your data, even if you keep it in a NAS server that looks impervious to the typical threats against data integrity. Quality storage decreases the probability of suffering data loss, but does not remove it. ■■■

Info

[1] Schroeder, B., E. Pinheiro, and W. Weber. "DRAM Errors in the Wild: A Large-Scale Field Study." *In*: *Proceedings of SIGMETRICS '09*, (SIGMETRICS, 2009), *www.cs.toronto.edu/~bianca/papers/sigmetrics09.pdf*

[2] "A Complete Guide to FreeNAS Hardware Design, Part I: Purpose and Best Practices" by Joshua Paetzel, February 3, 2015, *https://web.archive.org/web/20151122065016/http://www.freenas.org/whats-new/2015/02/a-complete- guide-to-freenas-hardware-design- part-i-purpose-and-best-practices.html*

[3] ZFS and ECC RAM: *https://arstechnica.com/civis/viewtopic.php?f=2&t=1235679&p=26303271#p26303271*

[4] Common hardware malfunctions: *https://blog.storagecraft.com/hardware-failure*

[5] Unrecoverable errors in RAID 5: *http://raidtips.com/raid5-ure.aspx*

[6] Backblaze drive stats for Q1 2021: *https://www.backblaze.com/blog/backblaze-hard-drive-stats-q1-2021/*

[7] QTS and QuTS hero vulnerability: *https://www.qnap.com/en/security-advisory/qsa-21-57*

[8] DeadBolt: *https://www.qnap.com/en/security-advisory/QSA-22-02*

[9] ZFS and power failures: *https://www.klennet.com/notes/2021-04-26-zfs-and-power-failures.aspx*

[10] Cost of data center outages: *https://www.ponemon.org/research/ponemon-library/security/2016-cost-of-data-center-outages.html*

[11] Ceph: *https://ceph.io/en/*

Author

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

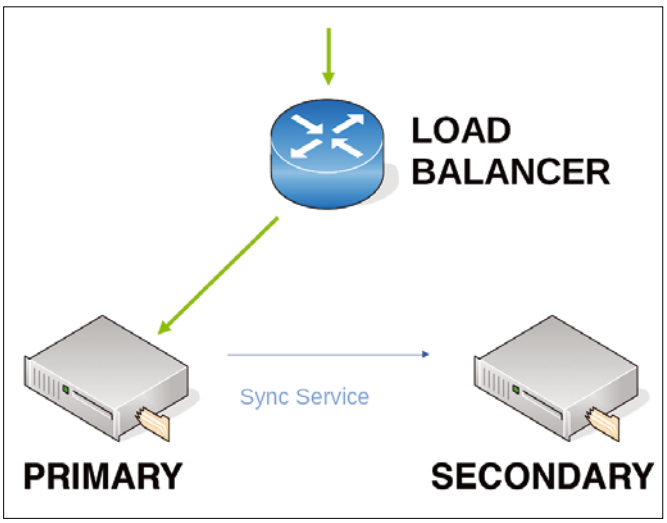


Figure 7: A naive high-availability cluster. A load balancer directs all traffic to a file server designated as the primary. The file server designated as a replica contains a copy of the primary's contents.

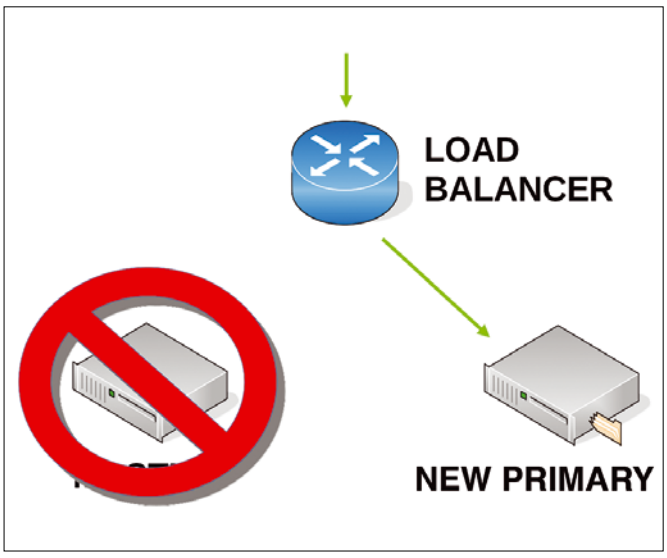


Figure 8: If the primary server goes offline, the replica is promoted to primary and the traffic is transferred to it.

Photo by Sereja Ris on Unsplash

A modern terminal emulator

Here, Kitty, Kitty

Kitty, a terminal emulator by the creator of Calibre, promises customization and graphical acceleration at the command line. *By Bruce Byfield*

Kovid Goyal is best known as the creator of Calibre, an ebook management application so far reaching that, despite a poor interface, it has become the definitive tool in its class. However, for the past five years, Goyal has also been developing kitty [1], a terminal emulator for the modern age, with features that range from text formatting and tiling to graphical acceleration. While kitty's list of features is intriguing, whether kitty can repeat the success of Calibre remains to be seen (Figure 1).

Versions of kitty are available in many Linux distributions, as well as for macOS and some BSDs. Installing from your distribution's package has the advantage of automatically integrating kitty with your environment. However, as with Calibre, new versions are released every three to six weeks, so users who want the most advanced version should download precompiled binaries (Figure 2) to their regular account with:

```
curl -L https://sw.kovidgoyal.net/↵
  kitty/installer.sh | sh /dev/stdin
```

If you'd prefer an overnight build, use:

```
curl -L https://sw.kovidgoyal.net/↵
  kitty/installer.sh | ↵
  sh /dev/stdin \ installer=nightly
```

To install the overnight build without overwriting an existing installation,

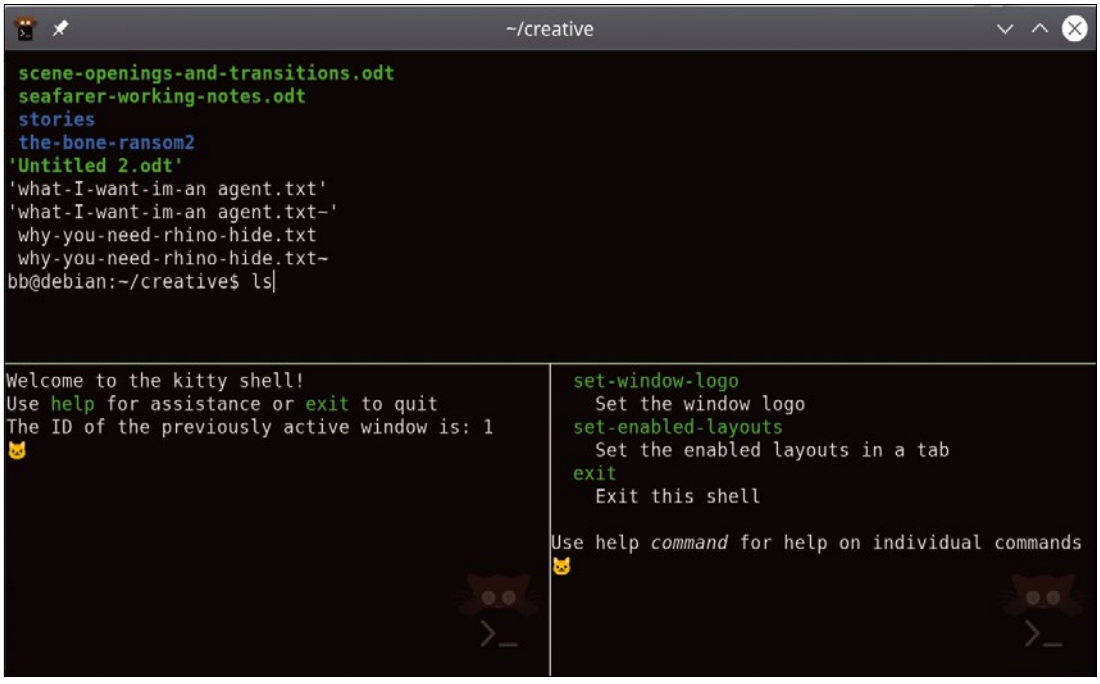


Figure 1: Kitty is a modern rethinking of the terminal emulator.

```
bb@debian:~$ curl -L https://sw.kovidgoyal.net/kitty/installer.sh | sh /dev/stdin
  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100  4299  100  4299    0     0  46728      0 --:--:-- --:--:-- --:--:-- 46728
Installing to /home/bb/.local/kitty.app
Downloading from: https://github.com/kovidgoyal/kitty/releases/download/v0.25.0/kitty-0.25.0-x86_64.txz

  % Total    % Received % Xferd  Average Speed   Time    Time     Time  Current
                                 Dload  Upload   Total   Spent    Left  Speed
100   660  100   660    0     0   1605      0 --:--:-- --:--:-- --:--:--  1605
100 13.6M  100 13.6M    0     0  3415k      0  0:00:04  0:00:04 --:--:-- 4321k
bb@debian:~$ kitty
bash: kitty: command not found
```

Figure 2: Kitty is in rapid development, so install kitty's available binaries rather than the package in your distribution's repository.

Listing 1: Integrating kitty

```
ln -s ~/.local/kitty.app/bin/kitty   ~/.local/bin/

cp ~/.local/kitty.app/share/applications/kitty.desktop ~/.local/share/
applications/

cp ~/.local/kitty.app/share/applications/kitty-open.desktop ~/.local/share/
applications/

sed -i "s|Icon=kitty|Icon=/home/$USER/.local/kitty.app/share/icons/
hicolor/256x256/apps/kitty.png|g" ~/.local/share/applications/kitty*.desktop

sed -i "s|Exec=kitty|Exec=/home/$USER/.local/kitty.app/bin/kitty|g" ~/.local/
share/applications/kitty*.desktop
```

add `dest=DIRECTORY` at the end of the command.

In both cases, the files to run kitty are installed to `~/.local/kitty.app`. For convenience, you can incorporate kitty into your path, desktop, and file manager by running the series of commands shown in Listing 1.

Alternatively, you can simply download the latest version each time you want to experiment with kitty. A new download opens kitty. Keep in mind that at kitty's current stage of development, some users may not want to use kitty for regular work.

Configuring kitty

Kitty has all the features you would expect in a modern terminal emulator, including multiple command prompts, tabs, background opacity, and fonts. However, many of these features are far more elaborate than in standard emulators. For example, fonts can be set separately for individual characters, by pixel or points, line height, or column width, and can be mapped to any Unicode symbol. Font ligatures are enabled by default, but they can be disabled, and letters can be mapped to use special features, such as zero with a slash through it to distinguish it from a capital *O*. In short, most of the features available via HarfBuzz's font-rendering engine are available for Latin characters in kitty. Other features have their own numerous details. For instance, cursors can be set for color, size, and shape, while the size and display of the scrollback history can be adjusted for each prompt. You can set many of these features in `kitty.conf`, the configuration file, by pressing Ctrl + Shift + Escape (Figure 3).

Others features can be set with keyboard shortcuts, such as Ctrl + Shift + Equal to increase font size and Ctrl + Shift + Minus to decrease font size. A long list of keyboard shortcuts is also available to navigate between prompts, tabs, and windows.

One especially important configuration option lets you choose from seven different layouts for multiple windows:

- Fat: One or more full-width windows on top of the screen, the rest side by side at the bottom
- Grid: All windows tiled
- Horizontal: All windows side by side
- Splits: Windows in arbitrary patterns created by horizontal and vertical splits
- Stacked: One single maximized window displays at a time
- Tall: One or more windows are shown full height on the left, the rest one below the other on the right
- Vertical: All windows displayed horizontally

```
# vim:fileencoding=utf-8:foldmethod=marker

#: Fonts {{{

#: kitty has very powerful font management. You can configure
#: individual font faces and even specify special fonts for particular
#: characters.

# font_family      monospace
# bold_font        auto
# italic_font      auto
# bold_italic_font auto

#: You can specify different fonts for the bold/italic/bold-italic
#: variants. To get a full list of supported fonts use the `kitty
#: +list-fonts` command. By default they are derived automatically, by
#: the OSes font system. When bold_font or bold_italic_font is set to
#: auto on macOS, the priority of bold fonts is semi-bold, bold,
#: heavy. Setting them manually is useful for font families that have
#: many weight variants like Book, Medium, Thick, etc. For example::
```

Figure 3: `kitty.conf` consists of dozens of heavily commented fields for customization.

Figure 4: Each terminal can be named for ease of navigation.

Users can cycle through layouts with Ctrl + Shift + L. In `kitty.conf`, the default layout can be set and others disabled.

A Grab Bag of Features

Few aspects of terminal emulators are left untouched by kitty. To list and explain them all would take a small book. Judging by the online help, it would also be pointless because right now kitty's emphasis is on development rather than documentation, and a given statement could soon become obsolete. However, here are a few of kitty's most convenient features:

- Set tab title (Ctrl + Shift + Alt + T) – useful when numerous command lines are open (Figure 4)
- Open URL in web browser (Ctrl + Shift + E) – must be full URL (Figure 5)
- Control from scripts or remotely [2]
- Launch programs in a specific window or tab set in `kitty.conf` [3]
- Text drag and drop (Figure 5)
- Opening the scrollback history in `less` for easier analysis
- Mark text on screen using regular expressions

In addition, kitty is built modularly, which makes creating extensions easy. Kitty calls extensions kittens [4]. Existing kittens include:

- `icat` – displays images in the terminal
- `diff` – a version of `diff` optimized for kitty
- `broadcast` – typing something in one terminal displays it in other terminals
- `ssh` – a version of SSH optimized for kitty
- `panel` – a dock panel showing output from a kitty terminal

Already, kitty is encouraging extensive rethinking of the venerable terminal emulator.

Display and Performance Tweaks

By far its most important feature, kitty works efficiently with the system's GPU. Kitty caches all the font glyphs used in a session in video RAM, so latency (the time between typing a character and it appearing on the screen) is reduced and scrolling on the screen becomes faster. In `kitty.conf`, the `repaint_delay`, `input_delay`, and, sometimes, `sync_to_monitor` fields can be adjusted to find the ideal balance [5].

Table 1: Terminal Emulators and CPU Usage

Emulator	CPU Usage
kitty	6-8%
xterm	5-7% (but scrolling was extremely janky)
termite	10-13%
urxvt	12-14%
gnome-terminal	15-17%
konsole	29-31%

Source: Kovid Goyal [5]

Hearing of this effort, many might dismiss these performance tweaks as minor. However, Goyal has posted a comparison table of computer terminals and their CPU usage online [5] (see Table 1). The CPU usage is surprising and obviously depends on circumstances, but using `top`, I have discovered that Plasma's Konsole is consistently in the top five apps for CPU usage, even when not in use. In fact, Konsole rivals Firefox and LibreOffice in the resources it consumes. Assuming that Goyal's other figures are also more or less correct, it seems absurd that terminal emulators should consume so many resources in an era when desktop environments are far more popular. Whatever else kitty does, reducing the consumption of resources seems a much overdue goal.

A Judgement Delayed

Kitty is a work in progress. As I write, the latest version is 24.4, which at Goyal's usual release rate could mean that the general release is several years away. Therefore, kitty's complete list of features cannot yet be known. However, I have to wonder how useful some of the implemented features are to command-line users. For example, can text formatting or emoticons have any useful purpose at the prompt? Perhaps in email? Or can a new purpose evolve? And if a new purpose does evolve, will users care about it? Users have long ago learned to use ASCII art for emoticons, asterisks to indicate italics, and underscores to indicate a bold weight. Will many see a need to change? Just because a feature is practical does not mean that is needed.

Kitty does have some features that change how commands are entered or code is written, such as marking text and multiple buffers. So far, the most useful feature may be its graphical acceleration. It seems absurd, for example, that Konsole should be consistently in the top five resource-hogging applications, second only to Plasma's window manager. For graphical acceleration alone, kitty may end up being worthwhile. For now, all that can definitely be said is that kitty is a project worth watching. In the long run, it might just be the future of terminal emulation. ■■■

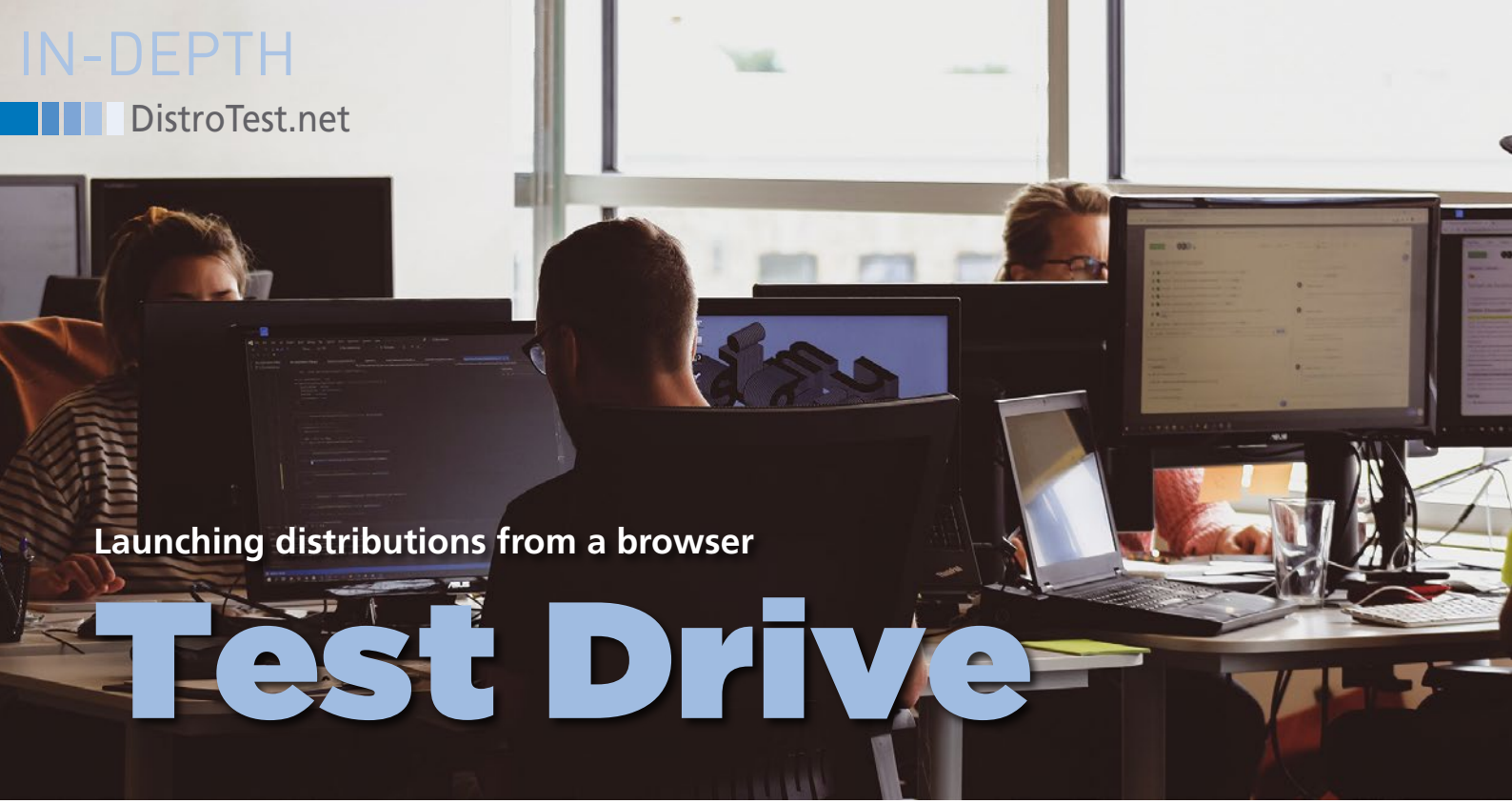
Figure 5: Full URLs in kitty can be opened in a browser, which can be especially useful in comments.

Info

[1] kitty: *https://sw.kovidgoyal.net/kitty/*
[2] Working remotely: *https://sw.kovidgoyal.net/kitty/remote-control/*
[3] Launching: *https://sw.kovidgoyal.net/kitty/launch/*
[4] Extensions: *https://sw.kovidgoyal.net/kitty/kittens_intro/#kittens*
[5] Performance: *https://sw.kovidgoyal.net/kitty/performance/*

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/*.



Launching distributions from a browser

Test Drive

With so many Linux distros to choose from, you can spend a lot of time downloading and installing before you find the right one for your needs. DistroTest.net lets you test a variety of Linux distributions from the browser without the up-front installation work. *By Erik Bärwaldt*

If you've been thinking about switching from another operating system to Linux, you initially will be confronted with a large choice of distributions, desktop environments, and package managers. More than half a dozen popular desktops and several hundred distributions compete for your favor.

Until now it has been difficult, especially for newcomers, to find the Linux derivative best suited to their needs, due to all these choices. That dilemma is now done and dusted. The DistroTest.net website [1] lets you to try out more than 370 Linux distributions online, without having to install them or launch a live system from a removable disc.

Concept

DistroTest.net currently offers some 790 versions of nearly 370 distributions for testing. It provides the Linux derivatives as virtual machines (VMs), which can be used very much like locally installed computer systems with the help of a VNC client. You can upload files to the VMs, and you can install your own software, which is especially useful for testing purposes.

Of course, these virtual Linux systems do not have network access – primarily to prevent possible misuse. You also cannot set up software that involves additional dependencies on the VM. The entire site infrastructure uses free software.

Getting Started

After opening the web page in the browser, you are taken to the splash page, which lists all available distributions in alphabetical order (Figure 1). To view more details about a distribution, click on its name in the list. The web page now displays a screenshot of the desktop for most distributions, as well as a table showing some technical data. For example, you can find out which hardware architecture the distribution supports, how much RAM is available in the VM, and whether it supports the EFI boot mechanism.

In addition to this, you will usually find information about the system's release date, which allows you to draw conclusions about support for new hardware components. If you need authentication data at start-up time, you will also find it here. In addition, a note appears

Photo by Sigmund on Unsplash

Figure 1: From a selection of over 370 distributions, simply launch your choice in a web browser.

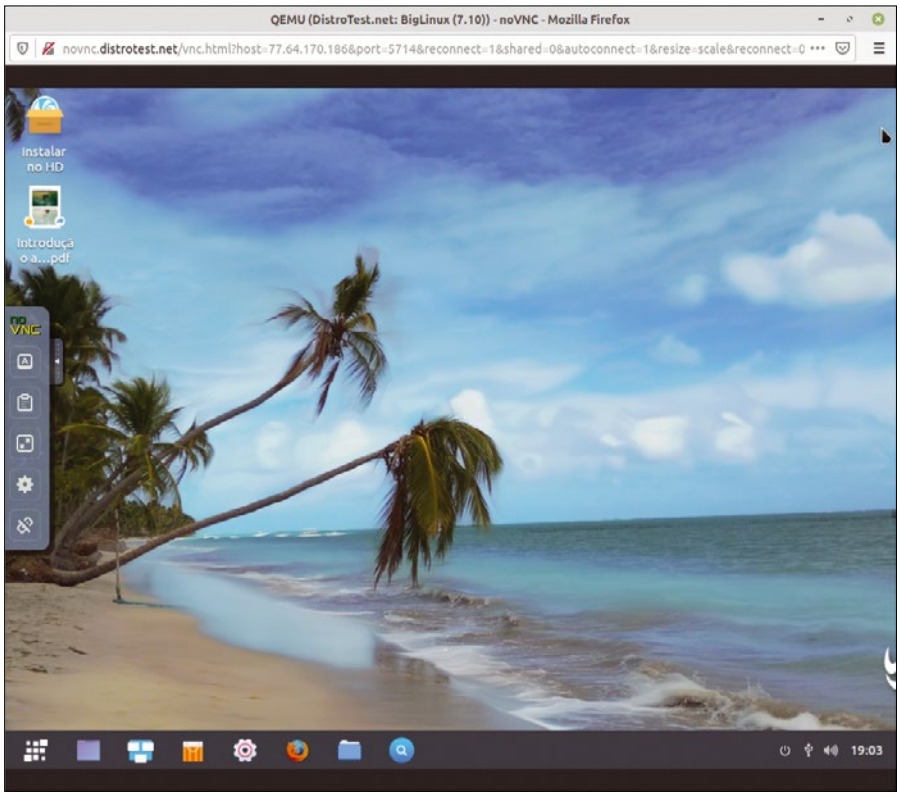


Figure 2: If desired, you can use the noVNC client via a dock that folds out on the left side of the window.

telling you that uploads are limited to a maximum total volume of 10MB.

Below the screenshot, you'll find the *System start* button. Pressing it takes you to another dialog that tells you how many slots will launch before you. If there are several, it will take a few minutes for your system to boot. Please note that the page starts the VMs in a pop-up window, so you need to allow the page to open pop-ups. When your slot is ready, the selected Linux distribution launches in the default VNC client. Each slot remains active for 30 minutes and can be extended by a quarter of an hour by clicking *Extend time (+ 15 minutes)* if required.

If the distribution window does not open, click the *Open VNC Viewer* button below the slot display. The software then opens the free noVNC HTML VNC client in a separate window. You are then taken to the GRUB boot menu for most distributions where you can boot the system with various parameters as you would with a local installation. After that, the system's desktop appears in the client (Figure 2), usually with XGA resolution (1024x768 pixels).

On the left side of the window, you can open the VNC client's configuration dock by clicking on the small arrow symbol in the center of the window. You can use it to switch the virtual desktop to full-screen mode, for example. The client automatically detects the input devices so that you can use the mouse in windowed mode on both the local and the virtual system without needing to learn any keyboard shortcuts.

The virtual systems contain the same applications as the original, and they can be used without restrictions.

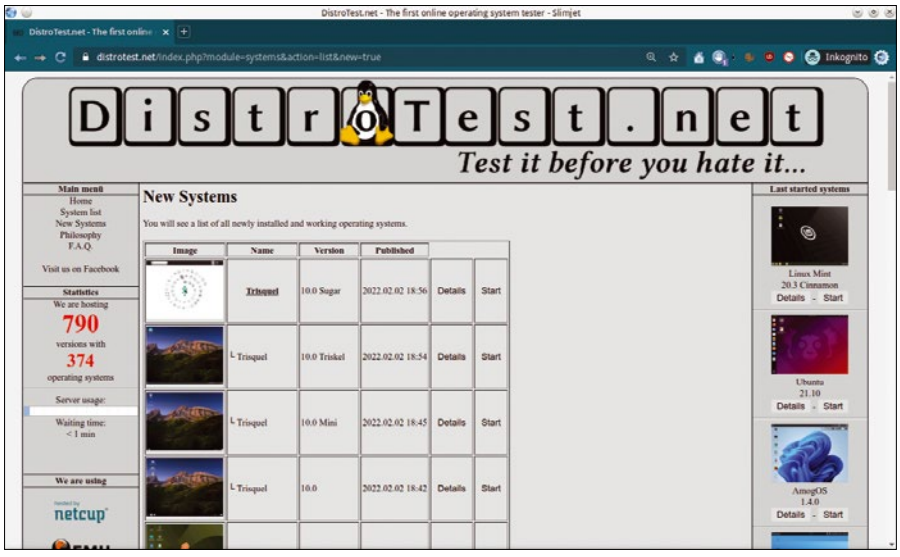


Figure 3: The project quickly updates new versions in the system.

However, depending on the available bandwidth, noticeable latencies sometimes occur, especially if you decide to launch a large application such as Gimp, Firefox, or LibreOffice.

Willing to Change

If you want to terminate a distribution after reviewing it, open the configuration dock on the left side of the noVNC viewer and terminate the session by clicking on *Disconnect* and then close the window. Clicking on *Home* in the top left corner takes you back to the distribution list. The *New Systems* link at top left on the main page takes you to the most recently added systems. These systems are listed chronologically in the window's main pane with the latest distributions always at the top of the list (Figure 3). This means you can always see which projects have released new versions in the past few days.

Conclusions

Thanks to DistroTest.net, you can easily test out different Linux distributions to find the one that best meets your needs. Because the individual distributions are ready to use, you can save the time-consuming overhead of downloading and installing multiple distributions. DistroTest.net is therefore highly recommended as the first port of call for anyone looking for an overview of the various Linux derivatives. ■■■

Info

[1] DistroTest.net: *https://distrotest.net*

Exploring the BlueSpice wiki tool

Spice Is Nice

Keep your team collaborating with BlueSpice, a wiki tool focused on professional customers. *By Martin Loschwitz and Markus Feilner*

MediaWiki [1] is the open source, highly collaborative knowledge management tool that runs the world's sixth largest website: Wikipedia. Wiki solutions such as MediaWiki make it easy for large communities to collaborate, offering common access to an information store with built-in editing tools and version control.

MediaWiki is designed to serve a vast community on a global scale, but what if you need to serve a less diverse audience of users in a local or organizational setting? Wikis are excellent tools for business because they support central storage of shared information and eliminate the knowledge bottlenecks that occur when knowledge is shared haphazardly through email and personal connections.

Several open source projects support wiki tools for business environments, including XWiki, DokuWiki, and BlueSpice (for a comparison of these projects, see [2]). BlueSpice, from Hallo Welt!, is a German MediaWiki distribution that focuses on usability in enterprise environments. BlueSpice 4.1 [3], released in early 2022, offers a large set of features. The best way to determine whether a tool such as BlueSpice is right for your company is to set it up and explore it. In this article, we'll show how to set up BlueSpice in a typical enterprise setting.

More on BlueSpice

BlueSpice comes in a free version (BlueSpice free) and a commercial version (BlueSpice pro). BlueSpice free, which is the primary focus for this article, is an extension built on top of MediaWiki. Whereas MediaWiki is very simple and austere (Figure 1), BlueSpice free offers several additional options, along with a more sophisticated look and feel (Figure 2).

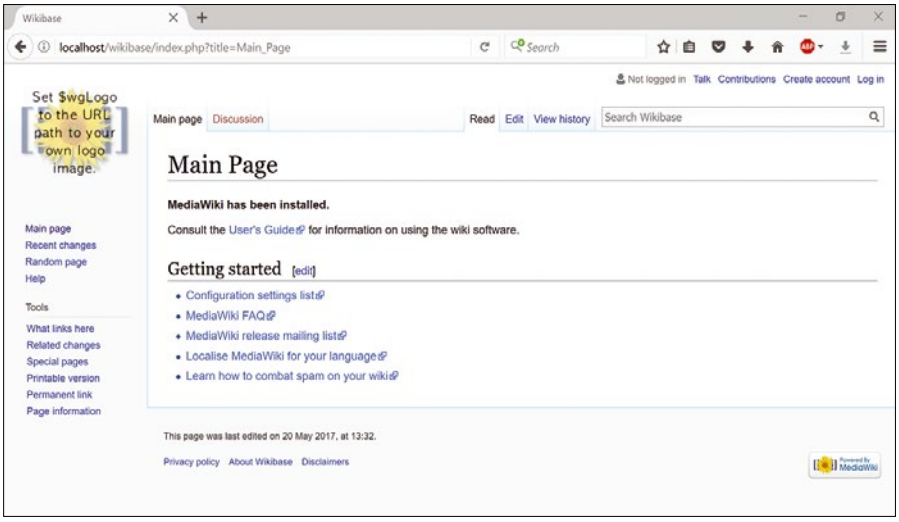


Figure 1: MediaWiki has a clear but unadorned user interface.

Deploying BlueSpice

You have two options for getting BlueSpice up and running. The "classical" way is to set up BlueSpice on a working LAMP stack. To get BlueSpice up and running, you simply extract BlueSpice to the `DocumentRoot` directory of your web server, start its built-in configuration agent (in a browser), and enter relevant data (such as the database to use along with the required credentials).

However, if you don't happen to have a working LAMP stack on hand, BlueSpice also offers a Docker image of BlueSpice free 4.1 that contains all the necessary components. As this Docker image comes straight from Hallo Welt! and the image's sources are also publicly available, it's easy to verify what the image does internally, doing away with the latent danger of running a "black box" on your hardware. Preparations for getting BlueSpice up and running in Docker are a bit different from what you usually do in non-Docker environments, but overall, the Docker image should be less work than configuring a working version of BlueSpice on an existing LAMP stack.

Hardware

Although BlueSpice can cause quite a bit of CPU and RAM load, it usually isn't necessary to supply the software with huge amounts of hardware. Dedicating a physical machine will almost always be overkill. Only a huge enterprise with thousands of concurrent users accessing the tool will see BlueSpice consume dozens of CPU cores or hundreds of gigabytes of RAM. The ideal (and normal)

Photo by Tamanna Rumee on Unsplash

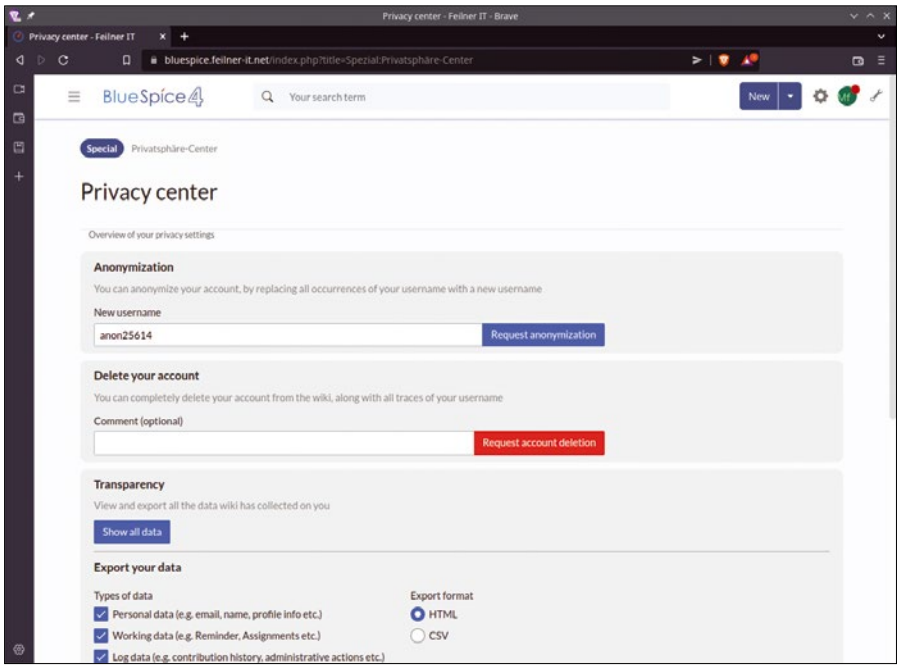


Figure 2: Bluespice free offers several options that aren't part of Mediawiki, such as the privacy center and an a ElasticSearch engine.

deployment scenario is inside of a virtual machine (VM). You can use standard tools such as VMware, Proxmox, or KVM-based virtualization as a VM in one of the hyperscaling public clouds (e.g., Amazon Web Services, Azure, etc.).

However, you should pick a storage device for your BlueSpice VM that is replicated on the infrastructure level. Otherwise, once BlueSpice is up and running, there would be no way to ensure proper high availability (HA) for the service.

For an average-sized BlueSpice setup, a small machine with eight (virtual) CPU cores and 16GB of RAM should be absolutely sufficient. As the amount of stored articles in the wiki grows, RAM needs might increase. Inside the container, BlueSpice uses ElasticSearch for searching and indexing; the more data, the more resources you will need. If you are running BlueSpice inside of a VM, and the VM runs out of resources, just add more virtual resources to the system.

Definitions

In this article, we'll set up BlueSpice on the recently released Ubuntu Linux 22.04. The steps to get BlueSpice up and running include installing the Docker Community Edition on your system, configuring a persistent storage volume, preparing your SSL certificates, and finally deploying BlueSpice itself. Afterward, you will also want to ensure your container is running permanently by using a systemd unit file. This is the only way to ensure that a crash of the system hosting the BlueSpice VM can be compensated by automatically starting the virtual instance elsewhere and without requiring human intervention.

Preparing Ubuntu

This article assumes that you have just logged in to a freshly installed Ubuntu Linux 22.04 (for the first time). Docker has already made packages for its container runtime available for the brand new Ubuntu release, so the next step is to prepare the system for these and then install them.

As the initial step to prepare the system, add Docker's PGP key to the list of keys accepted by the Apt package manager as trustworthy. Only after this will `apt` download packages from the Docker repositories. To enable the key, use

```
curl -fsSL https://download.docker.com/
  linux/ubuntu/gpg | sudo gpg --dearmor
  -o /usr/share/keyrings/
  docker-archive-keyring.gpg
```

To add the required repository definitions to your system, use

```
echo "deb [
  arch=$(dpkg --print-architecture)
  signed-by=/usr/share/keyrings/
  docker-archive-keyring.gpg]
  https://download.docker.com/
  linux/ubuntu $(lsb_release -cs)
  stable | sudo tee /etc/apt/
  sources.list.d/docker.list > /dev/null
```

Then, `apt update` will update the list of packages known to your local package manager before finally downloading the required packages with:

```
sudo apt install docker-ce
```

Next, run `sudo docker ps` to check whether the Docker runtime is installed and running. You should see no output because there are no containers running yet, but there should also not be an error message.

Preparing Your Storage

A Docker volume needs to be created and inserted into the container to provide the container with persistent storage inside of the VM (remember to assure HA storage on the infrastructure level through network filesystems, NAS, or SAN). This is necessary because without a persistent storage device, BlueSpice data would disappear after a BlueSpice container is deleted and recreated. In fact, this will happen when updating the Docker container provided by BlueSpice. Hence, this article will assume that a storage device named `/dev/sdb` is available within the instance, that it is replicated outside of the virtual instance, and that said device is dedicated to BlueSpice usage only.

The next steps include a few basic tasks: First, create a partition table on the device with a single partition spanning across all of it, resulting in the existence of `/dev/sdb1`. XFS will be used for BlueSpice data, so the *xfsprogs* package will have to be installed next with

```
apt install xfsprogs
```

Then, a filesystem needs to be created on said partition, using

```
mkfs.xfs -L DATA /dev/sdb1
```

The `-L` parameter ensures that the new filesystem has a label that allows you to refer to the filesystem without knowing the full device path. This is relevant for the next step, which is to ensure that the

XFS filesystem is automatically mounted at boot time. To do this, use

```
mkdir -p /srv/data
```

and the line

```
LABEL=DATA /srv/data xfs defaults 0 0
```

in `/etc/fstab`. Once done, `mount -a` should finally mount the XFS partition to the system, and the drive will be automatically mounted after a reboot. The final step is to allow Docker to create volumes on this device:

```
docker volume create ➥
  --driver local ➥
  --opt type=none ➥
  --opt device=/srv/data ➥
  --opt o=bind bluespice-data
```

Deploying BlueSpice

Thanks to Docker, the next step is deploying BlueSpice – no need to worry about a web server, etc. But for the sake of this example, a few extra settings should be taken into consideration. A BlueSpice instance facing the Internet must support SSL encryption under all circumstances (e.g., to ensure that passwords are not transmitted in plaintext format). BlueSpice is configured to accept external SSL certificates out of the box, but they have to be available to the BlueSpice container.

Listing 1: Systemd example unit file for BlueSpice

```
[Unit]
Description=BlueSpice service
After=docker.service
Requires=docker.service

[Service]
TimeoutStartSec=0
Restart=always
Environment="bs_lang=en"
Environment="bs_url=https://www.example.net"

ExecStartPre=-/usr/bin/docker exec %n stop
ExecStartPre=-/usr/bin/docker rm %n
ExecStartPre=/usr/bin/docker pull owncloud/ocis
ExecStart=/usr/bin/docker run --rm --name %n \
  -ti -p 80:80 -p 443:443 \
    -v bluespice-data:/data \
    bluespice/bluespice-free

[Install]
WantedBy=default.target
```

Three files are required: `ssl.cert` containing the actual SSL certificate, `ssl.key` containing the corresponding key, and `ssl.ca` if the SSL certificate used requires an intermediate CA certificate to establish the chain of trust.

For the BlueSpice setup, placing the certificates in the previously created folder `/srv/data` is sufficient. That folder will later be inserted into the container as `/data` using a bind mount. `/data` is where BlueSpice looks for its SSL certificate files, so you're done. You can now pull the Docker image for BlueSpice free from Docker Hub using:

```
docker pull bluespice/bluespice-free
```

Firing Up the Docker Image

To deploy BlueSpice, use the most important command in this procedure:

```
sudo docker run --rm --name bluespice ➥
  -ti -p 80:80 -p 443:443 ➥
  -v bluespice-data:/data ➥
  -e "bs_lang=en" ➥
  -e "bs_url=https://www.example.net" ➥
  bluespice/bluespice-free
```

The parameters appended to the `docker` command are straightforward. Ports 80 and 443 must be assigned to the Docker container running BlueSpice, so use `-p 80:80` and `-p 443:443`. Docker will now automatically configure port forwarding from the host's IP address to the container's internal (and virtual) address.

To set the language to English, pass in the environmental variable "`bs_lang=en`". Knowledge management tools will be better accepted by staff members when the interface uses a known language. However, even if the standard language in the container is English, users will still be able to configure their profile's language.

Following the `bs_url` parameter, you will need to replace `https://www.example.net` with the full URL to your newly created Wiki to meet local needs. After a few seconds, BlueSpice will be up and running, using a standard login of *WikiSysop* and a standard password of *PleaseChangeMe*. Change those values as quickly as possible; they are documented in the official BlueSpice installation guide and hence publicly available. To do so, log in to BlueSpice with a browser using the URL you have inserted into the command and change the password in the account's profile.

Enable BlueSpice at Boot Time

Like most modern Linux distributions, Ubuntu uses systemd. To ensure that a Docker container is automatically started when the system boots, a systemd unit file is required. Listing 1 shows the example for the BlueSpice deployment as described in this article.

To make systemd recognize the new unit file, place the contents of Listing 1 in a file named `/etc/systemd/system/bluespice.unit` and run

```
sudo systemctl daemon-reload
```

To restart Docker and all systemd units depending on it, use

```
sudo systemctl restart docker
```

From now on, whenever the system reboots, BlueSpice will automatically launch afterwards.

It makes sense to have a look at the configuration options provided by BlueSpice in its administration back end. The software tries to use generic defaults wherever possible. Even though they work fine, a few default settings might need changes for special setups. Note that Hallo Welt! also offers a migration path and tool to migrate existing content from Confluence to BlueSpice. A set of scripts automate the migration as much as possible [4] .

BlueSpice free and BlueSpice pro

Besides BlueSpice free, Hallo Welt! also offers a commercial version, BlueSpice pro, which is targeted at corporate and enterprise users and offers support and services. From an administrator's viewpoint, the most notable difference is BlueSpice pro's role-based access control (RBAC) implementation that can also connect to external user directories such as LDAP or Active Directory.

BlueSpice free only supports a local user database and comes with predefined "permission sets" that may be applied to pages, users, and groups. BlueSpice pro features a semantic search using the official MediaWiki plugin, along with the ability for semantic browsing.

Apart from that, a few convenience features are missing in BlueSpice free, such as the ability to embed media players in pages, the display of mathematical formulas using LaTeX, or the ability to directly attach files to pages using drag and drop. To highlight its corporate user target group, BlueSpice pro also comes with a number of functions related to compliance. BlueSpice free does not offer blog entries, making it impossible to use BlueSpice free as a corporate blog, and texts in BlueSpice Free cannot be rated. In addition, all features related to document quality assurance and audit trails are missing from BlueSpice free. Accordingly, document reviews, document read confirmations, and digital signatures using SignHere [5] are also unavailable, as is the option to see whether a certain page version is the most recent, officially approved version of the document. A full list of features is available on the BlueSpice website [3].

BlueSpice vs. MediaWiki

A quick look at the differences between BlueSpice software and its predecessor, MediaWiki, shows the different target groups. MediaWiki continues to manage and run the largest collection of knowledge in the world; the software is proving its quality every day. Naturally, MediaWiki has been highly optimized for usage within Wikipedia. Its design may appear rather traditional, even conservative in the eyes of some users, and certainly MediaWiki is very different from other tools such as Confluence. MediaWiki's development is focused on the needs and standards of the encyclopedia, not on the needs of administrators or businesses.

BlueSpice, on the other hand, targets enterprise users. So it should come as no surprise that BlueSpice's visual appearance in its default setup is by far the biggest difference between the two solutions. In stark contrast to MediaWiki, BlueSpice makes no secret that it is intended to be a company wiki rather than an encyclopedic tool; it is a Confluence alternative built on MediaWiki. BlueSpice pro adds advanced tools (Figure 3) and extra focus on compliance.

Because MediaWiki does not offer (official) containers, getting it up and running is a bit more complex. In order to install MediaWiki, a full LAMP deployment along with the execution of MediaWiki's setup script is necessary. In the past, some issues with PHP libraries have been reported, especially when "PHP hell" made upgrades difficult. All of these are issues that BlueSpice might share, but delivery as a container removes those challenges for the admin.

Those factors aside, it's important to remember that BlueSpice is MediaWiki with a number of MediaWiki extensions added out of the box and a revamped user interface. Most of the features available in BlueSpice (both free and pro, for that matter) could be implemented using MediaWiki, too. That would, however, consume large amounts of time and a rather steep learning curve – a luxury most admins will likely not have in today's fast-paced business.

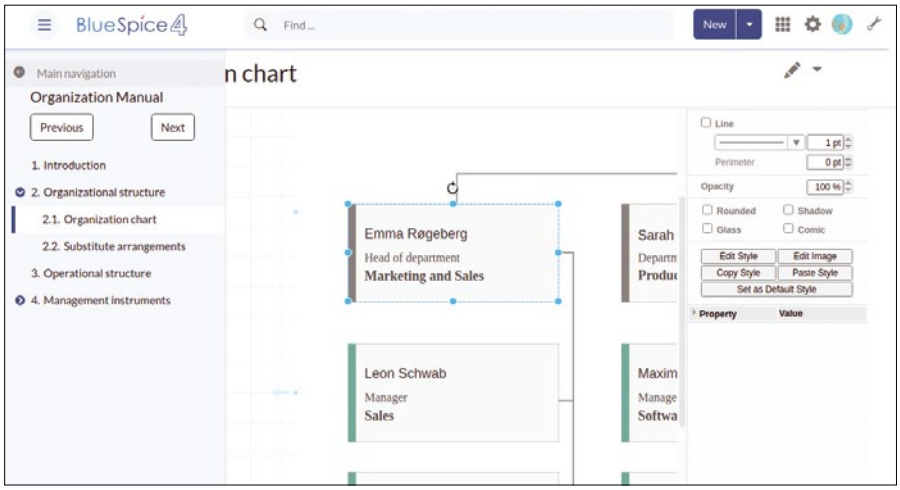


Figure 3: Thanks to open source draw.io integration, BlueSpice pro brings interactive charts and workflow graphics to the browser.

Summary

For organizations looking for an on-premises wiki-style solution, projects such as XWiki, DokuWiki, and BlueSpice prove that there is an open source alternative to Confluence to meet their business needs. BlueSpice offers a modern knowledge management system that is easy to deploy as a Docker container. Getting BlueSpice free up and running is a straightforward and convenient process.

Although BlueSpice free lacks a number of features relevant for corporate and enterprise users, it may still be a valid alternative for companies looking to move away from Atlassian and Confluence. If you are willing to pay for BlueSpice pro, you can use the same procedures outlined in this article; Hallo Welt! also offers Docker containers for BlueSpice pro. ■■■

Info

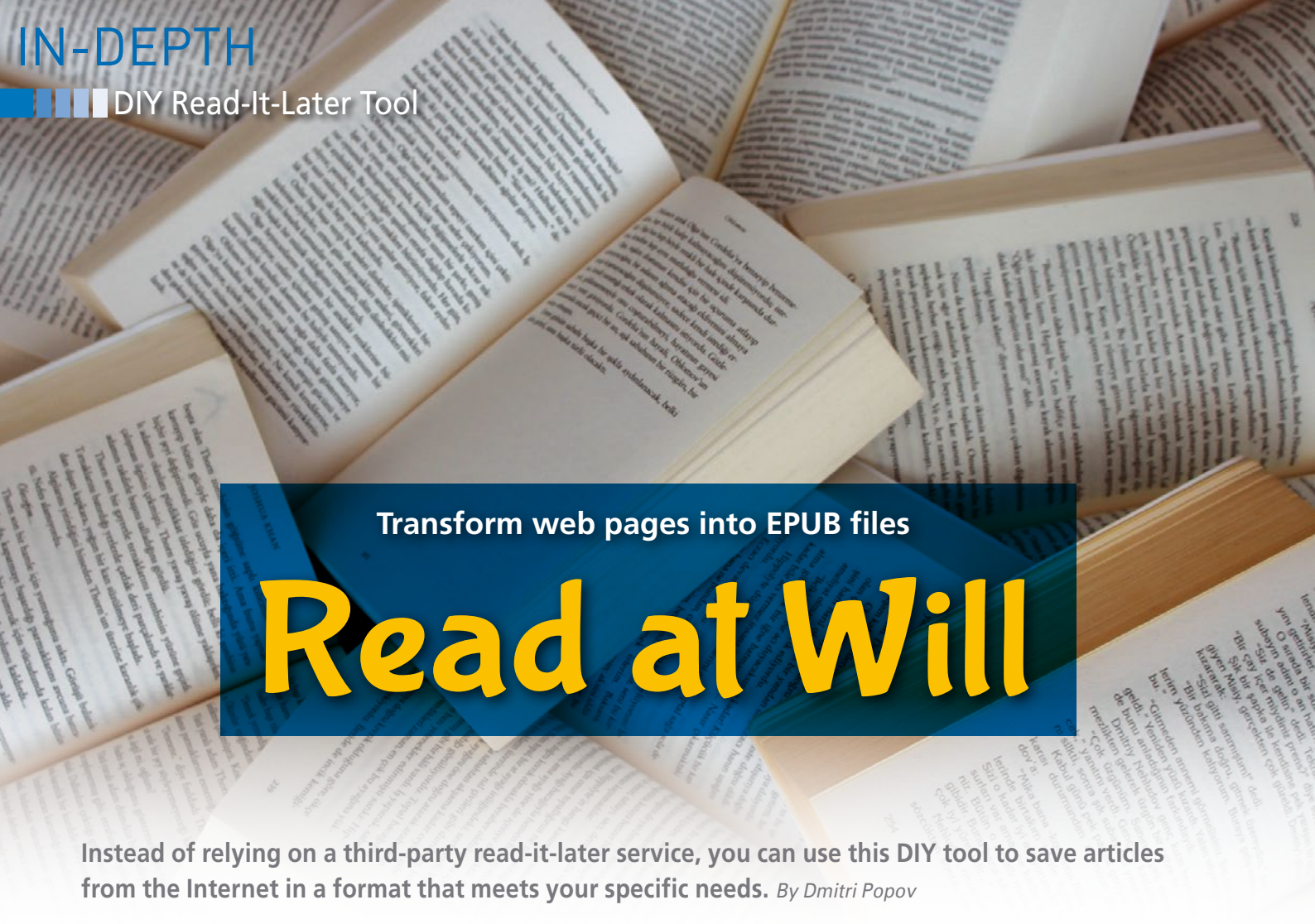
[1] MediaWiki: *https://www.mediawiki.org/wiki/MediaWiki*
[2] "3 open source alternatives to Confluence" by Martin Loschwitz, September 15, 2020: *https://opensource.com/article/20/9/open-source-alternatives-confluence*
[3] BlueSpice: *https://bluespice.com*
[4] Confluence to BlueSpice migration: *https://github.com/hallowelt/migrate-confluence*
[5] SignHere: *https://www.signhere.be/en/*

Author

Martin Gerhard Loschwitz has been working as an open source journalist for more than two decades. He covers topics such as OpenStack, Ceph, and Kubernetes.

Author

Markus Feilner has been working with Linux and open source software since 1994. He was deputy editor-in-chief of the German *Linux Magazine* and *IX* and documentation team lead at SUSE. He has run his company, Feilner IT, specializing in documentation, digital sovereignty, and OSI layers 8, 9, and 10, for 22 years.



Transform web pages into EPUB files

Read at Will

Instead of relying on a third-party read-it-later service, you can use this DIY tool to save articles from the Internet in a format that meets your specific needs. *By Dmitri Popov*

Few of us have time to read long-form web articles during the day, which is why services that let you save interesting reads for later can come in handy. Popular services such as Pocket and Instapaper even offer apps you can use to read the saved content offline on your preferred device. Better still, the saved articles are reformatted for better readability and scrubbed of all ads, scripts, trackers, and other junk.

Hosted services are like restaurants, though. No matter how great the food and the service, you eventually start longing for home-cooked meals, not only because cooking at home is cheaper and more convenient, but because you can make any dish you wish just the way you like it and have fun in the process. In a similar vein, why settle for a ready-made, read-it-later service, when you can cook up your very own solution with a bit of creative thinking, the right mix of open source tools, and a dash of shell scripting magic? That's exactly what is on today's menu: a DIY read-it-later tool.

Instead of saving and serving slimmed down versions of web pages, this DIY read-it-later application is going to process pages and transform them into ePub files. This way, you can read the saved content on practically any device, and you can choose whatever ebook reading app you like. Because the DIY read-it-later tool is a simple shell script that relies on Linux tools, you don't need a server to host it. If necessary, you can run the tool on a remote Linux machine and serve ePub files via a dedicated Open Publication Distribution System (OPDS) server or simply publish the files on the web. In short, the DIY read-it-later tool gives you plenty of room for experimenting and setting up the solution that works best for your specific needs. Moreover, the fact that an ePub file is essentially a ZIP archive containing an XHTML file along with stylesheets, fonts, and so on makes the saved content future-proof and editable.

Preparatory Work

You don't have to code the DIY read-it-later tool from scratch, because I've already done the hard work for you and published the fruits of my labor, `readiculous.sh`, on GitHub [1]. All you need to do is download the source code as a ZIP archive and unpack it, or clone the project's Git repository using the command:

```
git clone https://github.com/dmpop/readiculous.git
```

Before getting down to the nitty-gritty, you need to do some preparatory work. The first order of business is to install the required software. The main `readiculous.sh` shell script relies on Pandoc, ImageMagick, `jq`, `wget`, and Go-Readability [2]. With the exception of Go-Readability, all of these dependencies are available in the official software repositories of most mainstream Linux distributions, so you can install them using the default package manager. To do this on Debian or an Ubuntu-based distribution, run the command:

```
sudo apt install pandoc imagemagick jq wget
```

The source code on GitHub [1] includes a binary version of the Go-Readability tool compiled for the x86_64 architecture. If you plan to use the script on any other platform, or you want to have the very latest version of the tool, you will have to compile it yourself. Fortunately,

Photo by Gülfer ERGİN on Unsplash

it's a rather straightforward thing to do. Install the Go language package (use the `sudo apt install golang` command on Debian and Ubuntu), and then run the following command to compile the command-line version of Go-Readability:

```
go get -u -v github.com/go-shiori/go-readability/cmd/...
```

Once the compiling process is finished, you'll find the resulting binary in the `~/go/bin` directory. Move the binary file into the `readiculous` directory, and you're done.

How It Works

The `readiculous.sh` script (Listing 1) starts working by fetching the desired

Listing 1: readiculous.sh

```
01 #!/usr/bin/env bash
02 if [ ! -x "$(command -v convert)" ] || [ ! -x "$(command
       -v pandoc)" ] || [ ! -x "$(command -v jq)" ]; then
03   echo "Make sure that the required tools are installed"
04   exit 1
05 fi
06
07 # Usage prompt
08 usage() {
09   cat <<EOF
10 $0 [OPTIONS]
11 ------
12 $0 transforms web pages pages into readable EPUB files.
13
14 USAGE:
15 ------
16   $0 -u <URL> -d <dir> -m auto
17
18 OPTIONS:
19 --------
20   -u Source URL
21   -d Destination directory (optional)
22   -m Enable auto mode (optional)
23
24 EXAMPLES:
25 ---------
26 $0 -u https://psyche.co/guides/
   how-to-approach-the-lifelong-project-of-language-learning
   -d "Language"
27 $0 -m auto
28
29 EOF
30   exit 1
31 }
32
33 #Read the specified parameters
34 while getopts "u:d:m:" opt; do
35   case ${opt} in
36   u)
37     url=$OPTARG
38     ;;
39   d)
40     dir=$OPTARG
41     ;;
42   m)
43     mode=$OPTARG
44     ;;
45   \?)
46     usage
47     ;;
48   esac
49 done
50 shift $((OPTIND - 1))
51
52 if [ ! -z "$dir" ]; then
53   dir=Library/"$dir"
54 else
55   dir=Library
56 fi
57 mkdir -p "$dir"
58
59 readicule() {
60   # Extract title and image from the specified URL
61   title=$(./go-readability -m $url | jq '.title' |
           tr -d \")
62   # Generate a readable HTML file
63   ./go-readability $url >>"$dir/$title".html
64   # Generate a cover
65   wget -q https://picsum.photos/800/1024 -O cover.jpg
66   convert -background '#0008' -font Arvo -pointsize 35
      -fill white -gravity center -size 800x150
      caption:"$title" cover.jpg +swap -gravity south
      -composite cover.jpg
67   if [ -z "$title" ]; then
68     title="This is Readiculous!"
69   fi
70   # convert HTML to EPUB
71   pandoc -f html -t epub --metadata title="$title"
      --metadata creator="Readiculous" --metadata
      publisher="$url" --css=stylesheet.css
      --epub-cover-image=cover.jpg -o "$dir/$title".epub
      "$dir/$title".html
72   rm cover.jpg "$dir/$title".html
73   echo
74   echo ">>> '$title' has been saved in '$dir'"
75   echo
76 }
77
78 # If "-m auto" is specified
79 if [ "$mode" = "auto" ]; then
80   file="links.txt"
81   if [ ! -f "$file" ]; then
82     echo "$file not found."
83     exit 1
84   fi
85   # Read the contents of the links.txt file line-by-line
86   while IFS="" read -r url || [ -n "$url" ]; do
87     readicule
88   done <"$file"
89   rm links.txt
90   exit 1
91 fi
92
93 if [ -z "$url" ]; then
94   usage
95 fi
96
97 readicule
```

page, scrubbing it clean, and reformatting it for better readability. To do all that, the script uses the nifty Go-Readability tool. Go-Readability also extracts the page title and passes it to ImageMagick, which creates a cover image with the obtained title. Finally, the Pandoc tool transforms the saved page into an ePub file complete with the generated cover.

The script accepts three parameters: `-u`, `-d`, and `-m`. The mandatory `-u` parameter specifies the URL of the target page, while the optional `-d` parameter determines in which subdirectory the resulting ePub file should be saved. If the `-d` parameter is omitted, the script saves ePub files in the default `Library` directory. By specifying the subfolder, you can automatically sort the created ePub files by topic (for example, Language, Travel, Long Reads, and so on), or any other criteria. The `-m` parameter allows you to convert several saved URLs at once, but I'll take a closer look at it later. The script uses a combination of the `getopts` tool, the `do...done` loop, and the `case in` control structure to read the values passed by the specified parameters and assign these values to variables (lines 34-50 in Listing 1). If the default `Library` directory doesn't exist, the script creates it (lines 52-57).

Listing 1's `readicule()` function does the actual work. First, Go-Readability obtains the metadata of the specified page. The metadata is returned in the JSON format, and the `jq` tool extracts the title, while the `tr` tool strips double quotes (line 61). The same Go-Readability tool fetches the page using the specified URL and saves the processed version as an HTML file (line 63).

The next step is to create a cover for use with the ePub file. Strictly speaking, covers are not necessary, but they do make it easier to find the file you need in the library, and they make the ePub file look less bland. To generate a cover, the script uses the `wget` tool for fetching a random 1024x800 image from the Lorem Picsum service and saves the file as `cover.jpg` (line 65). Then, the `convert` tool superimposes the obtained title onto the cover image (line 66).

There are, of course, plenty of other ways to create covers if you don't want the script to rely on a third-party service. For example, you can create covers with random background colors. To do this, you need to tweak the script so that it generates three random numbers between `0` and `255`. The `convert` tool can then use the numbers as red, green, and blue values for generating a cover:

```
r=$(shuf -i 0-255 -n 1)
g=$(shuf -i 0-255 -n 1)
b=$(shuf -i 0-255 -n 1)
convert -size 800x1024 xc:rgb\(
  $r,$g,$b\) cover.jpg
```

If solid colors are not your cup of tea, you can use the `convert` tool to generate a random colorful fractal image and specify the `-paint` and `-blur` options for a more artistic effect:

```
convert -size 800x1024 plasma:fractal
  -paint 10 -blur 10x20 cover.png
```

Finally, Pandoc finishes the task. It assembles the saved HTML file, the generated cover, and the obtained data into an ePub file and saves it either in the default directory (line 71) or in the subdirectory specified by the `-d` parameter.

But that's not all. If you read a lot, running the script every time you want to save a page for later can quickly become a nuisance. That's why the script also features the `-m` parameter. When specified with the `auto` value, the script picks URLs from the `links.txt` file one by one and generates ePub files for each one. The `if...then...fi` block that starts on line 79 checks whether the `$mode` value is set to `auto`. If so, the `while...do` loop (lines 86-90) reads URLs from the `links.txt` file and calls the `readicule()` function to generate ePub files. If the `$mode` value is not specified, the script simply calls the function to generate an ePub file using the URL passed by the `-u` parameter.

To speed up the process of transforming articles into ePub files, you can create a simple helper script:

```
#!/usr/bin/env bash
url=$(xclip -o)
echo $url
cd /path/to/readiculous
./readiculous.sh -u $url
notify-send "Added to Readiculous"
```

Replace `/path/to/readiculous` with the actual path to the `readiculous` directory, and save the script under an appropriate name (for example, `add-to-readiculous.sh`). Install the `xclip` tool on your system, and assign a keyboard shortcut to the script.

The Matter of Reading

Saving articles in the ePub format means that you read them using practically any device on any platform. Better yet, if you

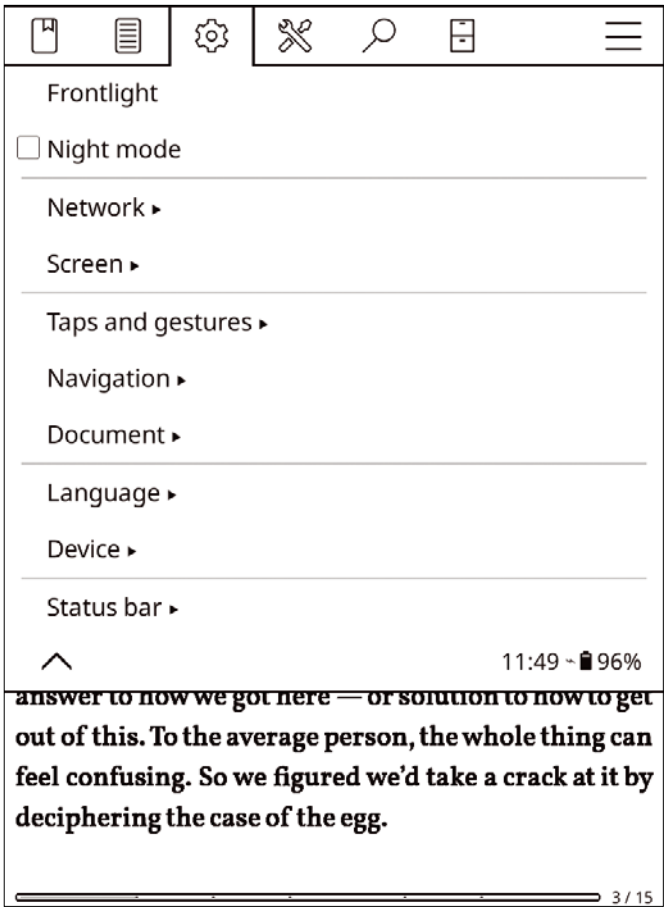


Figure 1: KOReader is arguably the most powerful open source ebook reader on any platform.

use Apple Books or Google Books, you can take advantage of the features these apps offer, including synchronization across multiple devices, saving highlights, library management functionality, and more.

However, if you've gone to the trouble of rolling out your own read-it-later tool, it probably doesn't make much sense to use a third-party commercial platform for reading. Enter KOReader [3], an open source ebook reader application available for Linux, Android, and a slew of dedicated readers. Despite its deceptively simple interface, KOReader packs an impressive array of features, including syncing, highlights, gesture support, note-taking capabilities, extensions, and much, much more (Figure 1). So if you want to keep your entire read-it-later toolchain open source, you should use KOReader.

Remote Work

Transferring the saved articles in the ePub format from your machine to a dedicated reading device such as Kobo or Kindle is as straightforward as it gets. However, you might want to host your library on a remote server, so you can access the saved content from anywhere. While you can set up a fully-featured OPDS server on a remote machine using applications such as Calibre, a simpler solution would do the job just fine. You can modify the script to have it sync the contents of the `Library` directory to a remote web server. Here's an example command that uses the `rsync` tool to back up the `Library` directory to the remote web server via SSH:

```
rsync -avhz --delete -P -e "ssh
  -p 22" /readiculous/Library/
  user@127.0.0.1:/var/www/html/Library
```

Theoretically, you'd need a proper `index.html` page for the published library, but the ingenious h5ai [4] software can take care of that. Drop h5ai into the document root of your web server, and it will render any directory without an index page as a sleek and easy-to-navigate catalog (Figure 2). You can then access your library using a regular browser and download the desired ePub file to your reading device.

Similarly, you can install `readiculous.sh` onto a remote Linux server and run it from anywhere via SSH with one additional tweak. In its original form, the `readiculous.sh` script uses the Arvo font for generating cover titles. To install the font on a remote machine running Debian or an Ubuntu-based server, put the appropriate `.ttf` font file into the `/usr/local/share/fonts` directory and run the command:

```
fc-cache -f -v
```

Of course, establishing an SSH connection to a remote server to run the script every time you want to save an article is somewhat impractical. The supplied `add.php` PHP script provides a solution to the problem (assuming that your web server supports PHP). The script adds a simple HTML form that makes it possible to add URLs to the `links.txt` files. You can then run the `readiculous.sh` script with the `-m auto` parameter to convert the saved links to ePub files in a single action. Even better, you can create a cron job that performs the operation at regular intervals. For this setup to work, you need to adjust the path to the `links.txt` in the `readiculous.sh` script.

Closing Remarks

While `readiculous.sh` does lack the elegance and advanced functionality of commercial read-it-later services, it provides something more important: full control over your data and tools, limitless tweaking options, and the fun of working on your own coding projects. ■■■

Info

[1] readiculous.sh on GitHub: *https://github.com/dmpop/readiculous*

[2] Go-Readability: *https://github.com/go-shiori/go-readability*

[3] KOReader: *https://koreader.rocks*

[4] h5ai: *https://larsjung.de/h5ai/*

Author

Dmitri Popov has been writing exclusively about Linux and open source software for many years, and his articles have appeared in Danish, British, US, German, and Spanish magazines and websites. You can find more on his website at *tokyoma.de*.

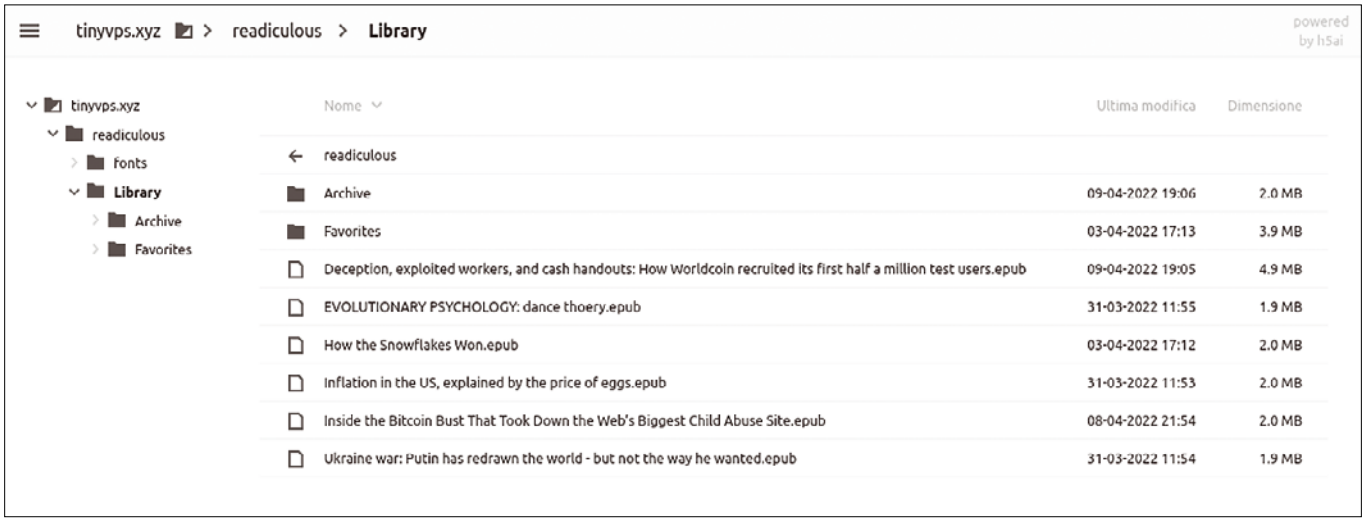


Figure 2: h5ai turns any directory on a web server into a sleek catalog.

Boost your Wordle streak with Go

Wordle Booster

Wordle, a simple online word game, took the world by storm in February. Mike Schilli has developed a command-line tool to boost his Wordle streak using some unapproved tactics.

By Mike Schilli

It's often difficult to predict what will fail and what will generate massive hype, but no one would have expected the Scrabble-style Wordle [1] game to become an Internet hit in 2022. Entire magazine articles deal with the phenomenon [2], people at work show off their results every day, and the *New York Times* acquired the game from the developer for millions of dollars [3].

The game involves guessing a five-letter word. If the player enters the word correctly, they win the game. If the player's guess doesn't quite get the word, the game marks letters that are in the word and in the right place in green and letters that are in the word but in the wrong place in yellow. Wordle marks guessed letters that do not appear in the target word in gray. To win, you have to guess the word in no more than six attempts, using the app's clues in the smartest way possible.

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.

Automate Me!

Worlde picks words from the Scrabble dictionary, but excludes conjugations or plurals as solutions. In the English version, *CAMEL* would appear, but not *CAKES*. Alas, Wordle will accept all five-letter Scrabble words, including plurals, as guesses, of which there are 12,972 in the English language [4].

The game's simple rules make writing a computer program to run against the puzzles a no-brainer. After all, Wordle color-codes the guesses, and a program can use these clues to filter out entries from a complete word list, whittling down the options in each round. The player just needs to pass in the Wordle clues to a helper program in a coded format: I'll use `2` for a letter match, `1` for a letter in the wrong place, and `0` for letters that do not occur at all. For example, if the puzzle solution is *CAMEL* and I guessed *LAMER*, the code would be `12220`: The *L* at the beginning of the guess is in the wrong place, the middle letters *AME* are spot on, and the final *R* does not occur in the target word.

In Figure 2, my `wordle` cracker program suggests *OATER* as the first word. The reason for this unusual word is that it contains a bunch of common vowels, and

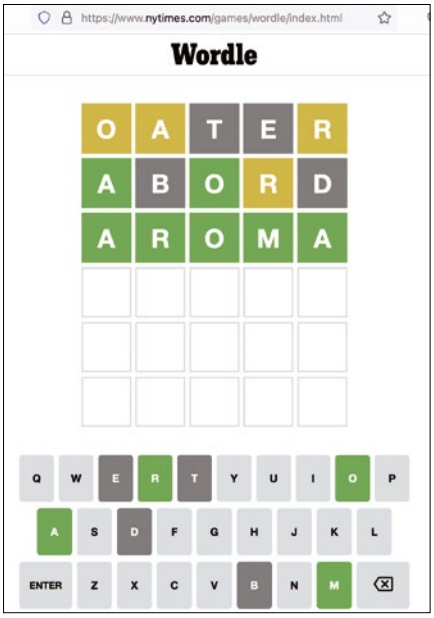


Figure 1: This Wordle was solved in three guesses …

```
$ ./wordle
12972 matches ('l' to list).
Try next: 'oater'
hints(12972)> oater 11001
110 matches ('l' to list).
Try next: 'abord'
hints(110)> abord 20210
aroha
aroma
Try next: 'aroha'
hints(2)>
```

Figure 2: … with a little help from a Go program.

Lead Image © bowie15, 123RF.com

checking them right at the start often quickly takes you to a solution – but I'll return to that subject later. On the official Wordle page, Figure 1 shows that the word suggested by the program returns three yellow partial matches for *O*, *A*, and *R*, which means these letters occur in the target word, but in another location. If I now pass in this information to the cracker as `oater 11001`, the Go program then lists `110 matches` (i.e., there are still 110 words left on the list). Everything else has been eliminated based on the evaluation. At this point, you could type `l` to list these 110 words and select one manually, but instead I'll just blindly follow the cracker's choice, *ABORD*.

The Wordle web page in Figure 1 evaluates this new guess with two green full hits for *A* and *O*, while *R* is still in the wrong place as shown by the yellow highlighting. If I now pass in this result to the cracker as `abord 20210`, the cracker immediately decides that only two words are left from the initial list, *AROHA* and *AROMA*, and it displays them without further ado. Because it is unlikely that the *New York Times* would use the New Zealand Maori word for love and affection (aroha) as the solution, I go with *AROMA* and ignore the cracker's suggestion. Figure 1 confirms that this is the right choice.

Rating Algorithm Exposed

How does the Go program whittle down the word list? First of all, my Wordle cracker needs a reference list of valid words to choose from. A list of Scrabble words [4] is a good starting point. Words in the file are in all caps, one per line, of which there are 279,496. Because Wordle only allows five-letter words, line 23 in Listing 1 filters out everything else, leaving us with 12,972 entries. If you're wondering why the Go program determines the length of each word using the function `RuneCountInString()` from the *utf8* package, that's because I might want to support foreign languages one day, and their multibyte characters. If you limit the scope to English, a call to `length()` will do instead.

Listing 1 defines Wordle's supported word length in a constant named `WordleLen` in line 10. In other words, it could easily tackle six-letter Wordles if you just changed one line. The `dictRead()` function starting in line 12 expects the name of the file with the word list. It opens the file for reading in line 15 and uses a line-by-line scanner in line 20 to read all the words one by one starting in line 21. The lists contain words in uppercase, just like the Scrabble tiles, which explains why line 22 converts the words to lowercase to allow you to enter them in lowercase, relieving you from holding down the Shift key for input.

Listing 1: dict.go

```
01 package main
02
03 import (
04   "bufio"
05   "os"
06   "strings"
07   "unicode/utf8"
08 )
09
10 const WordleLen = 5
11
12 func dictRead(file string)
     ([]string, error) {
13   words := []string{}
14
15   f, err := os.Open(file)
16   if err != nil {
17     return words, err
18   }
19
20   s := bufio.NewScanner(f)
21   for s.Scan() {
22     word :=
         strings.ToLower(s.Text())
23     if utf8.RuneCountInString(word)
         != WordleLen {
24       continue
25     }
26     words = append(words, word)
27   }
28   err = s.Err()
29   if err != nil {
30     return words, err
31   }
32
33   return words, nil
34 }
```

Listing 2: wordle.go

```
01 package main
02
03 import (
04   "fmt"
05 )
06
07 const dictFile = "scrabble.txt"
08 const startWord = "oater"
09
10 func main() {
11   dict, err := dictRead(dictFile)
12   if err != nil {
13     panic(err)
14   }
15
16   newWord := startWord
17
18   for round := 0; ; round++ {
19     list(dict, false)
20
21     if round != 0 {
22       newWord = bestBang(dict)
23     }
24     fmt.Printf("Try next: '%s'\n",
                  newWord)
25     fmt.Printf("hints(%d)> ",
                  len(dict))
26
27     var word, score string
28     fmt.Scanf("%s %s", &word,
                  &score)
29     if len(score) == 0 {
30       if word == "l" {
31         list(dict, true)
32       } else {
33         fmt.Printf("Invalid
                      input\n")
34       }
35       continue
36     }
37     dict = filter(dict, word,
                     score)
38   }
39 }
40
41 func list(matches []string, full
             bool) {
42   if len(matches) > 30 && !full {
43     fmt.Printf("%d matches ('l' to
                  list).\n",
                  len(matches))
44   } else {
45     for _, w := range matches {
46       fmt.Printf("%s\n", w)
47     }
48   }
49 }
```

Start Cracking

The `main()` program in Listing 2 defines the Scrabble dictionary file as `scrabble.txt` in the current directory. You can download the file online [4]. The `startWord` variable in line 8 defines the word that the algorithm will suggest first in absence of any hints, and I'll use "oater" because it has a nice assortment of vowels.

The `for` loop starting in line 18 guides the user through the rounds of guesses. The `Scanf()` function in line 28 waits for user input. The user can either type `l` to list the words that are still in the race or enter the result of a guess in the previously discussed `oater 01201` format. For the `list` function, the code calls `list()` starting in line 41 to output the remaining words in the array slice `dict` line by line. The `full` flag here determines whether `list()` displays massively long lists or only those that contain 30 words or fewer. `wordle()` outputs short lists automatically in each round; the full list is only displayed if explicitly requested by the user pressing `l`.

The Machine's Brain

The Wordle cracker's real brain power lies in the `filter()` function (called in line 37 of Listing 2) and whittling down the list of words. The main program passes in the list of words still in play, the current guess, and the evaluation score returned by the online Wordle app. The function returns the word list reduced to reflect the evaluation rules, which the main program immediately assigns again to the words-in-play list `dict`. Following this approach, the program keeps shrinking the word list until only the correct solution remains.

Listing 3 shows the implementation of `filter()`, along with a `grades()` function that, much like the online app, can evaluate a guess attempt against a target word. It tells the user which letters are correct, which are in the wrong place, and which do not occur at all. The `grades()` function returns the rating for a target word in the `want` parameter and for a guess in `guess`, containing a string of numbers such as *01201*.

To determine the rating of a guess attempt, line 16 generates an array slice with integers whose positions match those of the letters in the guess. If there is a `2` in the corresponding position in the array slice, the letter in the guess is in the correct position, and so on. To initialize the array slice, the `for` loop, starting in line 17, first sets all entries to `NoMatch` (i.e., `0`) because of the enum-style constant in line 10, which enumerates the constants in ascending order starting at `0`, thanks to the keyword `iota`.

Line 21 then creates a hash map named

Listing 3: filter.go

```
01 package main
02
03 import (
04   "strconv"
05 )
06
07 type Grade int
08
09 const (
10   NoMatch Grade = iota
11   OtherPos
12   Match
13 )
14
15 func grades(guess, want string) string {
16   hints := make([]Grade, len(guess))
17   for i, _ := range hints {
18     hints[i] = NoMatch
19   }
20
21   wantMap := map[byte]int{}
22
23   // wanted letter counts
24   for i := 0; i < len(want); i++ {
25     wantMap[want[i]] += 1
26   }
27
28   // full matches
29   for i := 0; i < len(guess); i++ {
30     guessRune := guess[i]
31     if guessRune == want[i] {
32       hints[i] = Match
33       wantMap[guessRune] -= 1
34     }
35   }
36
37   for i := 0; i < len(guess); i++ {
38     guessRune := guess[i]
39     if hints[i] == Match {
40       continue
41     }
42     if wantMap[guessRune] > 0 {
43       hints[i] = OtherPos
44       wantMap[guessRune] -= 1
45     }
46   }
47
48   res := ""
49   for _, hint := range hints {
50     res += strconv.Itoa(int(hint))
51   }
52   return res
53 }
54
55 func filter(words []string, guess, graded
                string) []string {
56   res := []string{}
57
58   for _, word := range words {
59     if graded == grades(guess, word) {
60       res = append(res, word)
61     }
62   }
63
64   return res
65 }
66
67 func bestBang(words []string) string {
68   best := ""
69   count := 0
70
71   for _, word := range words {
72     runes := map[rune]bool{}
73
74     for _, rune := range word {
75       runes[rune] = true
76     }
77
78     if count == 0 || len(runes) > count {
79       count = len(runes)
80       best = word
81     }
82   }
83
84   return best
85 }
```

wantMap that counts how many times each letter should be seen in the word. For example, if the word to be guessed is *LOOSE*, it assigns a value of 1 to the letters *L*, *S*, and *E*, and a value of 2 to the letter *O*. The for loop starting in line 29 then goes through all the letters in the guess and sets the corresponding hints entries to 2 if the letter exactly matches the solution in want. Each of these matches decrements the value of the total required number for this letter in wantMap by one.

Starting in line 37, the cracker program evaluates the positions that contain a letter that needs to be somewhere else. If the current position is not a direct hit, but contains a letter from the wantMap, the hints array slice will contain a value of 1 for the current position, and the counter in the wantMap will be decremented by one. If the letter appears again later in the word and its counter in wantMap is still not used up, you would find another 1 there.

When it's all said and done, grades() converts the integer array slice containing the ratings of each letter element by element into a string using the strconv.Itoa() conversion function to return it to the caller as a compact packet. The Wordle website probably uses a similar scoring algorithm.

Filtering by Score

But what does all this have to do with filtering out words from the Scrabble list that are no longer potential solutions based on the Wordle page's evaluation of the guesses? The algorithm in filter() uses the following trick to drop words that are no longer possible: It goes through the remaining list of words entry by entry, assuming in each round that the current entry is the secret solution to the Wordle puzzle. Then line 59 consults the grades() evaluator, asking it what the evaluation of the current guess word is compared to the assumed solution from the word list. Now think about this: If grades() comes back with the same rating as the algorithm from the official Wordle website (which you have because you entered it in its codified format), the assumed solution word is a genuine candidate for the solution. Otherwise, the filter can delete it – simple but effective.

To enable the main program to come up with a good suggestion from the remaining word list on the Try next prompt, the bestBang() function, starting in line 67, picks a word with as many unique letters as possible. This increases the probability that the solution will actually contain one or more of these letters and that the user can eliminate even more wrong solutions in the next step. To do this, bestBang() counts the number of different letters in each word in a hash map named runes. It remembers the word with the highest score (i.e., with the highest entropy) and returns it to the caller when done.

To compile the listings in this article, you just need to call

```
go build wordle.go dict.go filter.go
```

This gives you an executable wordle binary. Note that the cracker does not need any third-party libraries. Instead, it makes do with the pool from the Go standard library. The English Scrabble file required to run the program can be downloaded free of charge from the web [4].

```
$ ./train
To guess: 'lunge'
Guess: 'oater' Grades: [00010]
787 words remaining.
Guess: 'becks' Grades: [01000]
126 words remaining.
Guess: 'dinge' Grades: [00222]
3 words remaining.
Guess: 'lunge' Grades: [22222]
1 words remaining.
Final guess: 'lunge'
Success after 4 guesses.

To guess: 'thump'
Guess: 'oater' Grades: [00100]
311 words remaining.
Guess: 'bhuts' Grades: [02210]
2 words remaining.
Guess: 'thump' Grades: [22222]
1 words remaining.
Final guess: 'thump'
Success after 3 guesses.
```

Figure 3: The Wordle automaton plays games against itself.

Flying Start

If you want to be the first to cross the finish line, you need to get off the starting block quickly – and not only in a 100-meter race. Since Wordle's meteoric rise, scientists have thrown information theory math at the problem to determine the word that gives you a flying start in the game [5].

In general, words with many different letters turn out to be a good choice. At the same time, common letters are extremely valuable because they improve the probability of good tips returned by the Wordle scorer. This why the I started with *OATER* to seed the cracker.

Artificial Intelligence

So much for my proof-of-concept Wordle cracker – but as always, not all options have been exhausted in this Snapshot, because DIY code invites you to experiment. Figure 3 offers a sneak peek into the secret explorations I conjured up after putting the basic solver in place: I let the machine automaton play against itself, by picking a random word and then trying to guess it. This means that new algorithms can be tested in terms of efficiency and with a view to continuous improvement.

And there you have it. The purpose of this article was to illustrate how to quickly put together a proof of concept to help a simple Go program understand the rules of the Wordle game and become an efficient player. Keep in mind that cheating at Wordle is strongly discouraged. Where would be the fun in that?! Remember, the world's best Scrabble players go through great efforts to memorize huge lists of unusual words. There's a reason that computers aren't permitted at the competitions! ■■■

Info

[1] Wordle: *https://www.nytimes.com/games/wordle/index.html*

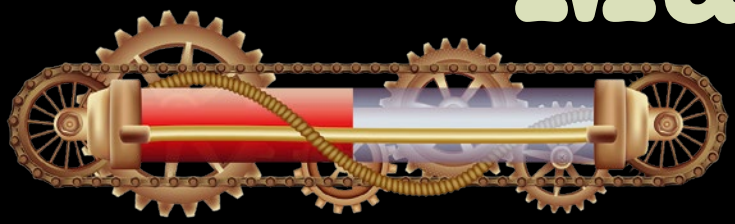
[2] "I Figured Out Wordle's Secret" by Ian Bogost, *The Atlantic*, January 29, 2022: *https://www.theatlantic.com/technology/archive/2022/01/solving-wordle-puzzle/621413/*

[3] "Wordle: New York Times Buys Viral Game for Seven-Figure Sum" by Lois Beckett, *The Guardian*, January 31, 2022: *https://www.theguardian.com/games/2022/jan/31/wordle-new-york-times-buys*

[4] Scrabble word list: *https://boardgames.stackexchange.com/questions/38366/latest-collins-scrabble-words-list-in-text-file*

[5] Best first word: *https://www.inquirer.com/science/wordle-starting-word-answer-win-play-20220203.html*

MakerSpace



Create a digital spirit level with the ESP32

Leveling the Game

The small MPU6050 sensor contains a gyroscope and an accelerometer, which means that you can build a digital spirit level with it. *By Martin Mohr*

To build a digital spirit level, you first need a sensor such as the MPU6050, which determines the position of an object in space. It has an accelerometer and a gyroscope for each axis in space and measures just 4x4 millimeters. For our test setup, we used a module to hold the semiconductor. You can get it from AZ-Delivery for EUR4.79 [1]. If it is sold-out there, you can also purchase the module on Amazon or from Reichelt.

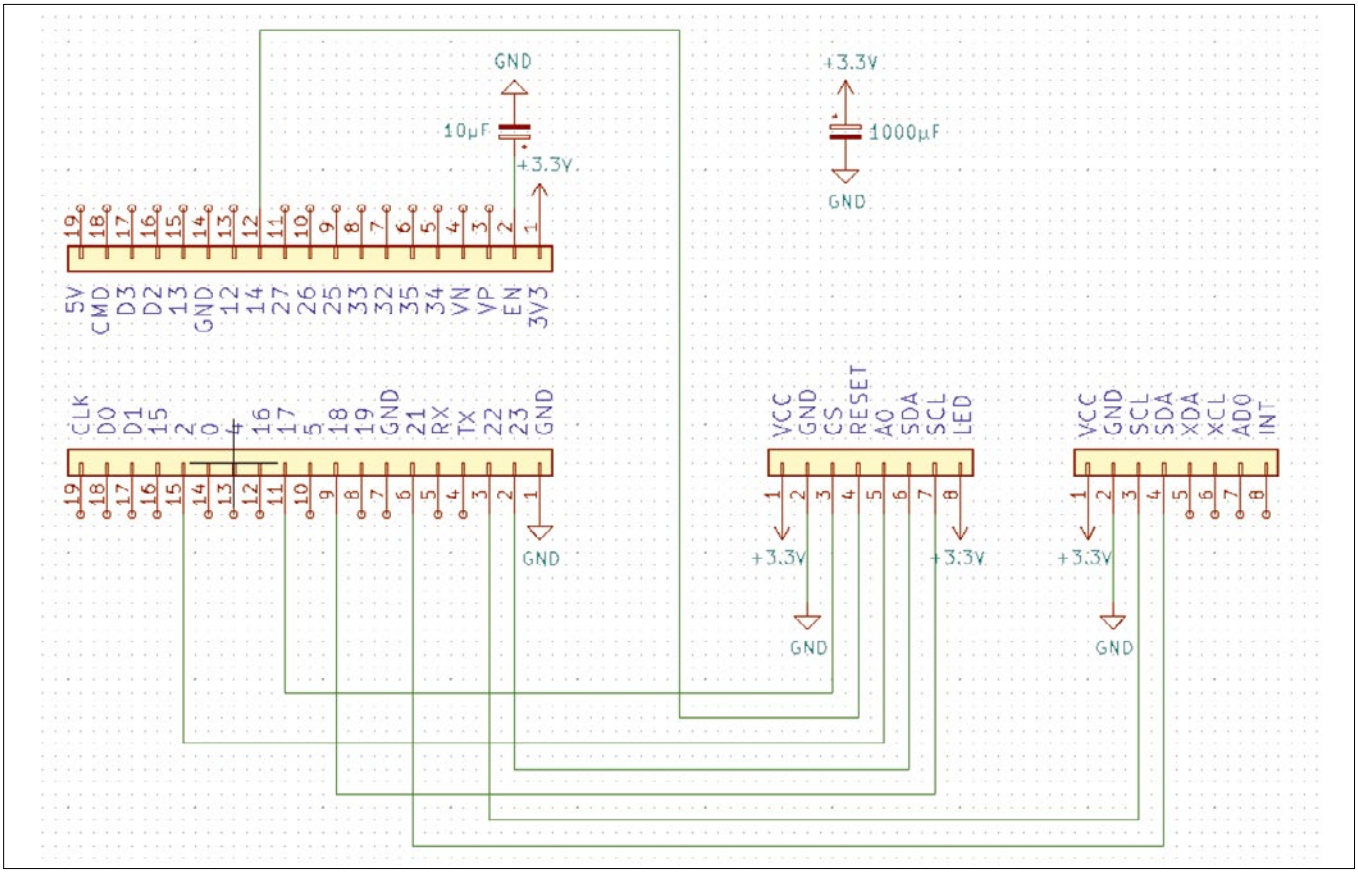


Figure 1: The circuit diagram for the test setup for our digital spirit level.

Lead Image © Uladzimir Kazhamiakin, 123RF.com

The MPU6050 communicates with the Raspberry Pi via the I2C bus. The *AD0* connector defines whether the sensor resides on bus address `0x68` (*AD0* to *GND*) or `0x69` (*AD0* to *VSS*). The operating voltage for the module is in the range of 3.3 to 5 volts. If you need more information about the MPU6050, take a look at the datasheet [2]. To access the sensor, we will use a library with simple functions for accessing the sensor's readings.

I went for an ESP32, a highly integrated microcontroller with an unbeatable price/performance ratio, to process the data from the sensor. I will be using an ESP32 development kit [3], which you can pick up fairly cheaply for EUR9.49. The Arduino IDE provides the development environment.

Last but not least, the project needs a display to show the measured values of the sensor graphically. An inexpensive 1.8-inch TFT color display for EUR7.99 [4], which can be addressed via a Serial Peripheral Interface (SPI), is a good choice here. Because controlling the display is quite tricky, we will be using the GFX library from Adafruit [5]. The library's source code is available on GitHub [6].

Test Setup

The circuit diagram for our spirit level (Figure 1) shows you how to connect all the components for the project. Let's take a closer look at the two capacitors at the very top of the schematic. The 10-µF capacitor between the *EN* pin and ground lets you to load the program into the ESP32 without pressing the boot button. Theoretically this should work without an additional capacitor, but practice shows that this is often not the case. The basic problem is the USB driver in combination with certain hardware components of the ESP32 development board.

The second capacitor (1000 µF) is an option. If the backlight of the display flickers, the capacitor will eliminate this issue. It stabilizes the operating voltage, preventing the flickering. If sporadic problems occur in other digital circuits that are difficult to isolate, it never hurts to build a large capacitor into the power supply. It often helps to eliminate problems of this type.

The remaining connections in the schematic connect the sensor to the ESP32 via I2C. The display uses the SPI and also comes with an additional reset connection. Be sure to connect all the power supplies present in the schematic. As Figure 2 shows, the experimental setup consists of two breadboards with the electronics. On each of the two boards, I removed one of the power supply strips to achieve a setup where the development board fits neatly, and you can easily access all the connections.

First Test

First of all, you need to start up the MPU6050 sensor. To do this, install the appropriate library in the Arduino IDE. In the menu, navigate to *Sketch* | *Integrate library* | *Manage library*. Once you get there, look for the *MPU6050_light* library and install it (Figure 3). If you are prompted to do so, install all the dependent libraries, too.

After the install you will find a sample program for reading the sensor values in the *File* | *Examples* | *MPU6050_light* | *GetAllData* menu (Figure 4). The software outputs the results of the measurements to the serial console. If you move the sensor, you can easily see how the individual values change. In addition to the data that the sensor inherently provides – acceleration in the direction of the spatial axes, angular acceleration about the axes, and temperature – it also outputs the angle of the axis as a computed value. We will be using this as the basis for our spirit level.

Testing the Display

The next step is to fire up the display. To run the display you need three libraries: TFT LCD library [7], Adafruit ST7735 library [8], and Adafruit GFX library [6]. Install all three in the Arduino IDE as described above. Again, you may be prompted for the libraries.

After the install, go to *File* | *Example* | *Adafruit ST7735 and ST7789 Library* |

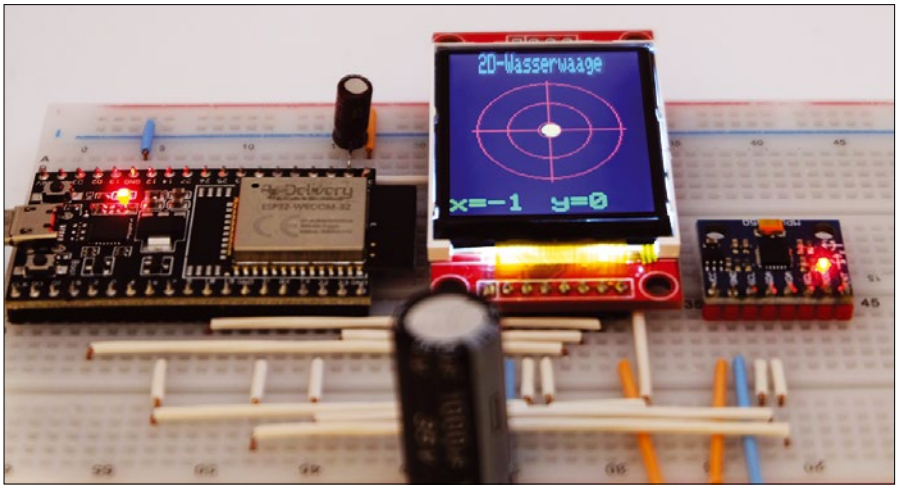


Figure 2: The complete test setup on the specially adapted breadboard.

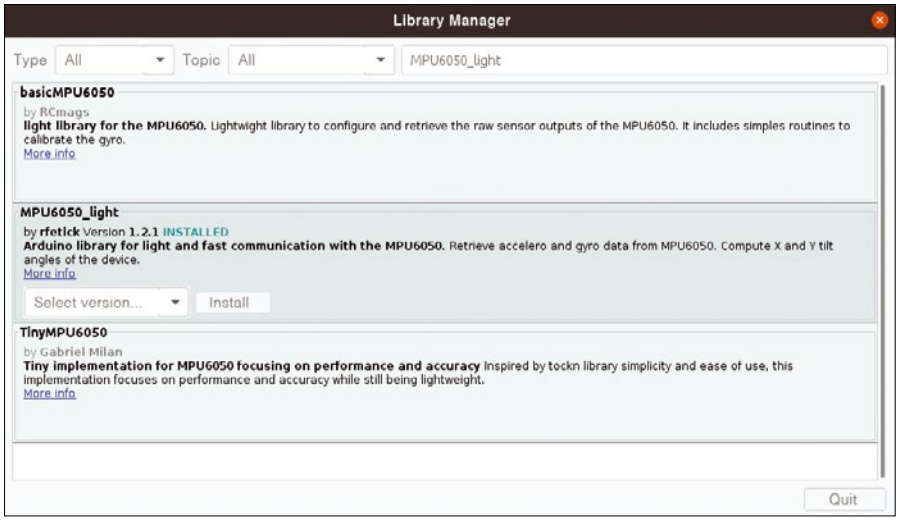


Figure 3: Finding and installing the MPU6050 library.

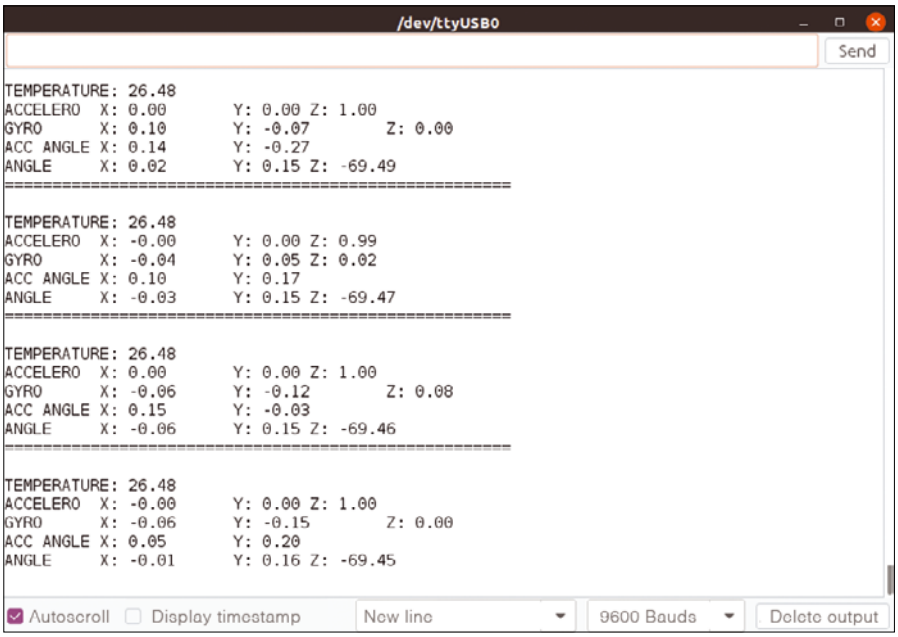


Figure 4: Typical output from the sample program.

Listing 1: Adaptations

```
// For 1.44" and 1.8" TFT with ST7735 use:
#define TFT_CS 17
#define TFT_RST 14
#define TFT_DC 2
Adafruit_ST7735 tft = Adafruit_ST7735(TFT_CS, TFT_DC, TFT_RST);
```

graphicstest to find a program to test the display. To adapt it to our setup, you need to assign different values to `TFT_CS`, `FT_RST`, and `TFT_DC`. The code needed to do this is shown in Listing 1. Simply copy this into the program at the appropriate place. After uploading it to ESP32, a sequence of test graphics will run across the display.

Spirit Level Program

After testing and performing a trial run with the components of our spirit level without any errors, it's time to take a closer look at the program (Listing 2) for the spirit level. You will find it, along with some other files, in the download section for this article [9]. The first block imports the required libraries (lines 1 to 5). To make the code more readable, the program then defines some constants (lines 6 to 8).

Lines 10 and 11 define the objects that we will use to address the display (`tft`) and the sensor (`mpu`). The `setup()` function starting in line 13 initializes all of the objects, with the `calcOffsets()` method to calibrate the sensor. After starting the program, it will take a while

Listing 2: spiritlevel.ino

```
01 #include "Wire.h"
02 #include <Adafruit_GFX.h>
03 #include <Adafruit_ST7735.h>
04 #include <SPI.h>
05 #include <MPU6050_light.h>
06 #define TFT_CS 17
07 #define TFT_RST 14
08 #define TFT_DC 2
09
10 Adafruit_ST7735 tft = Adafruit_ST7735
     (TFT_CS, TFT_DC, TFT_RST);
11 MPU6050 mpu(Wire);
12
13 void setup(void) {
14   Wire.begin();
15   mpu.begin();
16   mpu.calcOffsets();
17   tft.initR(INITR_BLACKTAB);
18   tft.fillScreen(ST77XX_BLACK);
19 }
20
21 int x, y, xold, yold;
22
23 void loop() {
24   mpu.update();
25   x=floor(mpu.getAngleX());
26   y=floor(mpu.getAngleY());
27   if ((x!=xold) or (y!=yold)) {
28     tft.setTextSize(1, 3);
29     tft.setCursor(20, 0);
30     tft.setTextColor(ST77XX_BLUE);
31     tft.print("2D-Wasserwaage");
32     tft.fillCircle(64-x, 80+y, 12, ST77XX_BLACK);
33     tft.fillCircle(64-x, 80+y, 5, ST77XX_YELLOW);
34     tft.drawCircle(64, 80, 7, ST77XX_RED);
35     tft.drawCircle(64, 80, 27, ST77XX_RED);
36     tft.drawCircle(64, 80, 47, ST77XX_RED);
37     tft.drawLine(64, 30, 64, 130, ST77XX_RED);
38     tft.drawLine(14, 80, 114, 80, ST77XX_RED);
39     tft.fillRect(0, 142, 160, 30, ST77XX_BLACK);
40     tft.setTextSize(2);
41     tft.setTextColor(ST77XX_GREEN);
42     tft.setCursor(0, 142);
43     tft.print("x=");
44     tft.println(x);
45     tft.setCursor(64, 142);
46     tft.print("y=");
47     tft.println(y);
48   }
49   xold=x;
50   yold=y;
51   delay(50);
52 }
```

for the values to settle. The variable definitions for the measured value of the X and Y axis then follow (line 21). The `xold` and `yold` variables are used to determine whether the values changed since the last measurement. If so, the program updates the contents of the display. This procedure reduces the volume of data sent to the display to the required minimum.

The `loop()` function calls the `mpu.update()` method to start a measurement (line 24). Then the `mpu.getAngleX()` and `mpu.getAngleY()` methods read the measurement values. The library returns floating-point values, which the `floor()` function converts to integers. Some sensor accuracy is lost in the process, but as a positive side effect, the values will not fluctuate as much. At this point, there are certainly many solutions that offer more accurate results. But for our spirit level, this simple procedure is perfectly OK. Just bear in mind that with more overhead you can tickle more accurate values out of the sensor.

The commands in the `if` statement starting in line 27 update the display. What is important to note here is that the program always drops new display information on top of the existing information. This means that you can't read older display information after some time because everything is filled up. Erasing the entire display before each write isn't a good idea either because it causes annoying flickering.

The solution is to selectively edit the areas that change. The command

```
tft.fillRect(0,142,160,30,ST77XX_↵
  BLACK)
```

in line 39 erases just the part where the measured values are shown. Parts that remain constant, such as the heading, are overwritten by the program. Apart from this, the code in this section is pretty much self-explanatory. If in doubt, consult the library documentation [5].

At the end of the `loop()` function starting in line 40, the program copies the values of the `x` and `y` variables to `xold` and `yold`. This is followed by a `delay()` to the flow before the program starts the next round of the loop. A YouTube video [10] shows you how the program works.

Conclusions

This article shows how to build a digital spirit level with simple materials. All in all, there are still many options for increasing the sensor's accuracy. A 3D printed housing for the technology would massively improve the stability of the setup.

The sample program, which is fairly simple, makes it easy to implement quite a bit more accuracy and functionality. The combination of ESP32 and the display allows for many more interesting projects to be built. All told, this is a neat project that you can learn a great deal from. I hope you makers out there enjoy building your personal spirit level. ■■■

Author

Martin Mohr experienced the complete development of modern computer technology live. After finishing university, he mainly focused on developing Java applications. The Rasp Pi rekindled his old love of electronics.

Info

[1] MPU6050 module: *https://www.az-delivery.de/en/products/gy-521-6-achsen-gyroskop-und-beschleunigungssensor*

[2] Data sheet for MPU6050: *https://invensense.tdk.com/wp-content/uploads/2015/02/MPU-6000-Datasheet1.pdf*

[3] NodeMCU ESP32 development kit: *https://www.az-delivery.de/en/products/esp-32-dev-kit-c-v4*

[4] 1.8 inch TFT color display: *https://www.az-delivery.de/en/products/1-8-zoll-spi-tft-display*

[5] Adafruit GFX library: *https://learn.adafruit.com/adafruit-gfx-graphics-library*

[6] Adafruit GFX library (Quellen): *https://github.com/adafruit/Adafruit-GFX-Library*

[7] TFT LCD library: *https://www.arduino.cc/en/Reference/TFTLibrary*

[8] Adafruit ST7735 library: *https://github.com/adafruit/Adafruit-ST7735-Library*

[9] Source code for this article: *ftp://ftp.linux-magazine.com/pub/listings/MakerSpace/*

[10] Spirit level in action [in German]: *https://youtu.be/ib4KpretMa8:*

MakerSpace

Change internal logic from relays to an Arduino

Redo

An electronic project at a local science center was showing its age, calling for a refresh: in this case, rebuilding it almost from scratch with an Arduino instead of relays. *By Scott Sumner*

A museum exhibit called Buzzwire looks like outlines of two oversized hands giving you a high five (Figure 1). Each hand has a metal handle with a loop, and your goal is to move the loops up and down the hand without touching it. For an even bigger challenge, you can try to do both hands at once in the same or opposite directions. If either loop touches the hands, a buzzer and light come on and the handle vibrates.

The original circuitry for the hands comprised interconnecting timer relays to switch the assorted components. The design had no microcontrollers or anything smarter than a switch, which had several drawbacks – but the main one was that if the puzzle was abandoned mid-run, the light, buzzer, and vibration motor would run continuously until the handles were removed.

Figure 1: The Buzzwire exhibit (after the rebuild).

Physical Rebuild

While I rebuilt the control electronics, I sent the case out to be rebuilt as well. The original design was made out of plywood and had started to accumulate a lot of dings and scratches. The case rebuild was very helpful because it showed areas that needed to be reinforced. Ultimately, I had the entire front panel replaced with Delrin, a very hard plastic that will withstand wear and tear better than plywood. It also brings a fresh look for anyone that has seen it before.

The physical rebuild also incorporated larger vibration motors in the handles. The old version used pager motors, which were more heard than felt. The machine shop I worked with fabricated new handles from scratch that incorporated larger motors, making the buzz much more noticeable.

Table 1: Arduino Connections

Pin	Connection
2	Left handle input
3	Right handle input
8	Relay 1 control signal
9	Relay 2 control signal
10	Relay 3 control signal
11	Relay 4 control signal
GND	Left hand (puzzle)
GND	Right hand (puzzle)
GND	Relay board ground
5V	Relay board power

Because one of the components had burned out on more than one occasion, the decision was made to rework the exhibit with upgraded technology that addresses some of its problems.

New Tech

To design the new controller, I started with an Arduino Mega because I had one on-site. An Arduino Uno would have worked just as well, but in this case, the Mega is a standard part, is easy to trade out, and is kept in stock. (See the "Physical Rebuild" box.)

I started by assigning pins on the Arduino. The handles each need an input pin, so I started with those. Other than timers the handles are the only inputs to the circuit design. For outputs, I wanted several things to operate with different timing, so I set up four pins to control relays, which will handle both the assorted voltages and the ability to switch things on and off at different times (Table 1; Figure 2).

Finally, for ease of connections into the exhibit itself, I assembled everything on a piece of plywood and brought all of the connections to a pair of barrier strips (Table 2). Each hand got its own barrier strip, and the two are identical except for power input on connections 11 and 12.

Table 2: Barrier Strip Connections

Connection No.	Left Barrier Strip	Right Barrier Strip
1	Hand (sculpture)	Hand (sculpture)
2	Handle	Handle
3	Ground (1)	Ground (1)
4	Ground (1)	Ground (1)
5	Ground (1)	Ground (1)
6	12V (switched) (2)	12V (switched) (2)
7	12V (switched) (2)	12V (switched) (2)
8	12V (switched) (2)	12V (switched) (2)
9	Ground	Ground
10	5V (switched)	5V (switched)
11	Ground	Ground
12	12V power in	5V power in

(1) Interchangeable ground terminals. Choose one.

(2) Interchangeable 12V terminals. Choose one.

Logic Flow

Now that all of the hardware was assembled and simple test programs tested the activation of the relays, which in turn vibrated the handle or turned lights and buzzers on and off, I needed to figure out the flow of the control program.

One of the things that is often overlooked in a project is documentation. So as not to fall into that trap, I made copious notes and typed them up into a documentation package. One of the things I included was a flowchart to describe the logic (Figure 3; see the "Flowcharts" box).

Walking through the flowchart, the first decision is the diamond right below Start asking whether the ring is touching the puzzle. If not (exit to the right), you wait 250ms, turn off the relays, and return to the original decision. It should be noted that *Turn relays off* will happen even if they are already off.

If the ring *is* touching the puzzle (exit the decision through the bottom), then turn the relays on. The next decision diamond checks whether the

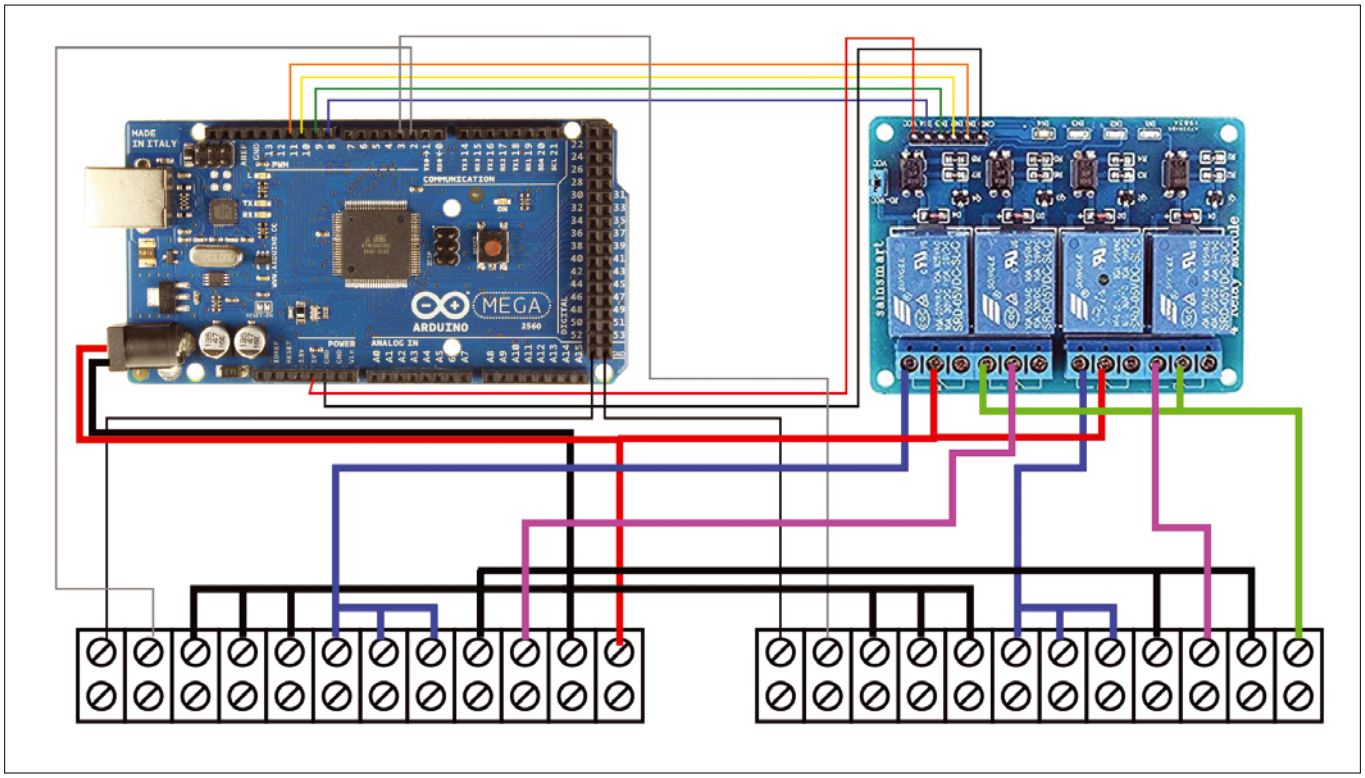


Figure 2: The wiring diagram includes relay and barrier strip connections. All connections were brought out to a barrier strip for easier servicing.

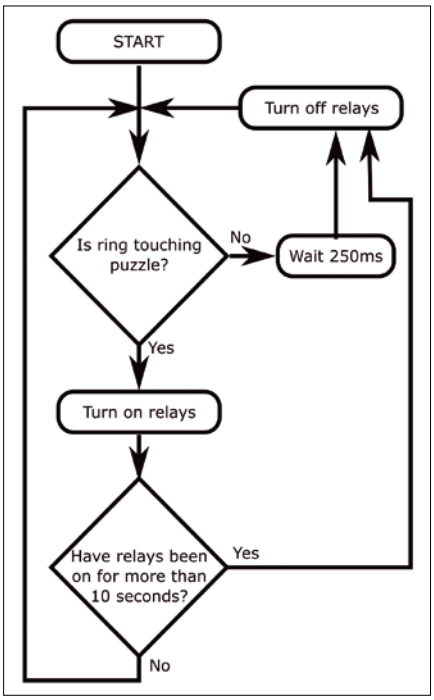


Figure 3: A flowchart is a handy way to visualize what you want your code to do – or show what your code does, once it's written. Note that some functions are called multiple times, even if the task the function accomplishes is already in the tested state.

relays have been on for more than 10 seconds. If so (exit right), you turn off the relays and return to the first decision. If not (exit bottom), just return to the first decision.

That process will loop infinitely until power is removed from the Arduino. You should also note that this flowchart shows a simplified process. I don't specify assigning variables or duplicating the process for both hands, just the theory of operation about what's happening inside the logic. In this way, the program flow is easy for even a non-programmer to follow.

Code

Arduino code is made up of two sections. The `setup` section runs once when the Arduino powers up, and the `loop` section then runs infinitely until power is removed. As their names imply, they are designed to initialize everything and then run continuously.

The setup() Function

In Listing 1, lines 1-17 comprise the Arduino `setup` function. In this section, you tell the Arduino what you want each pin to do, set up and turn on any special functions, and define initial variables (although that's not needed in this particular code).

The `pinMode` functions (lines 3-9) set the Arduino input and output status. The first argument is the pin number, and the second argument is the mode. I use `INPUT_PULLUP` for pins 2 and 3, which are where the wires from the handles come in. A `PULLUP` mode tells the Arduino to turn on the pull-up resistor of a GPIO pin.

A pull-up resistor is the electronic equivalent of a default value. It uses a weak resistance to force the pin to V+. An incoming signal can literally overpower the resistor and pull it to ground. Without the pull-up resistor, electrical noise can easily make the circuit operate sporadically. Because in this case the wiring would be connected to nothing when not touching the puzzle, the pull-up setting provides a solid change of state.

Arduino pins 8-11 (lines 6-9) are set to `OUTPUT`. In this mode the pin will be either 5V or ground depending on its state. The `digitialWrite` functions in lines 11-14 set the initial states of the relays.

All states are set to `HIGH` to turn the relays off. Most relay modules you order online (I got mine from Amazon) operate in an inverted state. That is, a ground signal will turn them on and anything above a threshold level (usually about 1.2V) will turn them off. This setup makes them compatible with either 3.3V or 5V microcontrollers. Because the decision is either grounded or not grounded, it doesn't matter how high the voltage goes over the threshold, as long as it is within the limits of the board.

The last line of the `setup` section calls `Serial.begin`, which turns on the built-in

Flowcharts

Flowcharts are an excellent way to visualize flow through a system, program, or other process. In the early days of computers when programs were built on punched cards and you "programmed" on paper, they were an essential tool to understand flow. Although still useful for design, they aren't at the forefront, as they once were. Flowcharts are seeing a resurgence, though, in online forms where data moving through a system is now visualized in the flowchart style that once originally designed it.

Table 3: Program Variables

Line No.	Type	Name	Description
21	Integer	`iInput2`	Stores `digitalRead` value from first handle
22	Integer	`iInput3`	Stores `digitalRead` value from second handle
23	Integer	`iOut1State`	Stores whether indicators should be on for first handle
24	Integer	`iOut2State`	Stores whether indicators should be on for second handle
25	Integer	`iMotorReset1`	Tracks whether motor 1 has been turned off because the handle has been in contact with the puzzle too long
26	Integer	`iMotorReset2`	Tracks whether motor 2 has been turned off because the handle has been in contact with the puzzle too long
27	Unsigned long	`ulReset1`	Time in `millis` to turn off the first motor in normal operation
28	Unsigned long	`ulReset2`	Time in `millis` to turn off the second motor in normal operation
29	Unsigned long	`ulMotor1`	Time in `millis` to turn off the first motor because its been in contact with the puzzle too long
30	Unsigned long	`ulMotor2`	Time in `millis` to turn off the second motor because its been in contact with the puzzle too long

Listing 1: Buzzwire

```
01 void setup() {
02   // put your setup code here, to run once:
03   pinMode ( 2 , INPUT_PULLUP );
04   pinMode ( 3 , INPUT_PULLUP );
05
06   pinMode ( 8 , OUTPUT );
07   pinMode ( 9 , OUTPUT );
08   pinMode ( 10 , OUTPUT );
09   pinMode ( 11 , OUTPUT );
10
11   digitalWrite ( 8, HIGH );
12   digitalWrite ( 9, HIGH );
13   digitalWrite ( 10, HIGH );
14   digitalWrite ( 11, HIGH );
15
16   Serial.begin(19200);
17 }
18
19 void loop() {
20   // put your main code here, to run repeatedly:
21   int iInput2 = 0;          // Wand 1
22   int iInput3 = 0;          // Wand 2
23   int iOut1State = 0;       // General output for Hand 1
24   int iOut2State = 0;       // General output for Hand 2
25   int iMotorReset1 = 0;     // Tracks if motor 1 has been
                                 shut off due to long
                                 contact
26   int iMotorReset2 = 0;  // Tracks if motor 2 has been
                               shut off due to long contact
27   unsigned long ulReset1 = 0;  // Time in millis to turn
                                    off the Hand 1 buzzer
                                    and light
28   unsigned long ulReset2 = 0;  // Time in millis to tunr
                                    off the hand 2 buzzer
                                    and light
29   unsigned long ulMotor1 = 0;  // Time in millis to turn
                                    off the hand 1 motor
                                    for long contact
30   unsigned long ulMotor2 = 0;  // Time in millis to turn
                                    off the hand 2 motor
                                    for long contact
31
32   while ( 1 )
33   {
34     iInput2 = digitalRead ( 2 );  // Are we touching the
                                       hand 1 puzzle?
35     iInput3 = digitalRead ( 3 );  // Are we touching the
                                       hand 2 puzzle?
36
37     if ( iInput2 == LOW )         // If we're touching
                                       the hand 1 puzzle...
38     {
39       ulReset1 = millis() + 500; // Set a timer for 1
                                      second from now
40       if ( ulMotor1 == 0 ) ulMotor1 = millis() + 10000;
           // Set a timeout for 10 seconds from now
41       if ( iMotorReset1 == 0 ) {
42         Serial.println ( "Motor On" );
43         digitalWrite ( 8 , LOW );  // Turn on the motor
                                        if it hasn't been
                                        disabled
44         digitalWrite ( 9 , LOW );   // Turn on the light
45       }
46
47     }
48     else    // We're not touching the puzzle
49     {
50       iMotorReset1 = 0;   // Clear the timeout flag
51       //ulMotor1 = 0;       // Clear the timeout timer
52       Serial.println ( "Motor Off" );
53     }
54
55     if ( iInput3 == LOW )   // Same as above but hand 2
56     {
57       ulReset2 = millis() + 500;
58       if ( ulMotor2 == 0 ) ulMotor2 = millis() + 10000;
59       if ( iMotorReset2 == 0 ) {
60         digitalWrite ( 10 , LOW );
61         digitalWrite ( 11 , LOW );
62       }
63     }
64     else
65     {
66       iMotorReset2 = 0;
67       //ulMotor2 = 0;
68     }
69
70     if ( millis() > ulMotor1 && ulMotor1 != 0 )   // If
          current time (millis) is greater than the motor
          timeout AND we're watching for a timeout...
71     {
72       digitalWrite ( 8 , HIGH );    // Turn off the motor
73       digitalWrite ( 9 , HIGH );
74       iMotorReset1 = 1;       // Set the reset flag so we
                                    don't turn it on again
75       Serial.println ( "Motor Timeout" );
76     }
77
78     if ( millis() > ulMotor2 && ulMotor2 != 0 )
79     {
80       digitalWrite ( 10 , HIGH );
81       digitalWrite ( 11 , HIGH );
82       iMotorReset2 = 1;
83     }
84
85     if ( millis() > ulReset1 )    // Is it time to reset
                                       the buzzer / light?
86     {
87       digitalWrite ( 8 , HIGH );  // If so turn them off
88       digitalWrite ( 9 , HIGH );
89       ulMotor1 = 0;
90     }
91
92     if ( millis() > ulReset2 )
93     {
94       digitalWrite ( 10 , HIGH );
95       digitalWrite ( 11 , HIGH );
96       ulMotor2 = 0;
97     }
98   }
99 }
```

USB serial port on the Arduino. In this case, I'm only using it to send debug messages to the Arduino console, but for bigger projects you could also use it to communicate with a companion computer program.

The loop() Function

The rest of the program is the `loop` section. The first thing I do here is define a number of variables, as shown in Table 3.

The `loop` function cycles infinitely, but I don't want to redefine all of my variables on every pass, so after I finish declaring my variables at the top of the section, I start another infinite (`while`) loop in line 32 that runs the rest of the code.

The `digitalRead` functions in lines 34 and 35 determine whether the handle is touching the puzzle. This state is stored in `iInput2` and `iInput3`.

Touching the Puzzle

If the handle is touching the puzzle (i.e., the value of `iInput2` in line 37 is `LOW`), lines 38-47 whip into action. Line 39 uses `millis` (see the "Electrical Noise" box) and adds 500 to get a time half a second in the future. This value is stored in `ulReset1`. Line 40 does the same thing for `ulMotor1` if it is currently zero (not set). It also sets its time for 10 seconds (10,000ms) in the future instead of half a second.

The `if` in line 41 checks to see whether the motor has already been shut off because the handle has been touching the puzzle for too long. If not, lines 43 and 44 use `digitalWrite` to turn on the relays and, by extension, the motor, light, and buzzer.

Not Touching the Puzzle

The `else` function block (lines 48-53) starts the counterpoint to the `if` on line 37. In this case, the handles are not touching the puzzle, so I clear `iMotorReset1` by setting it to `0` and perform a debug print to show that the motor has been turned off.

The Long Timeout

The `if` block in lines 70-76 checks two conditions: Is `millis` (current time) greater than `ulMotor1`, the long timeout value, and is a timeout timer active (`ulMotor1` is not equal to `0`)? When the handle touches the puzzle, the delay is set for 10 seconds. If the handle remains in contact with the puzzle for that long, the motors are shut off.

The `digitalWrite` functions in lines 72 and 73 turn the motors off by setting the output pin to `HIGH` and turning off the relay. Line 74 sets the `iMotorReset1` flag, and the final function sends the debug message *Motor Timeout* to indicate that the handle was left on the puzzle too long.

The Shorter Timeout

The simpler `if` block (lines 85-90) checks whether the short timer has been exceeded. Once the handle comes off the puzzle, it continues to buzz for half a second before you can continue your attempt. The `digitalWrite` functions in lines 87 and 88 do the work of turning the relays (and thus the motors) off; then, `ulMotor1` is reset to `0` so that the buzzer can be triggered again the next time the circuit is triggered.

Conclusion

Any time you work on a project that was originally designed by someone else, you have an opportunity to learn. As you explore the existing circuitry and decide what you can reuse and what you need to replace, you discover the thought processes and electronic procedures that the previous designer incorporated into their work. This process not only gives you ideas for your next project, but also a road map for the current rebuild. Of course, if you can just talk to the original designer, the project will be much easier. As you know, that's not always the case.

While working on this project, I incorporated a few new features, like the long timeout for when the handle is left touching the puzzle halfway through. The hope is that this change will make the handles last longer and the motors won't burn out as quickly. I came up with a few other ideas, as well, but decided to hold those for yet another future iteration of this exhibit, which has been reinstalled on the museum floor, with guests once again enjoying getting Buzzwired. ■■■

Electrical Noise

One of the things I discovered when reworking the exhibit was that the motors that provided the handle vibration caused more electrical noise than expected. The Arduino was sensitive enough to pick up this noise as a touch, whereas the original relays did not. This new design challenge had to be overcome, and it was solved by adding a filter capacitor to each motor.

Filter capacitors can remove electrical noise that leaks back into your circuit. I described them to a coworker like this: Imagine you're at the local amusement park and you've just gotten some tasty snacks while you wait in a long line for the next roller coaster. Once you eat everything you still have all of your trash. If there's a trash can at the front of the line, you can deposit it there. Otherwise you're trying to hold onto all of your trash while you ride the roller coaster and it flies all over the park.

Filter capacitors are like the trash cans at the front of the line. They collect the extra unwanted bits while the stuff you want (electricity) goes through. Digital electronics are much more sensitive than relay coils, and this noise problem didn't exist in the original project, so the original designer didn't have to consider it.

Author

Scott Sumner has worked in the museum and nonprofit industry for most of his professional career. He enjoys exploring technology solutions with Arduinos, Raspberry Pis, microcontrollers, and Linux systems.

Search tools have been around since the beginning of computers. In fact, the power to search for stuff is one of the reasons computers were developed in the first place. But despite this venerable connection between computers and search, search methods remained the same for decades – at least, until the arrival of fuzzy search techniques.

In the old days, you had to type a term exactly right to search on it – mispellings and typos turned up nothing at all. If you guessed the title of a movie or book and were off by one character, you were nowhere. Modern fuzzy search tools offer a way to the answer even if you didn't get the question right. This month we look at the search tools Fzf and Fzy.

Elsewhere in this month's Linux Voice, we take a look at the Dim media manager and show you how to create graphs and visualize data with LibreOffice Chart.

Image © Olexandr Moroz, 123RF.com

LINUXVOICE

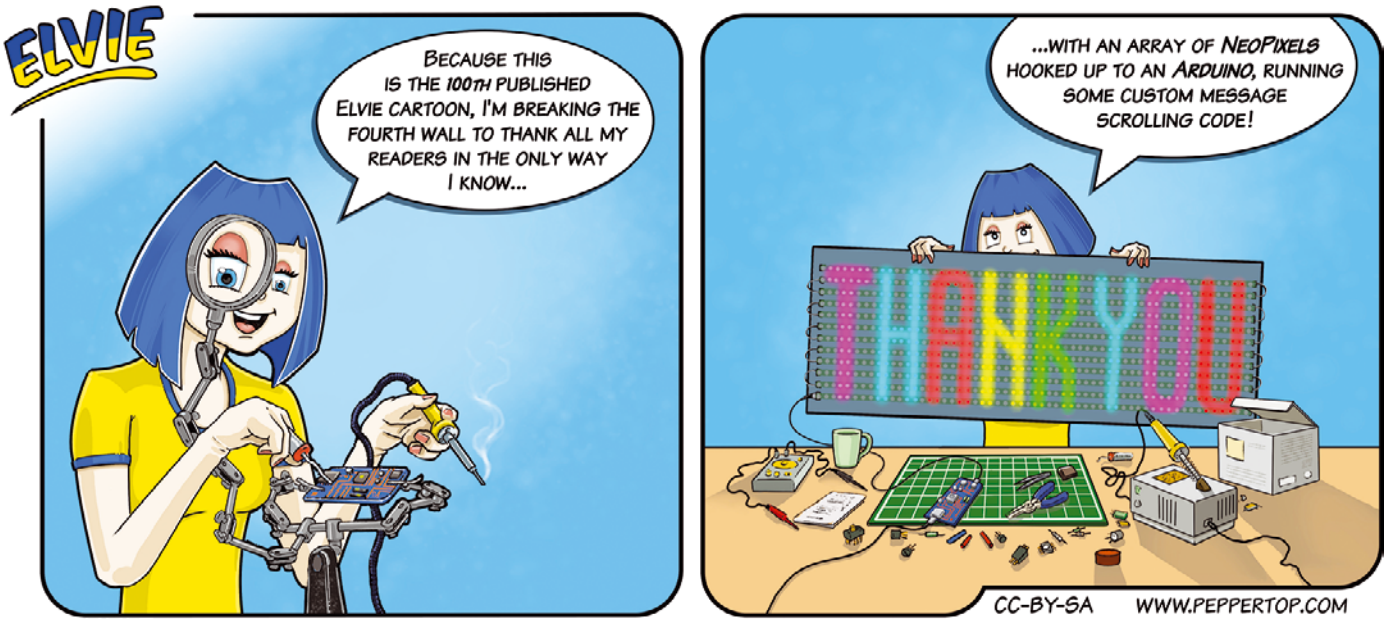
Doghouse – Local Internet Support 71
Jon "maddog" Hall
Tools such as community servers and local web traffic caches can help improve community Internet service.

Fzf/Fzy 72
Karsten Günther
Fuzzy finders retrieve useful results from data streams even if there are no exact matches.

Dim Media Manager 78
Ferdinand Thommes
Dim, a relatively new open source media manager, looks to implement the appearance and feature set of the commercial Plex media center.

FOSSPicks 84
Graham Morrison
This month Graham looks at TerraForge3D, Navidrome, qddcswitch, Difftastic, fheroes2, and more!

Tutorial – LibreOffice Charts 90
Marco Fioretti
Everybody needs charts sooner or later, and LibreOffice Calc is the easiest way to create them with free and open source software.



Hone your skills with special editions!

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

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

Available in print or digital format

Check out the full library!

shop.linuxnewmedia.com

MADDOG'S DOGHOUSE

Tools such as community servers and local web traffic caches can help improve community Internet service. BY JON "MADDOG" HALL

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®.

Local tools for better service

I recently traveled to Cuba to speak at Informática 2022, an international technical conference held in Havana with speakers and participants from all over the world. It allowed me to see firsthand a little of what Cuba is like today.

For those of you who are not familiar with the country, it is an island nation about 100 miles south of Florida, with a population of approximately 11 million people. It has been under a trade embargo from the United States for the past 60 years.

Because this is a technical magazine, I will talk about technical and related economic issues, not political issues.

From a computer standpoint, Cuba has modern desktop and laptop computer brands that people around the world would recognize. They purchase them from countries other than the US and typically use the same operating systems that most other people use. I did notice a lack of Apple-based systems, and I suppose this is because Apple is a US-based company and does not sell their products inside Cuba.

Cuba's government has decided to support FOSSHC as a strategy. They have developed their own distribution, NOVA [1], which will be used extensively throughout government and commerce.

The country has Internet service in most places, both WiFi and cellular. My cellular phone, which works in most countries I visit because I use Google Fi (which is not available in Cuba), did not work in Cuba unless I was inside my hotel and could easily use WiFi to make calls. I could have purchased a SIM card at the airport, but I determined that WiFi VoIP was good enough for the one week I would be in Cuba.

My hotel's WiFi, while adequate for most uses, was both expensive and based on connection time. To save money, I would disconnect from the WiFi when I was not actively using it, which also meant I could not receive notifications, and reconnecting every time I wanted to use the WiFi was clumsy. Periodically, I would forget to disconnect and later would find I needed to "recharge" my account.

Some of the issues with WiFi and cellular Internet might be attributable to fact that the sole Internet company on the island is owned by the state, and there is no competition. I am not sure what would be necessary to license and start another Internet company on the island, but for a population of only 11 million, splitting the population across multiple Internet providers might not be cost effective. Perhaps having the state collaborate more with the FOSSHC community would generate better Internet.

For example, charging by connection time is a strategy that helped US telephone companies – that used copper wire to connect to the central office (CO) – manage the connections to the crossbar switches of the previous century. Under that system, each time one person was connected, someone else could not be connected, so when dial-up Internet was the standard, charging by connection time was practical.

However, with modern-day digital connections and switching, it is better to eliminate connection charges and charge by data usage with data caps like most Internet companies do today. Perhaps this connection-charge method of billing was only something done by my hotel, but it certainly made using the Internet more difficult and less useful.

Another issue seemed to be that rural areas did not have as good a connection as urban areas. This – in Cuba and elsewhere – could be alleviated significantly by using projects such as Internet-in-a-Box [2], which helps by caching local data or data which is requested over and over again by local people.

Years ago private branch exchanges (PBX) were installed in hotels, companies, and other institutions because people realized that most of the telephone calls in the building were local to the building themselves. Instead of needing 500 sets of wires running from each phone to the CO of the telephone company, you would run these wires to a PBX in the basement of the building and then have some small percentage of wires running from the PBX to the CO. It would be very, very seldom that all of the shared wires from the PBX to the CO would be used at the same time.

The telephone book of the past century has been replaced by the web pages of today, and all you have to do now is reach those web pages, which – with a local approach similar to the idea behind PBXs – can be cached on your local Internet-in-a-Box.

For communities with similar challenges, setting up a community server with community networking using code such as that of FreedomBox [3] would allow a reduction of data traffic and better service response inside the community. An inexpensive single-board computer with SSD could provide a reliable, secure data cache unimaginable just a few years ago. ■■■

Info

[1] Nova: *https://en.wikipedia.org/wiki/Nova_(operating_system)#:~:text=The%20first%20version%20of%20Nova,Edition%20was%20based%20on%20Ubuntu*

[2] Internet-in-a-Box: *https://internet-in-a-box.org*

[3] FreedomBox: *https://en.wikibooks.org/wiki/FreedomBox_for_Communities*

Use fzf and fzy to add fuzzy search tools to the shell

Near Miss

Fuzzy finders retrieve useful results from data streams even if there are no exact matches. BY KARSTEN GÜNTHER

Today, fuzzy searches are an integral part of everyday IT life. They correct typos, detect similarities, and offer a way to find what you need with reasonable overhead, even in unstructured data. The basic principle is based on the Levenshtein distance [1], word distances defined in the 1960s (see the "Levenshtein" box). Developers have extended and optimized this algorithm more or less from the start.

Having said this, many standard tools ignore fuzzy searches or limit themselves to highly simplified variants up to this day. For example, the grep tool from the *coreutils* package processes arbitrarily complex regular expressions (patterns), but it does not support fuzzy searching. Fuzzy searching is supported by agrep [2] and ugrep [3] at the command line, and there are several other, less well-known tools. Agrep impresses here with a best-match option.

Fzf [4] and fzy [5] (see the "Little Brother" box) enrich the shell construction kit, adding two new, powerful tools. They act as interactive filters, i.e., by default, they source their data from the standard input and return the results to the standard output. Interactive in this context means that they provide interfaces for entering the search patterns, which allow the search patterns to be adapted and refined at runtime.

Levenshtein

The Levenshtein distance describes the minimum number of changes (replace, delete, insert) needed to transform one string into another. The weighted number of these actions determines how similar the two strings are. Weighting means that, for example, two swapped letters (like in a typo) are less important than, say, inserting additional characters. But the approach also involves a number of problems. Weighting depends on the concrete task (spell check, search, etc.) and the language.

fzf

Fzf stands for Fuzzy Finder. The tool acts as a general-purpose filter, which also acts as a menu generator in interactive mode. Calling `fzf` launches the tool in the shell. This normally just displays all the files in the current directory. Fzf uses a built-in `find` command to do this, which is the default if the program is not given any data via the standard input. This feature helps to keep the commands short. You can manage this behavior via the `FZF_DEFAULT_COMMAND` environment variable, specifying a different command if needed. You can store the options that you want to pass in to fzf as defaults in the `FZF_DEFAULT_OPTS` environment variable.

Most of the time, the program is used in conjunction with a pipe. This means that it receives its input data via the standard input (Listing 1).

Little Brother

Fzf is already in use as a standard tool on many systems today. According to its developer, the enhanced fzy tool offers an improved algorithm and a faster search speed. In addition, fzy limits itself to the bare essentials in terms of options, and this simplifies the application. At first glance, fzy really only looks like a variant of fzf, but this is deceptive. Although the output is similar, fzy can at most be considered fzf's little brother. It is also piped, but there is also a very simplified Finder, which gives users a kind of menu with a configurable prompt. But this already exhausts the list of similarities. This becomes particularly clear if you look at fzy's options. By default, its Finder uses only 10 lines of the terminal to display the list of results. This can be changed with the `-l` option. Fzy can only work with positive patterns; it does not support exclusions. In interactive mode, only a handful of keyboard shortcuts are available. They are listed by the tool's fairly concise man page.

Listing 1: locate and fzf via Pipe

```
$ locate / | fzf
[...]
.../Images-sda5/_1130192.png.out.pp3
.../Images-sda5/_1130192-1.png.out.pp3
.../Images-sda5/_1110520.jpg.out.pp3
.../Images-sda5/_1090399.dng.xmp
.../Images-sda5/_1090399.dng.pp3
.../Images-sda5/_1040511a.pts
.../Images-sda5/_1040511.pts
.../Images-sda5/_1040511-6.pts
.../Images-sda5/DANCE!-2020-2021
[...]
```

Listing 2: Combining Search Patterns

```
PATTERN1$ PATTERN2 ^PATTERN3 !PATTERN4
```

This takes you to interactive mode, which consists of two parts. The visible interface, the Finder, supports interactions: You can move around with the arrow keys or by pressing Ctrl+K and Ctrl+J. The terminal displays the current line in bold or inverse type depending on the settings. Pressing the Enter key accepts the selected line and sends it to the standard output or to the terminal.

In addition, a prompt (>) is displayed at the bottom edge of the Finder, prompting you for input. If you already passed a search pattern into fzf when you called it (as an argument of the `-q` option), the search pattern can be edited in this line. The second part works in the background, managing the index and the fuzzy search. In the second-to-last line in Figure 1, the tool shows you what is happening in this regard. A constantly changing symbol shows that it is creating an index from the input, which it will then search through for the pattern. The current size of this index is also shown at the bottom.

Search Pattern

The Finder is an essential feature of fzf. However, the tools's capabilities are not limited to editing existing search patterns. It also evaluates the patterns incrementally. Each additional character entered refines the search. In Extended Search mode, which is enabled by default, you can also enter additional patterns separated by spaces. The program assigns special meanings to some of the special characters used in these patterns (see Table 1). You can combine several search patterns as described in Listing 2.

You can use the arrow keys to navigate within the search pattern in the Finder. The highlighted line shows the selection. Alternatively, some default key mappings are available (see Table 2). All actions involving selecting or deselecting of multiple lines require you to launch the tool in multiple-selection mode with the `-m` option. On top of this, the Finder also supports mouse control. Scrolling with the mouse wheel, clicking, and double-clicking lets you enable and select. If you hold down the Shift key, a mouse click selects multiple items. A double-click ends the input.

Otherwise the program works with key bindings adapted to Emacs by default. Additional or deviating key bindings can be set with the `--bind` option. The `-e` command-line option disables the fuzzy search, returning only exact hits. Other important fzf options are summarized in Table 3.

You can define how the program determines and evaluates matches with a number of parameters. The input lines are evaluated on the basis of the criteria from Table 4. `index` can only occur at the end. Fzf uses this option by default if you do not specify anything. Fzf evaluates the inputs using these criteria, which you explicitly select as arguments for

Table 1: Searches for Power Users

Character	Example	Result
^	^linux	Lines that start with linux
$	linux$	Lines that end with linux
!	!linux	Lines that do not contain linux
'	'linux	Lines that contain precisely linux (not a fuzzy search)
sbtrkt	–	Token for fuzzy search (default)

Table 2: Controls

Key	Explanation
Ctrl+K	Move up
Ctrl+J	Move down
Ctrl+C	Cancel
Alt+F	Jump one word to the right in the search pattern
Alt+B	Jump one word to the left in the search pattern
Ctrl+A	Jump to the start of the search pattern
Ctrl+E	Jump to the end of the search pattern
Tab	Select multiple consecutive lines
Shift+Tab	Deselect the following selected lines successively
Ctrl+I	Select or deselect arbitrary lines

Table 3: Important Options

Option	Explanations
`-e/+x`	Display only exact matches, disable fuzzy search
`+i/-i`	Case (in)sensitive
`--algo=v1/v2`	Select algorithm: `v1` is faster; `v2` has better matches (default)
`--disabled`	Disable search, only provide interface
`+s/--no-sort`	Do not sort output
`--tac`	Reverse order of input
User Interface Options	
`+m/-m`	Enable/disable multiple selection; default, return only one match
`--cycle`	Continue search before the first/after the last entry
`--height=PERCENT`	Height of Finder as a percentage of the terminal size
`-q QUERY`	Start software with `QUERY`
`-1`	Do not start Finder for a single match
`-0`	Do not start Finder if there is no match
`--preview COMMAND`	Use `COMMAND` for the output
`--preview-window`	Define form, size, and type of preview

the `--tiebreak=CRITERION` option. By default, it uses `length` or `length,index`. Stipulating `end` tells the software to evaluate the lines in reverse order.

One typical fzf application is a fault-tolerant variant of the `locate` command. The latter searches for files on the local system by reference to the filenames in a previously created database. This search is non-fuzzy; it does not return results in case of typos or abbreviations. In the example in Listing 3, `locate /` creates a list of all entries and returns it to fzf via a pipe. This stipulates all the arguments passed in as patterns for the command. If you pass in `L` for no arguments, the entire list appears, and you can enter patterns directly in fzf's interactive mode.

If you have a very large database, you will probably want to exclude many matches to improve the clarity of the results. You can do this by passing in arguments to the function. In Listing 4, the pattern searches for `something` but excludes `butnotthat`. If the search still returns too many matches, such as in randomly-named browser caches, more additional patterns in interactive mode will lead to better results. At the command line, you must quote the exclamation mark in front of the excluding pattern with a backslash (`/!`) or use quotes around longer sequences. Otherwise, the shell will interpret the character as a `history` command and remove it. In interactive mode, `\!` has a different meaning.

Fzf supports a sophisticated preview mode for the matches. The argument in the `--preview` option is the command line for generating the preview. This is shown for the file currently selected with the cursor in a separate window that Fzf generates on the right (Figure 1). You can only use text output in a terminal, which means that, for many files, the metadata or the previously converted or extracted text contents appear as a preview instead of the binary data.

`less` is really useful as a universal previewer (Listing 5). After all, the pager is particularly well suited for displaying different file types with lesspipe [6].

Table 4: Modifiers

Option	Explanation
`length`	Prefer shorter lines with matches
`begin`	Prefer matches at the start of the line
`end`	Prefer matches at the end of the line
`index`	Prefer lines nearer to the start of the input stream

Listing 3: Fuzzy Search with locate

```
L () { locate / | fzf -q "${*}" ; }
```

Listing 4: Excluding Matches

```
> L something \!butnotthat
[...]
.../Music/The Temptations -  Papa was a rolling stone.mp3.mp3
.../Images/png/2-watermark.png
.../Music/Frumpy - How the Gypsy was Born.mp3
.../Music/2/Frickle - Some People, Follow Me.mp3
.../Music/Byron Metcalf - Heart Warriors+.mp3
.../Images/aerial image no channel water border.png
[...]
```

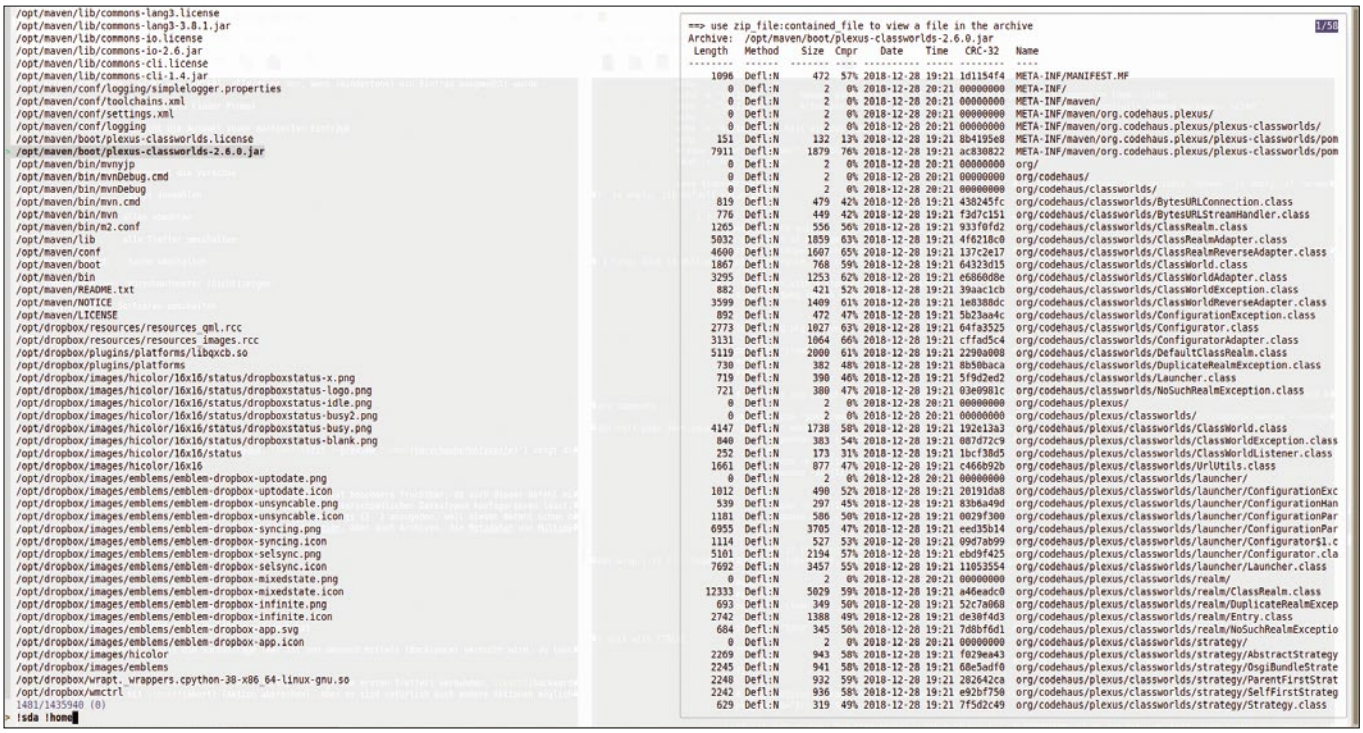


Figure 1: `fzf --preview 'less {}'` tells the program to display a preview via the less pager; however, this is primarily intended for text files.

You just need to specify `--preview 'less {}'` as the command. This is all it takes to display the contents of text files, for example, but also to archive content or the metadata of multimedia files, assuming that lesspipe is installed correctly.

You can define both the preview window's position and size using `--preview-window POSITION`. The main arguments for this option are the position (`top`, `bottom`, `left`, `right`), the size (as a percentage of the terminal width), and the mode (`wrap`, `cycle`, `hidden`). You need to enter this information in a colon-separated format without any spaces (Listing 6).

In addition to the preview, there is another way to generate a view of the content via actions. In the program, *Actions* means a whole group of functions that you can enable in the interactive Finder. You can also trigger Actions through events. The only events fzf can currently handle are `change`, if the search query changes, and `backward-eof` if the search query is empty and you still try to delete it using the backspace key. The change event is often used in combination with `first` (jump to first hit); `abort` is the default for `backward-eof`.

Key Bindings

One of fzf's special features is the ability to call other programs within the running program. This feature, which is controlled by the `execute` action, offers a wide range of possibilities. For example, you can open selected files to view or edit them. Fzf expects a command-line argument for the `execute` action. It then executes the command line in a subshell. The `{}` string stands for the selected files.

The `--bind` option is used to bind actions to a key or keyboard shortcut. You can use more than one action for a key binding. This is demonstrated by an example from the man page (Listing 7): The two forms are equivalent.

Creating your own definitions makes it possible to adapt the way fzf works to match other programs or even to add new functions. To group the `--bind` arguments, use single or double quotes if there are any spaces.

The current version of fzf supports only a subset of keys for bindings (see Table 5). The program ignores other combinations, such as Shift+Enter, which outputs a warning such as *un-Supported key: shift-enter*. But the software does accept mouse clicks in combination with the up arrow, down arrow, and double-clicks. Many actions simply delete characters or jump to certain positions in the

Listing 5: Match Preview

```
$ fzf --preview 'PREVIEWCOMMAND'
```

Listing 6: Preview Position

```
$ fzf --preview-window=top:55%:wrap:cycle
```

Listing 7: Defining Key Bindings

```
fzf --multi --bind 'ctrl-a:select-all+accept'

fzf --multi --bind 'ctrl-a:select-all' --bind 'ctrl-a:+accept'
```

line. Table 6 summarizes other, more specialized functions. Most of them are not normally mapped to keys, so they are not available by default.

There are two ways to execute external commands. If you use the `execute` action, fzf executes the command line, waits until the end of the command, and then displays its output. In many cases, this is precisely the desired behavior. To prevent fzf from waiting – and thus not accepting further input while doing so – you can append an ampersand to run the executed commands in the background. If you don't need the output of the executed commands, use the `execute-silent` action. The program will then run the commands but discard the output.

Conclusions

Both fzf and fzy are useful and practical tools capable of simplifying tasks in many areas. What I liked about fzf is the versatility of the feature set, the flexible options for defining the terminal layout, and in particular, the actions. What impressed me about fzy is its frugal approach and the optional ability to show scores. But both tools lack the ability to influence the algorithms in a targeted way, in contrast to, say, agrep. ■■■

Info

[1] Levenshtein distance: *https://en.wikipedia.org/wiki/Levenshtein_distance*

[2] "Better Finds" by Karsten Günther, *Linux Magazine*, issue 184, March 2016, *https://www.linux-magazine.com/Issues/2016/184/agrep*

[3] "Tutorial – ugrep" by Karsten Günther, *Linux Magazine*, issue 245, April 2021, *https://www.linux-magazine.com/Issues/2021/245/Tracked-Down/(language)/eng-US*

[4] fzf: *https://github.com/junegunn/fzf*

[5] fzy: *https://github.com/jhawthorn/fzy*

[6] lesspipe: *https://github.com/wofr06/lesspipe*

Table 5: Supported Keys

Group	Keys
Single keys	Enter, Space, Tab, Esc, Del, Up Arrow, Down Arrow, Right Arrow, Left Arrow, Pos1, End, Ins, F1 to F12, A to Z, \,]
Ctrl combinations	Space, Shift+7, A to Z
Alt combinations	Space, Up Arrow, Down Arrow, Right Arrow, Left Arrow, Backspace, A to Z
Shift combinations	Up Arrow, Down Arrow, Right Arrow, Left Arrow, ^, Space, Tab
Three-key combinations	Ctrl+Alt+A to Ctrl+Alt+Z, Alt+Shift+Up Arrow, Alt+Shift+Down Arrow, Alt+Shift+Right Arrow, Alt+Shift+Left Arrow

Table 6: Actions

Action	Explanation
`accept`	Apply the current entry
`accept-non-empty`	If an entry is selected, apply it
`change-prompt`	Change the Finder prompt
`clear-selection`	Deselect the previously selected entry
`preview`	Display preview
`refresh-preview`	Refresh preview
`select-all`	Select all
`deselect-all`	Deselect all
`toggle-all`	Toggle all matches
`toggle-search`	Find toggles
`toggle-preview`	Show (hide) preview window
`toggle-sort`	Toggle sorting
`disable-search`	Disable search
`reload`	Reload and rebuild index
`replace-query`	Replace search with current selection
`unbind`	Use `--bind` to clear the keyboard binding

Turn your ideas into reality!

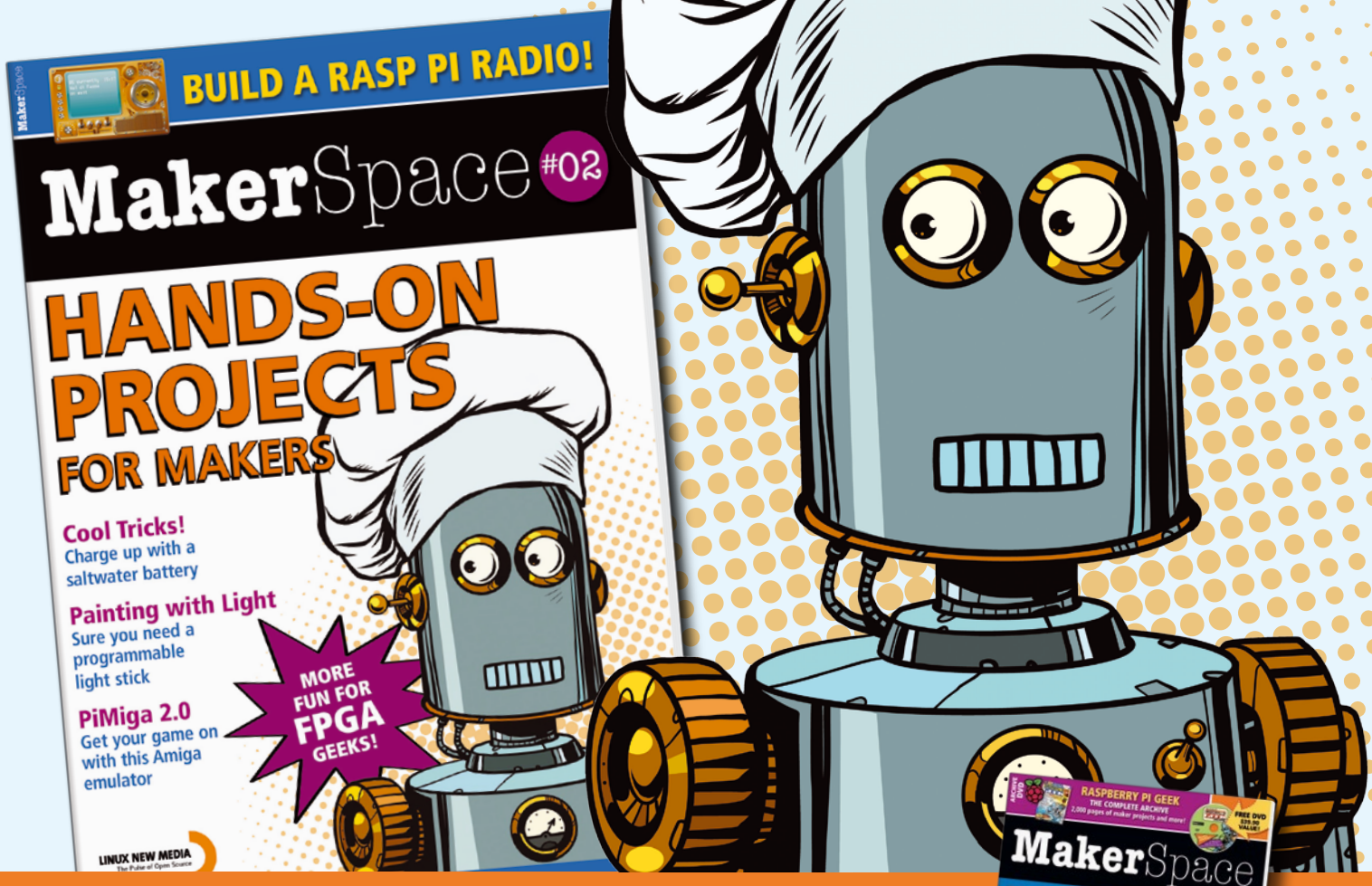
This is not your ordinary computer magazine! *MakerSpace* focuses on technology you can use to build your own stuff.

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

This special issue will help you dive into:

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

MakerSpace #02



ALSO LOOK FOR MAKERSPACE #01 AND ORDER ONLINE:
shop.linuxnewmedia.com/specials

A Plex alternative

Pilot Program

Dim, a relatively new open source media manager, looks to implement the appearance and feature set of the commercial Plex media center.

BY FERDINAND THOMMES

Many media players, also known as home theater software suites or media managers, are available for Linux. Of particular interest are the candidates that rely on the client-server principle, which users can host themselves and which also support secure access from the outside. The better-known representatives of this genre include the open source projects Kodi, Emby, LibreELEC, and Jellyfin, and the primarily proprietary Plex [1], which originated as a fork of the Xbox Media Center's (XBMC) Frodo version.

Although not open source, Plex is very popular on Linux because of its extensive functionality, good user interface, and suitability as a streaming media server. However, it has shifted too heavily in the direction of Netflix and Amazon Prime for many users, while basic features have been missing for years, such as smooth fading in and out of audio tracks.

For about a year, developers have been working on Dim [2], open source software that emulates Plex's appearance and functionality as closely as possible. First released in April 2021, the current version is Dim 0.3.0-rc6. By the time you read this article, version 0.3.1 should already be available or close to official release.

Developed from scratch, Dim is not based on an existing application. It aims to scan and play media from anywhere with minimal setup effort by the user. The project is still at an early stage, but Dim already promises to morph into a viable free alternative to the proprietary Plex. For that reason, I will take a closer look at how to install Dim, where the project stands today, and what its future plans are.

Installation

You can install Dim, which uses GitHub as its development platform, on your own hardware and

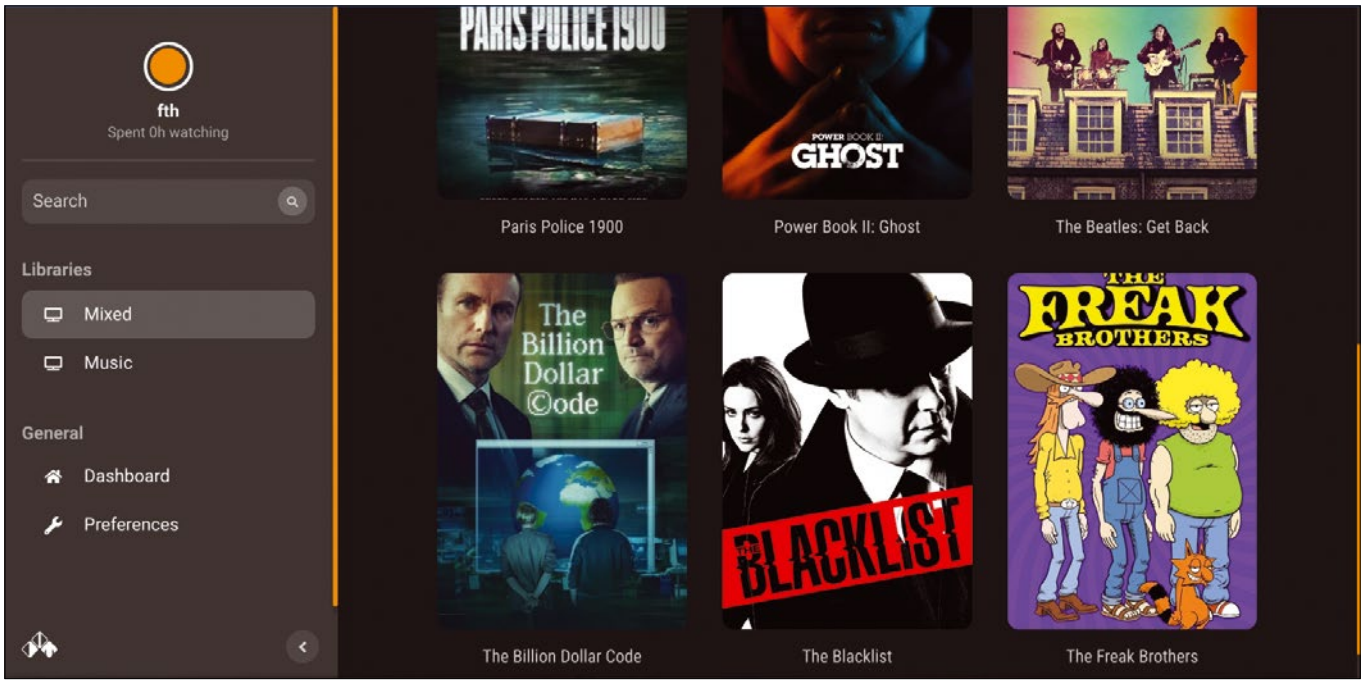


Figure 1: This view of a library shows the individual media as alphabetically ordered tiles. You can't set up a music library because the underpinnings are still missing.

Listing 1: Dependencies

```
$ sudo apt install libva2 libva-drm2 libharfbuzz-bin libfontconfig1 libfribidi0 libtheora0
  libtheora-bin libvorbis0a libvorbisenc
$ sudo dnf install libva libva-vdpau harfbuzz fontconfig fribidi libtheora libvorbis
```

control it via a web interface (Figure 1). You can set it up on a home computer or on a server on the web. All planned functions can be externally accessible if so desired. Containerization with Docker is an alternative to installing Dim directly.

One unique selling point is that Dim is implemented in Rust. The Rust programming language, which has been under development by Mozilla since 2010 and is now supported by a large community, is currently establishing itself as a second kernel language alongside C. The design brief for Rust is that it has to be simple and easy to use while offering better security and faster application execution speeds. Plex, on the other hand, which Dim is trying to emulate, is a Python program.

Because there are no binary packages of Dim for distributions yet, it is important to check the required dependencies on multimedia libraries before you install. The libraries should already be in place on a normal desktop installation, assuming that FFmpeg is installed. Listing 1 shows how to install the required packages on Debian and its derivatives (first line), as well as on Fedora and other RPM-based systems.

Listing 2: ZIP Installation

```
$ unzip ./release-linux.zip
$ tar -xvzf ./release.tar.gz
$ cd release && ./dim
```

ZIP or Docker?

Once all the dependencies are resolved, download the ZIP file of the current Dim version from GitHub [3]. Unpack it as shown in Listing 2, change to the unpacked directory, and run the configuration process there (Figure 2). After completing the action, call the Dim web interface by typing *http://0.0.0.0:8000* in your browser's address bar. If needed, you can change the

Listing 3: Docker Installation

```
### Install Docker for Debian and derivates:
$  sudo apt install docker.io
### Install Docker on Fedora:
$  sudo dnf install docker-ce
### Download the latest Dim image:
$  sudo docker pull ghcr.io/dusk-labs/dim:dev
```

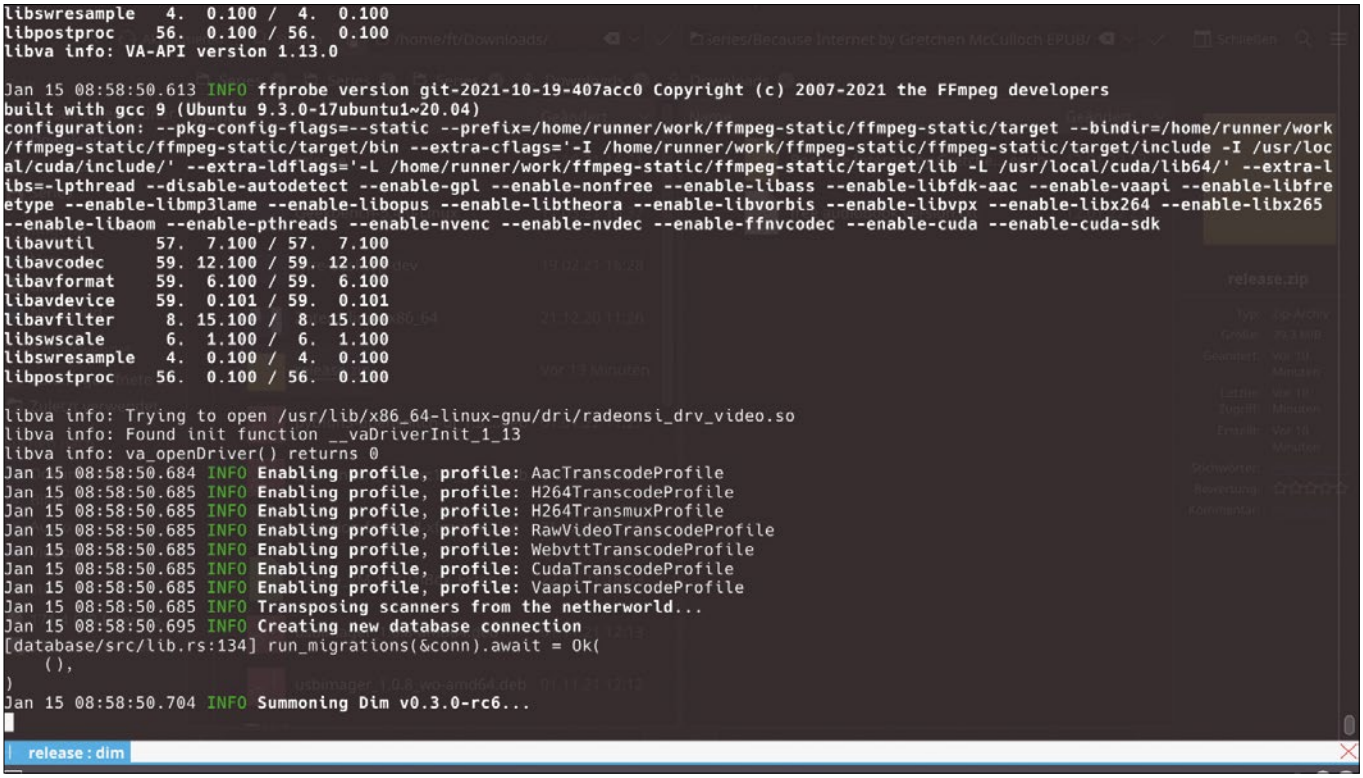


Figure 2: After unzipping the archive file, trigger the Dim configuration. The software checks if the dependences are in place.

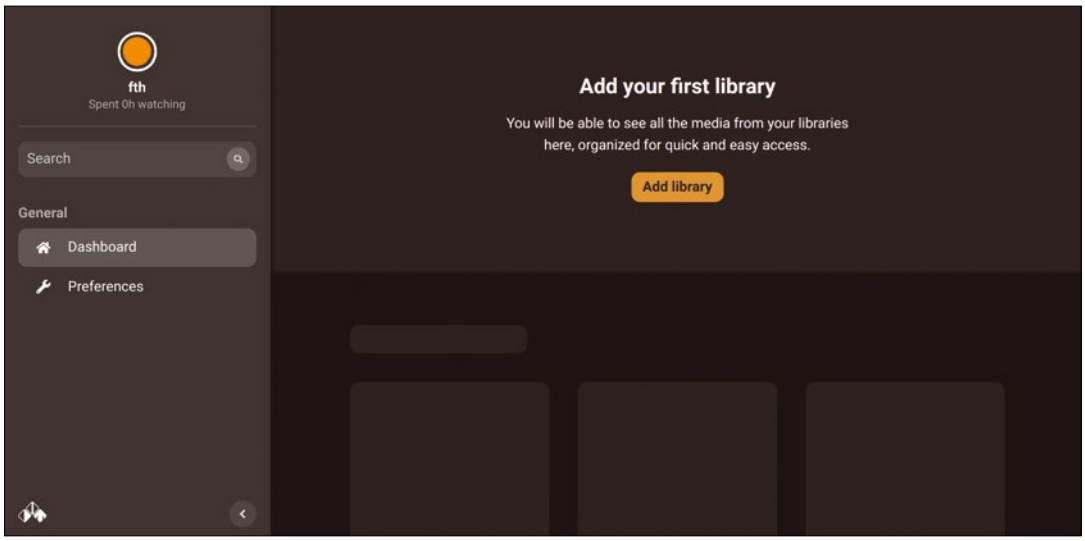


Figure 3: The Dim interface has been deliberately designed to be clutter free. On first launch, the application prompts the user to create a library.

port number for the service there. Listing 3 describes how to install with Docker. You will find the image, including a README file, at `/var/lib/docker/overlay2`.

Once you have set up Dim and opened the web interface in the browser, the next step is to create an account. You can then start populating the media database (Figure 3). Currently, Dim is limited to movies, series, and anime. Music, images, and other media formats will follow as soon as the developers are satisfied with the implementation of the formats that are already supported.

In the web view, like in Plex, a sidebar on the left with the controls takes up about a quarter of the window width. The rest of the display is used to show the indexed media. Below the user logo, which you can customize to suit your own preferences, you will find the search bar and below that again the collections you created. If so desired, you can hide the control sidebar using the small arrow bottom right, which means you can use the entire screen to display the media and metadata.

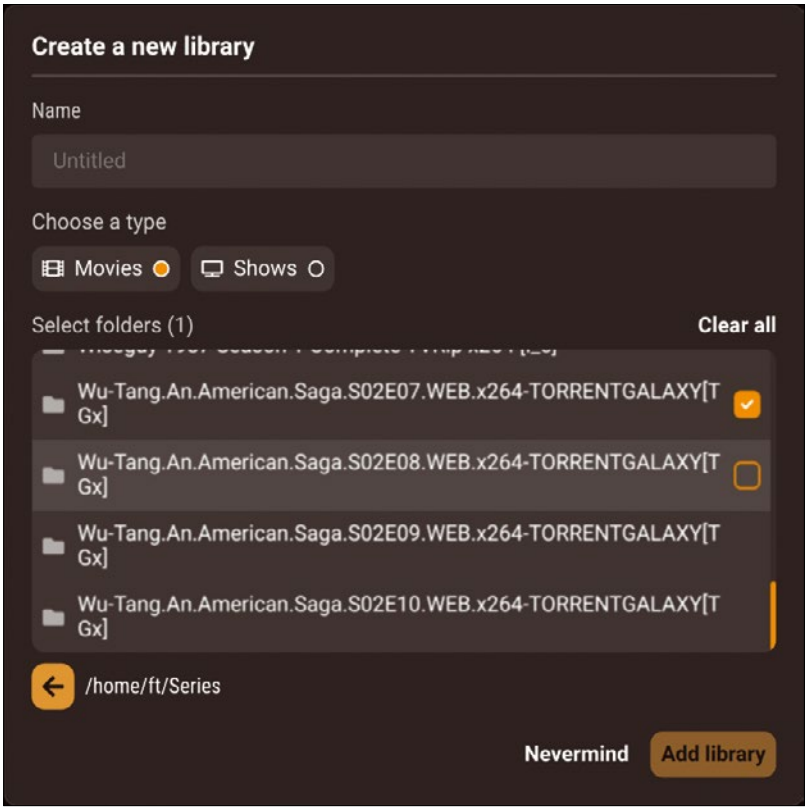


Figure 4: When creating a library, it is important to make sure that it has a name and that the boxes to the right of the folders are checked.

Libraries

When you log in for the first time, the application prompts you to create a library by assigning a name and passing the path to the desired files to the Scraper (a small utility or built-in routine that collects data from various sources before typically serving up the data centrally via a database). Use the option here to define whether the library is for movies or series. Dim treats the two categories differently as far as the metadata – which comes from The Movie Database (TMDB) – is concerned.

Dim only supports folders. It does not show individual files in the overview at all. However, a mode to include individual files is on the roadmap. If you hover the mouse pointer over a folder, an orange rectangle appears on the right. You can then click it to select the folder (Figure 4). Don't forget to assign a name to the collection, otherwise the *Add Library* button will remain disabled. If you want to create more collections, mouse over the *Libraries* tab and click on the plus sign that appears.

Below *Libraries*, the *Dashboard* tab shows the most recently viewed media at the top. Above the file name, you will find automatically generated

tags such as the year of creation or genre. Clicking on the keywords, which are defined as links, takes you to other media with the same tag. Dim does not currently support individual tagging.

Very Little to Set Up

The *Preferences* tab lets you change your username and password and customize the default port on which the web interface listens. There is a choice between a few dark and light view modes, and you have the option of customizing the paths for the cache and metadata. Under the hood, the application relies on Nightfall [4], a library for transcoding and streaming various video files on demand (Figure 5).

Due its early stage of development, Dim offers only a basic set of the planned features thus far. This includes the ability to import video files, including transferring the metadata (Figure 6), and the option to play back your media collection locally (Figure 7). During my testing, all of this worked without any trouble. Only one of the 20 loaded files lost its metadata during the process.

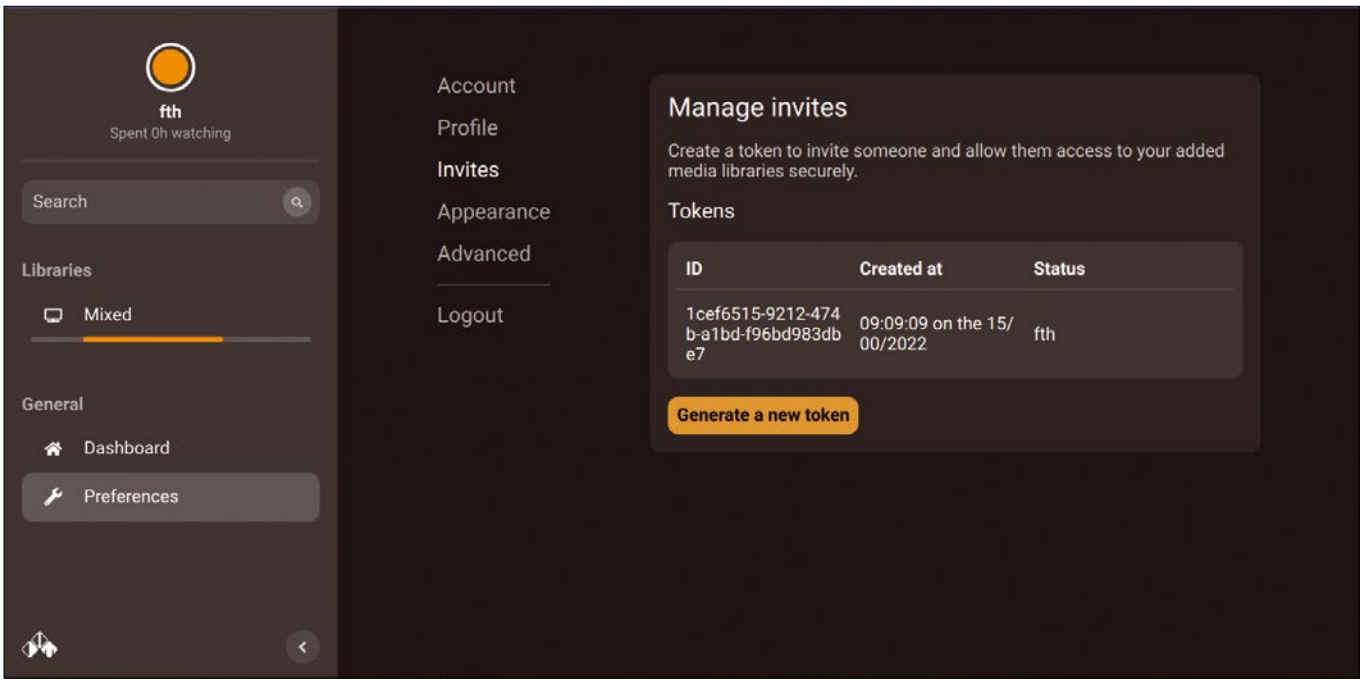


Figure 5: Clicking on *Preferences* takes you to the settings. In addition to the standard settings for your profile, there is also an *Invites* tab that lets you create tokens to allow secure third-party access to the libraries.

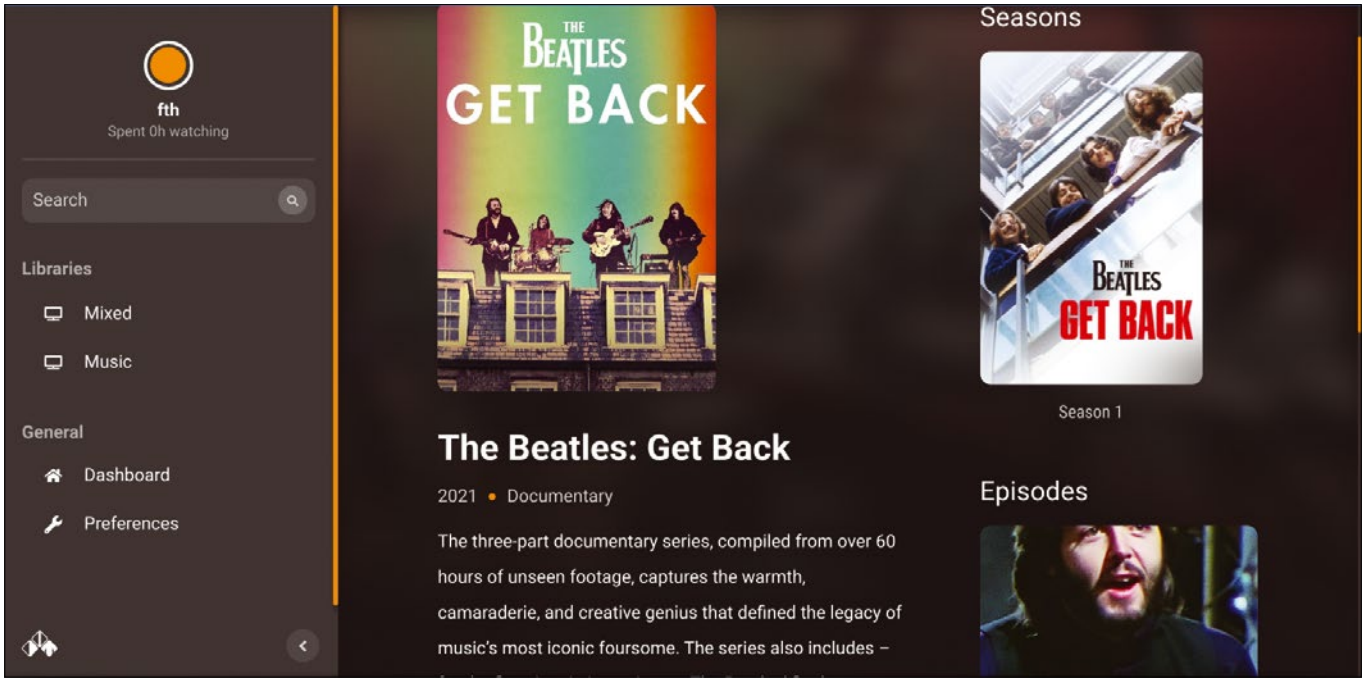


Figure 6: Clicking on the entry in the library opens the metadata view. This is also where you launch playback.

Outlook

I asked Dim's lead developer what plans the project has for the coming year or two. This year, the main focus is on polishing up the web UI as well as completing the mobile app for Android and iOS, which is currently in the design phase. The team is also working on enhancing the overall user experience. Alongside this, the developers are looking to deliver a native desktop app with MPV as the video player later this year. In the longer term, they plan to allow users in different geographical locations to stream a video together.

Clients for mobile devices and a large TV offering are among the most important prerequisites for the success of a media center. In 2022, the focus of development will therefore be mainly on Android TV and Apple TV clients. The feature list includes all the usual functions, including offline playback. In addition, there is DirectPlay support with a large number of codecs, so that there is no need to transcode the media in most cases for local playback. The developers are planning comprehensive subtitle support for the web interface. In addition to the text-based SRT format, Dim already supports the SSA format [5], which is often used in anime. In addition, the ability to play back videos with bitmap subtitles without transcoding will soon be available.

Given the scope and complexity of the upcoming developments, especially in the area of clients for mobile and TV, the project will need to pay its developers. The intent is to generate funds with paid plugins offering advanced features, such as multi-GPU support or transcoder load balancing.

Conclusions

The goal of replacing Plex with an open source application such as Dim is something that cannot be accomplished in just a few months; instead, it is likely to take years. Having said this, the Dim team has already laid down solid foundations in the first 12 months of development. The simple interface is appealing in its unobtrusiveness. The implementation in Rust promises speed advantages over an implementation in Python or C/C++. Right now, Dim is not suitable for practical use, but I suggest keeping an eye on this project in the hopes it can rightly claim a place in the living room. ■■■

Info

[1] Plex: *https://en.wikipedia.org/wiki/Plex_(company)*

[2] Dim: *https://github.com/Dusk-Labs/dim*

[3] Dim download: *https://github.com/Dusk-Labs/dim/releases/tag/v0.3.0-rc6*

[4] Nightfall: *https://github.com/Dusk-Labs/nightfall*

[5] SSA: *https://en.wikipedia.org/wiki/SubStation_Alpha*

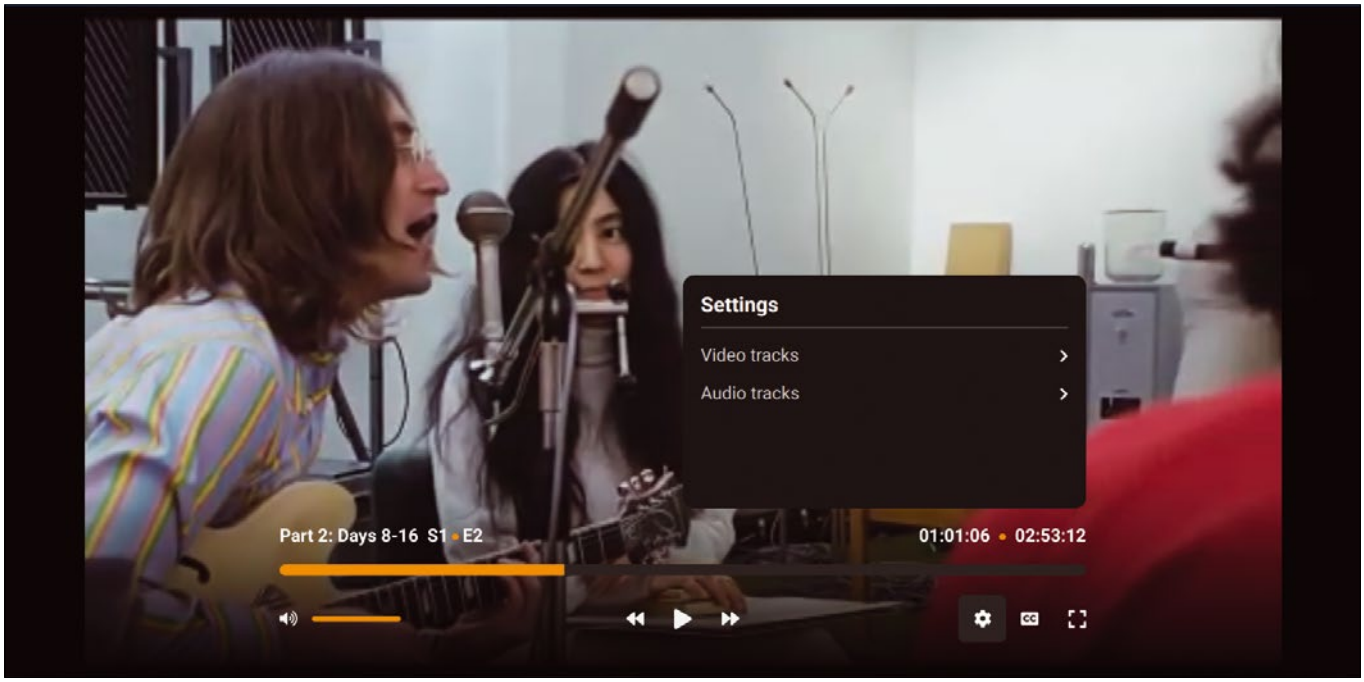


Figure 7: The player window shows the developers' focus on the essentials and only offers the controls users absolutely need.

Linux Magazine **is your guide to the world of Linux. Look inside for advanced technical information you won't find anywhere else!**

Expand your Linux skills with:

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

If you want to go farther and do more with Linux, subscribe today and never miss another issue!

Subscribe now!

shop.linuxnewmedia.com/subs

GET IT NOW!
FAST DELIVERY WITH OUR PDF EDITION

FOSSPicks

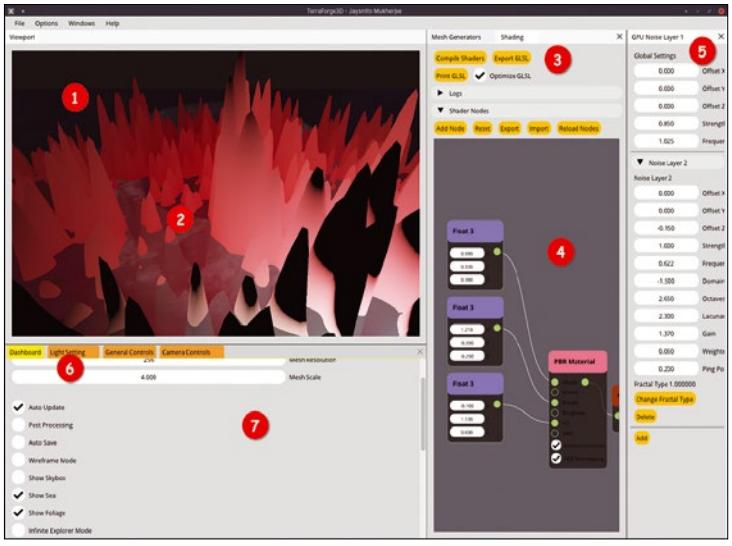
Sparkling gems and new releases from the world of Free and Open Source Software

As his Steam Deck has still not arrived, Graham this month has been able to get several open source games running on his ancient PlayStation Vita, including Heroes of Might and Magic. BY GRAHAM MORRISON

Landscape generator

TerraForge3D

Back in the early 1990s, if you were lucky enough to have a Commodore Amiga, then you would have encountered a brilliant application called Vista, and later, Vista Pro. These weren't games, yet they weren't directly productive either. Instead, they were three-dimensional portals for the imagination, letting you render and explore real and alien landscapes on old hardware. Early versions were even given away as magazine cover disks, and it was a piece of software that could severely test both your patience and your humble Amiga's computing resources. Vista generated landscapes which, at the time, looked incredibly realistic. Taking a seed from a fractal, or a real digital elevation map, Vista would draw a three-dimensional point-of-view of the landscape from your chosen location. But the clever part was how this view was augmented with grass and rock textures, rivers and lakes, and even trees in later versions. These elements were all generated algorithmically by interpreting the terrain, growing grass on lower gradients and tumbling rocks on higher gradients, and placing rivers between the valleys. A carefully crafted scene could look incredibly realistic for the time, and one such was even used by the great science fiction author, Arthur C. Clarke, to illustrate what a terraformed Mars might look like, using Vista Pro and real elevation data for Mars.



1. **Viewport:** Your virtual landscape is visible here, updated in GPU-accelerated real time. 2. **Water and sky:** A water level can add an animated sea, and there are sky box textures for the sky. 3. **Shaders:** GLSL shaders are supported to create deeply complex textures. 4. **Shader nodes:** Create shaders that react to the landscape with a Blender-like node editor. 5. **Mesh generators:** Layers of landscape distortion can be combined using the GPU or CPU, randomly or from fractals. 6. **Lighting:** Global lighting can be adjusted in color and angle. 7. **Explorer mode:** If you simply want to walk around your created landscape, there's a full screen explorer mode.

An Amiga 500 would take hours to generate a single image, so it's surprising that terrain-generating applications haven't appeared to continue Vista's premise after several generations of Moore's Law hardware updates. Modern video games probably get close to this potential, and they can look remarkable, but players rarely have any control over how their landscapes are generated, let alone getting to explore them without being attacked by Karstaag the frost giant. Which is why it's so exciting to see the release of TerraForge3D, a hugely ambitious cross-platform procedural terrain generator with the same objectives as the old Vista – a three-dimensional portal for the imagination.

TerraForge3D is still at a very early stage of development, and it is difficult to use. But even in this early phase, it's capable of rendering beautiful, surreal, and hyperrealistic terrain, usually in real time thanks to modern graphics hardware. The image generation process starts with a mesh, and either CPU- or GPU-bound displacements maps. Apart from the sky box and global lighting settings, almost everything else is modular and can be layered atop each other. Even the panels in the main window can be dragged out of their positions and locked into various edges and locations to create the best working environment for you. The same is true of textures and GLSL-based shaders which are added to the mesh to add the detail. Changes are rendered immediately, allowing you to adjust values and sliders to visually choose the output you prefer. The latest release lets you modify shaders and textures in a node view, where different attributes can be connected to each other to adjust and modify the overall effect. It's a lot like the node system in Blender, and equally arcane until you've read the documentation (or watched some of the excellent YouTube-hosted video tutorials), and connected lots of nodes together. The output will often need considerable tweaking and experimentation to look good, but with a little practice TerraForge3D can produce the kind of results worthy of any magical descendent of Vista Pro.

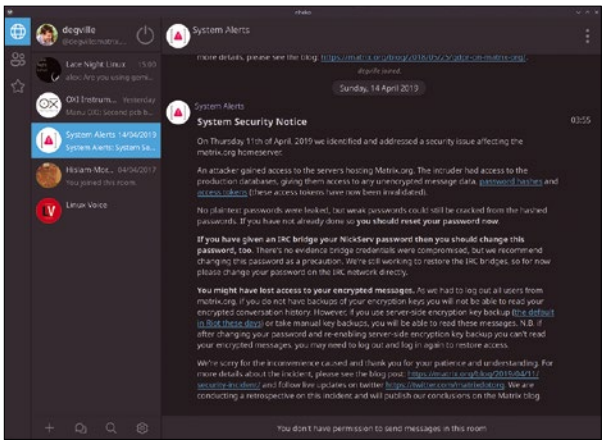
Project Website
https://github.com/Jaysmito101/TerraForge3D

Matrix client

nheko

The Matrix messaging protocol has been around for almost 10 years, and over that time, many of us have experimented with the network for personal and group chats at some point, usually before moving back to our regular sources of messaging distraction. In theory, Matrix is both an excellent successor to IRC and a solution to the problem of secure trusted messaging because it's a) simple, b) secure, and c) potentially self-hosted. In its early years, Matrix was lacking a diverse community of participants and clients, but its popularity has been increasing in waves, reaching a critical mass of acceptance. This acceptance has been helped by the development of some excellent clients, including the open source Element app on Android and a desktop application called nheko that's been specifically designed to replace your favorite graphical IRC client.

This is a good thing because Matrix can feel a lot like IRC, with its federated conglomerate of instances and shared networks. Similarly, it's not always intuitive, but the modern Matrix experience in nheko is refreshing. The Qt-based UI is clean and easy to navigate, mimicking the web interface of the Element Android app. There's even a welcome option for GUI scaling, which is great for high-DPI displays, and it's secure by default. You'll need to go through the Matrix cross-authentication process if you're using another client at the same time, but you can then be sure your encrypted conversations are secure, as are voice and video calls, all of which can be done from nheko.



Nheko is a beautiful Qt-native desktop app for accessing Matrix messaging and group chats.

The groups/rooms you're subscribed to are listed on the left with the messaging window on the right, and you can search for public groups and users. You'll find less engagement than on other social networks, but the quality of the conversation is generally higher. At least for now, as this is a typical trade-off between a platform's popularity and the level of signal-to-noise.

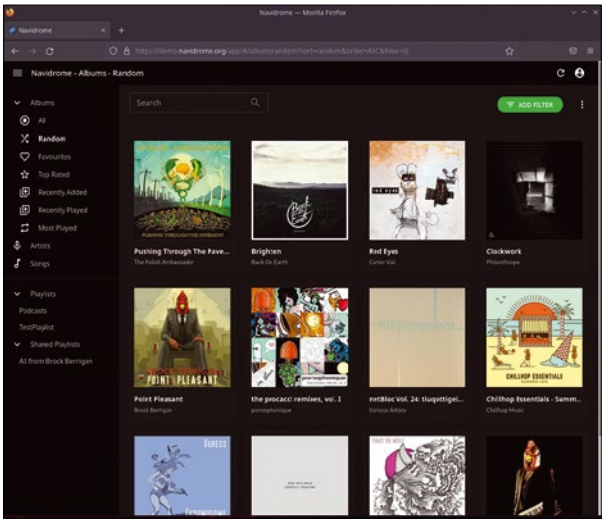
Project Website
https://github.com/Nheko-Reborn/nheko

Streaming server

Navidrome

Why is it that while we have more choice than ever before, it's sometimes harder to access exactly what we need in the ways we want to access it? Movie- and music-streaming services are the best examples of this problem, because, while there's never been more choice, if you happen to want to watch or listen to something specific, you're limited to whatever movies or music those services provide and the formats they provide them in. Which is why it's beginning to make sense, again, to start buying your own media and hosting your own collections. Again.

There's always been software for doing this, from the venerable Logitech Media Server (still being developed by the open source community), to the Plex-like multimedia server Jellyfin. But it was difficult to find something with a modern music focus, especially after the demise of Google Play Music. This is where Navidrome can help. It's an open source music server with a web front end that looks like Spotify and is compatible with the Subsonic API, a closed source alternative. This means you can use any Subsonic compatible app to play your music from your tablet, smartphone, or a simple web browser. The server itself is easy to install and perfect for a Raspberry Pi, but it's the web UI that really succeeds. The default theme is called "Spotify-ish," but it's quicker and less resource hungry than its namesake. Album art will animate, and music and podcasts are easy to search through, play, or pull into a playlist. There's even support for lyrics if you want to host your own karaoke evening. The most important thing is the sound quality, which will default to the native format of the audio but transcode to the client's capabilities if necessary. You can see what's happening for each client from the user account page. The only negative aspect is dealing with the security of the server yourself, but this can be solved by setting it up at home and accessing it via a VPN.



Navidrome even implements one of Spotify's best secret features, the ability to scrobble what you play to Last.fm.

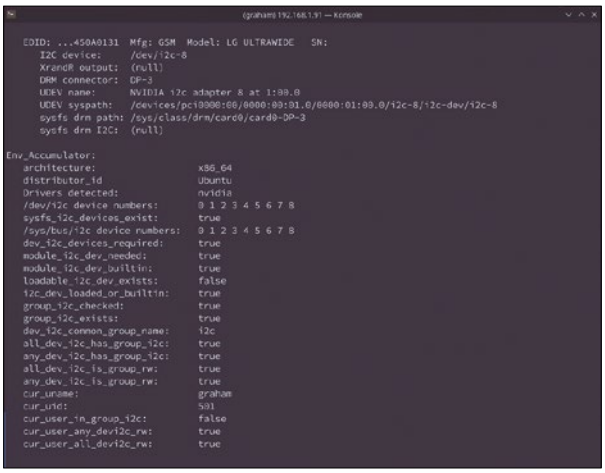
Project Website
https://www.navidrome.org

Monitor controller

ddcutil

It's amazing that the same companies responsible for incredible display hardware engineering, weaving organic light-emitting diodes together at the sub-millimeter level, can't create a decent on-screen navigation menu system. Perhaps it's a case of hardware engineers versus software engineers, but for whatever reason, many otherwise wonderful screens are crippled by a single button and a menu system designed to test the patience of a saint. Fortunately, the hardware engineers have a trick themselves in the form of the Display Data Channel (DDC), a two-way communications protocol that can transfer messages between your computer and the majority of modern screens. This means we're free to reinvent those menus ourselves.

The extent of what can be controlled over DDC is entirely dependent on the monitor, but it usually includes the elements you can find in their on-screen menus, including brightness, color profiles, and input selection. The only problem is how you access this functionality as a user. The answer is a brilliant little command-line tool called `ddcutil`, which enables access to whatever functionality your monitor hides within the DDC protocol. A good place to start is with its `ddcutil detect` command, which will attempt to find any connected screens and report on their capabilities. If this fails to find enough details, the `ddcutil interrogate` command will try harder and take longer. You can then use other commands, including `setvcp`, `display`, and `watch` to configure a color profile, switch hardware input, and watch changes respectively. There's a lot more power and complexity here, which can take some unravelling, but it's still easier to use than your monitor's on-screen menu.



Even ancient VGA ports could talk to your PC over I2C and DDC, though some modern displays now use USB, which can also be probed with `ddcutil`.

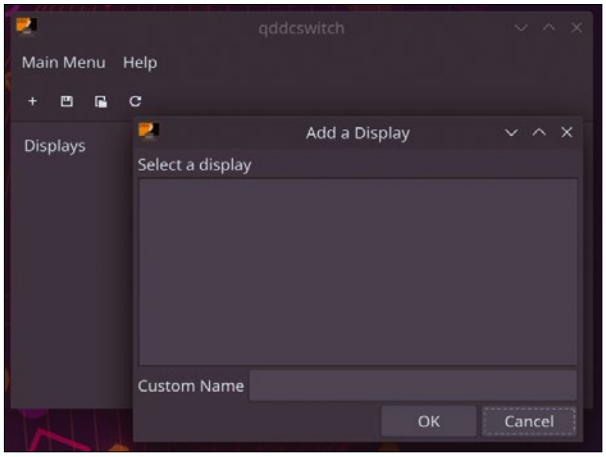
Project Website
http://www.ddcutil.com/

Display input switcher

qddcswitch

If you've read the above description of the wonderful `ddcutil`, used to control various aspects of your display, you'll see that `ddcutil` is not a simple tool to use. To get the most out of it, you need to introspect your hardware and understand the values it accepts and returns. It's powerful but complicated, especially if all you need to do is change which input to display. Many of us will have more than one computer connected to a single screen, which will often have more than one HDMI or DisplayPort input. But if all you want to do is change the input, then `qddcswitch` is a much better option. This is a small graphical application that does away with needing to understand anything about your hardware or the `ddcutil` commands necessary to control it. Instead, it works out what's connected to your computer automatically and presents this information in an easy-to-interpret grid.

This grid is most useful if you have more than one screen connected, with more than one computer connected to those screens. Screens are automatically detected and presented in an *Add a Display* list from which you can choose which screens are included. This is a useful way of making sure you don't switch the screen with `qddcswitch` while you're managing other connections. Each screen has its own row in the grid, and each screen's inputs are listed to the right in columns. You can then easily switch between the various inputs of all your connected screens. It's a brilliant way to switch between inputs and much easier than using the typical menu system you find on most monitors. The only thing missing is builds for other operating systems so you can have `qddcswitch` running on any computer you own. But this is hopefully just a compile away.



Alongside the simple Qt-based GUI, `qddcswitch` can also be driven by an equally easy-to-use command-line client.

Project Website
https://codeberg.org/Okxa/qddcswitch

Modern diff

Difftastic

There have been command-line tools for comparing files ever since there was a command line. The most famous is of course the venerable `diff`. This will, by default, compare two files and print output with annotated < and > symbols to show which lines have been removed and added. Other modes, such as the context format, can make this easier to understand and parse by other tools, including `patch`, but it always rigidly counts differences without any background context on what might cause those differences outside of an intentional edit. This is where Difftastic can help. It's a modern version of the same `diff` command that understands the context of the files it's looking at, hopefully reducing errors and making it easier to use.

Difftastic is run from the command line as `difft` and will take the same arguments as `diff`. Unlike `diff`, however, `difft` can specifically parse files written in over 20 programming languages. This means it can understand when differences are just whitespace in formatting rather than syntactical differences in a programming language. It can also understand nesting and whether line wrapping is meaningful for whatever languages it's checking for. Each of these examples would cause false positives from the original `diff` and make it considerably harder to see the real changes in the code, especially when dealing with pull requests written by different developers.

Difftastic is great as the default `diff` tool for Git and also for any CI system you may run because it makes it much easier to see any file differences.

There's also a side-by-side display option, as well as the normal `diff` output modes, and it will fall back to traditional difference tracking if it can't discern the languages or formats involved. It's a great alternative to the original `diff` if you're using one of the supported languages. If you're not, adding your language of choice to the project is a great way to contribute to open source.

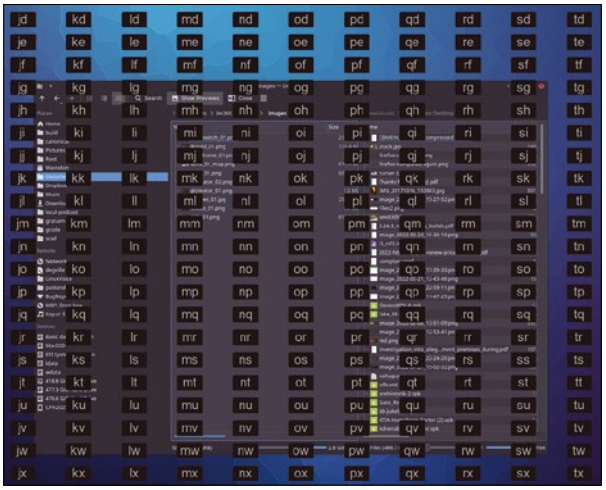
Project Website
https://github.com/Wilfred/difftastic

Virtual pointer

warpd

Not so long ago, we looked at a keyboard firmware editor called Chrysalis. It was designed to work only with a few specific open source keyboards using the Kaleidoscope firmware, usually based on an Arduino, but these keyboards were definitely something to wish for. Thanks to Chrysalis and the open firmware, you could easily modify the firmware to suit your own needs, including adding a variety of plugins contributed by other users. These plugins could do things such as run macros from shortcuts, create one-shot special keys without pressing shift, or turn the spectrum of LEDs beneath the keys into a rainbow. One plugin, however, could imitate mouse control, not only letting you move the mouse cursor with whatever shortcuts you configured, complete with acceleration and start/end thresholds, but also warp the cursor to different quadrants of the display with a few other shortcuts. Brilliantly, the `warpd` project has been able to recreate the same behavior in software for any keyboard.

Really the combination of two ideas, `warpd` gives you quick and easy control over your mouse cursor from the comfort of your keyboard. The first is the aforementioned warping, which `warpd` confusingly calls Grid Mode. With this enabled, you use the *u*, *i*, *j*, and *k* keys to navigate to different quadrants of the display, highlighted within a red border. Successive key presses reduce the size of the quadrant so you can quickly jump from one side of the screen to the next, with the *m*, comma (,), and period (.) keys used to trigger a left, middle, or right click when you get to your target. There are two other modes, too. The first is called Hint Mode, which works like Vimperator in Firefox. With this enabled,



Although you can move the mouse cursor with keys in KDE Plasma, `warpd` adds extra navigational modes that will work with any (non-Wayland) desktop.

the screen is covered with letters or keys you can press to immediately send the cursor to that location. It looks messy but is slightly more intuitive than Grid Mode. Finally, there's Normal Mode, which gives you keys for moving the cursor around manually with the old school Vim navigation keys. They all work brilliantly and can save you from repetitive strain injury and even from buying a mouse.

Project Website
https://github.com/rvaiya/warpd

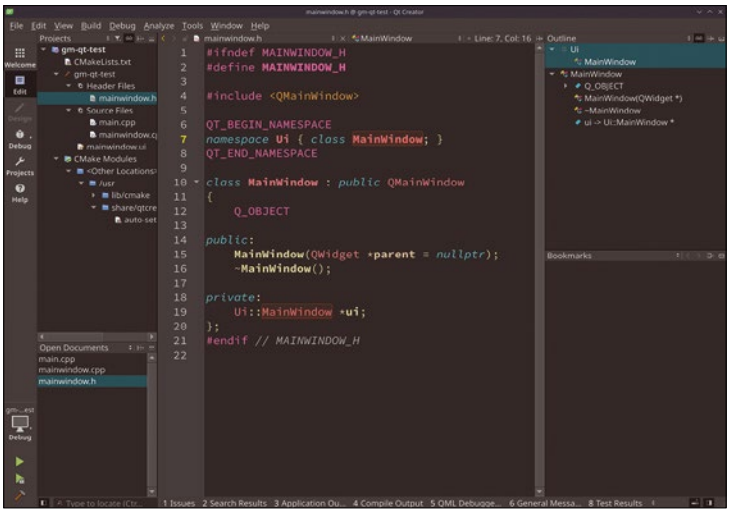
Integrated development environment

Qt Creator 7

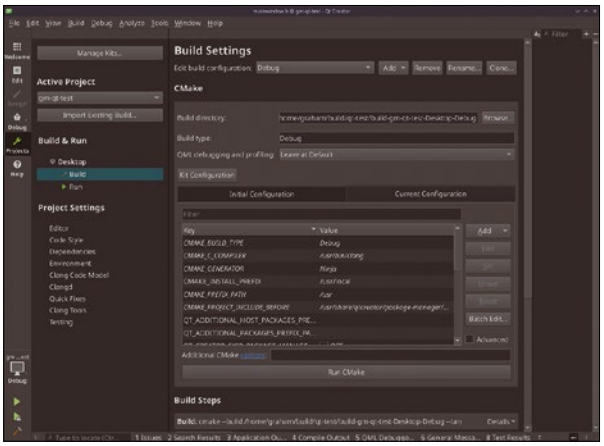
Qt Creator is an integrated development environment for building Qt-based applications that also happens to be great at building all kinds of other applications and projects too. It started life in 2007 as a side project called Greenhouse and was worked on by a few developers at Trolltech just before it was acquired by Nokia. Back then, there were few open source IDEs that could compete with Microsoft's Visual Studio, especially for C++-based languages or for projects using CMake to maintain the build environment. KDevelop had come close but was on hiatus while the project reinvented itself. This made QT Creator genuinely revolutionary when it did arrive because it was one of the first open source IDEs to include code completion, object abstraction, and an open back end for other languages and frameworks.

This latest release of Qt Creator finds itself in a new era with a new set of challenges. Qt has itself undergone several significant management and licensing changes, with the project now being owned and run by the Qt Project as its official body. The licensing changes mean binaries can only be downloaded with an account, and LTS releases are only available to commercial licensees. This isn't against the terms of its open source license, but it's not in the spirit of open source. Fortunately, Linux distributions get around this by building their own binaries and maintaining their own patches, but it's still awkward. And of course, Qt Creator is now no longer the only IDE. Microsoft's Visual Studio Code, in particular, has done a brilliant job capturing developer attention, from Arduino programmers and JavaScript tinkerers all the way to collaborative online coding with colleagues and project management, ironically (for Microsoft), without any open source code license ambiguity or requiring logins to download binaries.

Qt Creator, however, continues to improve, and version 7 is a big update. For Qt and QML application development, it remains unparalleled, and there are still many good reasons for using Qt. It's obviously the base for KDE Plasma development, and it offers an incredible breadth of libraries for accessing hardware, MQTT, NFC, PDF rendering, accelerated graphics, and all kinds of string, number, and time manipulation. There are hundreds of GUI elements that can be incorporated dynamically into your application with the Designer tool, and it's all genuinely cross-platform, marshaled by Qt Creator to build native-looking applications for macOS, Windows, and Linux (and Android) from the same codebase. Qt Creator helps you do all this by integrating help, example projects, and dynamic links to Qt's extensive API reference from the main editors.



Qt Creator hasn't changed a great amount since the early versions, but its syntax highlighting, integrated help, and code completion functions are still almost unequalled on Linux.



One of Qt Creator's best functions is to make and manage CMake files, in addition to auto-detecting toolchains for old projects you import.

This new release is also an important nod to the future. The most important change is that clangd is now the default back end for all C-language based processing, including code completion and highlighting. This replaces the archaic library back end which was slow and CPU intensive. CMake will also default to using C++17, and there's even support for Wayland on Linux, although you'll probably have to manually enable it first. Android app builds have been improved, as has Qt Quick for rapid application development, but the fundamental layout and process haven't changed. If you've been tempted by Virtual Studio Code recently but still work within the C++ programming domain, regardless of whether that's with Qt or not, Qt Creator is definitely worth a revisit.

Project Website
https://www.qt.io/

Space trading, exploration, and adventure.

Vega Strike

Vega Strike freely runs on Linux, as well as Windows and even Android, if you want some portable deep-level distraction.

The original version of Elite for home computers in the 1980s spawned several inferior clones. This was because no one had considered creating a game within a sandbox before, where you could choose whether to fight or trade, or simply lose yourself in the infinity of space. And it's been a similar story over the past 10 years, where Elite's modern sequel, Elite Dangerous, has dominated over the clones in the same way. But Elite Dangerous never officially ran on Linux, and recent updates have been disappointing. Leaving the community again looking for clones that could this time perhaps eclipse their inspiration. Vega Strike isn't quite that, but it's open source, runs on Linux, and gets close. The gameplay is similar to Elite where you fly your ship either into combat, adventure, or on trading missions, into beautiful space environments or against huge space stations and other shops. But most importantly, it's still being developed.

Vega Strike has been in development for 20 years and recently enjoyed a significant update with its 0.8.0 release. A lot of the work for this release has been to make Python 3 the default for the official build and to update the game engine to versioned APIs, which should make it easier for the game engine to be used with other games. The API is already being put into action with a split between the game engine and a story-mode package called Vega Strike: Upon the Coldest Sea, which features a fully dynamic universe that advances independent of the player's actions, much like Elite Dangerous. The core game now also features high-resolution background images and textures, and while graphics might still be considered austere, they don't affect the gameplay and have a charm of their own. This also means you can play Vega Strike on almost any hardware, including a Raspberry Pi, and it's definitely worth both the effort and the support for the future.

Project Website
https://www.vega-strike.org

Heroes of Might and Magic

fheroes2

To play Heroes of Might and Magic II, you'll need access to the original game files, which can still be purchased through GOG.

Heroes of Might and Magic was a brilliant series of games that were developed from the mid-1990s into the modern era. They were turn-based-strategy games with mission-based objectives. Depending on the character you started as, you'd get an assortment of artifacts, spells, and capabilities that shaped your approach to the game as you fought, developed resources, and adventured across the landscape in the hope of fulfilling your prophecy. The early games in particular are steeped in the pixelated minimalism and uncompromising gameplay that has become popular recently, especially when the teenagers who grew up playing the originals are now fully fledged software developers. Which of course makes them perfect targets for open source reinvention.

Open source reinvention has happened twice with Heroes of Might and Magic, and there was even a native Linux version of Heroes of Might and Magic III developed by Loki. This game can now be played with a modern open source engine called VCMI, although you'll need the original data files from the original Heroes of Might and Magic III: Shadow of Death or Heroes of Might and Magic Complete Edition, and not the files from the Loki version. Heroes III is a good place to start because it adds gameplay elements such as "waiting" and a more refined user interface, but its precursor, Heroes of Might and Magic II has its own open source engine too, called fheroes2. This is a great place to start if you've played Heroes before and intend to revisit Heroes III later. As with later titles, the game is split into strategy, exploration, and a combat mode that's similar to Final Fantasy Tactics, albeit across a hexagonal chess board. Thanks to this, it's still a lot of fun and definitely worth playing today.

Project Website
https://github.com/ihhub/fheroes2

Creating charts with LibreOffice Calc

Plotting and Data Visualization

Everybody needs charts sooner or later, and LibreOffice Calc is the easiest way to create them with free and open source software.

BY MARCO FIORETTI

Modern life is full of numbers. Even if one is not a mathematician, sooner or later comes the day when it's necessary to quickly understand or share with others the relationships among numbers. This tutorial introduces what is probably the simplest way to do just that with free software, the charts in LibreOffice Calc [1]. But are they charts or graphs? Let's clear up that question first.

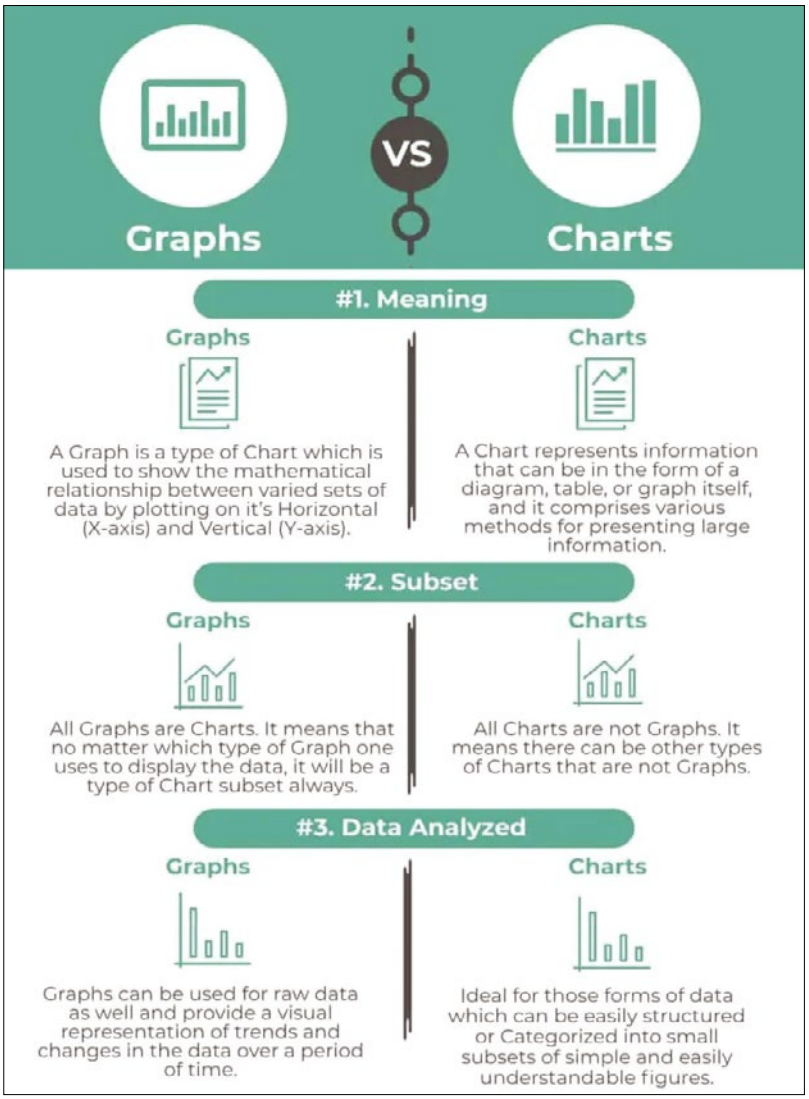


Figure 1: All graphs are charts, but not all charts are graphs [2].

Charts vs. Graphs

Both charts and graphs are tools to summarize and present information in a visual way. While most people use the two terms as if they were synonyms, strictly speaking, they aren't [2]. Some of the reasons are shown in Figure 1, but basically charts summarize datasets in ways that are (hopefully) intuitive and engaging, for example, with bars, pies, or other symbols. The primary purpose of charts is to convey the high-level meaning of data and the connections within it.

Graphs, instead, are the subset of charts specifically designed to show the actual mathematical relationship between raw numeric data points. Another way to express the difference may be that charts are better for business presentations or, in general, any high-level description of (not necessarily numeric!) data, whereas actual analyses of numeric formulas would require graphs.

In practice though, that distinction can often be quite hard to figure out, let alone apply. For most users the semantic differences above will likely matter much less than figuring out what *kind* of Calc chart style best fits their needs (more on this later). Therefore, because LibreOffice Calc calls its data visualizations "charts," I will only use that term from now on.

Because charts can only display already existing data, you first need to learn how to quickly fill lots of Calc cells with numbers to play with. Indeed, this is a trick well worth learning regardless of charts because it provides the raw material through which you can learn any function of any spreadsheet.

As an example, say I want to visualize the number of downloads of three different versions of a generic Linux distribution in the first 30 days after its release. This requires a spreadsheet table like the one shown in Figure 2, where days from release and numbers of different daily downloads are arranged in columns, with labels in the topmost cells of columns 2-4 to identify each distribution version.

I generated the leftmost column in Figure 2 by typing *1* in the second cell and then dragging the bottom right corner of that cell down for 30 rows.

When you do this on a cell containing integer numbers or dates, Calc fills the underlying columns with incremental values of the same type. The other three columns in Figure 2 were all created by selecting *Sheet | Fill Cells | Fill Random Number* in the main menu of Calc, which opens the pop-up window shown in Figure 3. The *Fill Cells* submenu has many more options, but here I selected the random generator of *Uniform Integers*, each time with equally random minimum and maximum values.

Day	Desktop	Server	Container
1	65	288	30
2	196	233	346
3	173	300	308
4	105	70	185
5	157	221	91
6	195	198	233
7	25	85	128
8	81	315	35
9	108	246	117
10	167	229	35
11	146	95	367
12	116	295	55
13	162	243	374
14	72	97	237
15	97	140	186
16	28	296	394
17	28	146	123

Figure 2: A table with labels and data can be quickly created with Calc's *Fill Cell* functions.

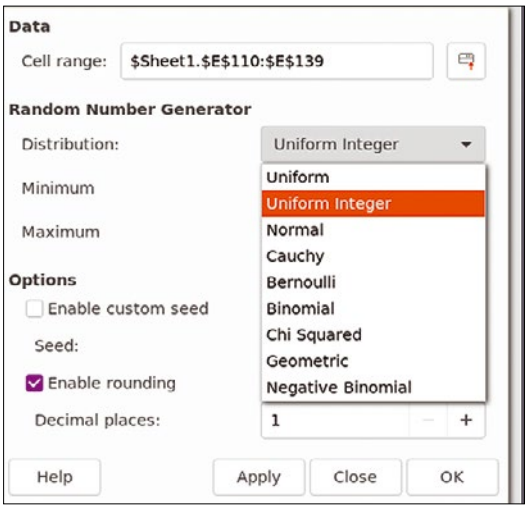


Figure 3: The Random Number Generator in LibreOffice Calc is used to filled the table in Figure 2.

Step-by-Step Chart Creation

Once the data is available, you can begin creating a chart: Select all the cells you want to visualize, headers included, and choose *Insert | Chart* in the main menu. This starts the four-step chart wizard shown in Figures 4 through 7.

In the first panel (Figure 4), you choose which kind of chart you want and set all of the options available for it. While it only takes a few clicks, this may very well be the most difficult step to get right.

One reason for this is that the right chart type to use depends heavily on both the actual nature of your data and on what, exactly, you want to see, learn, and tell others through that specific chart. It's not uncommon to create multiple, wildly different charts from the very same set of numbers, each illustrating one distinct issue.

The other reason is that creating a chart can be harder than necessary if the columns and rows of data aren't already all close to each other and in the right order when you first realize you may want to use them in a chart.

Never rush to the chart wizard when you need a chart. Instead, first draw a sketch of what your chart should look like on paper, trying different types until you are happy with the result. Then return to your spreadsheet. If data isn't already grouped in the best way for drawing the chart you want, rearrange it accordingly.

You should, for example, recognize which data is actual raw numbers to plot and which data isn't really numbers (as far as the chart is concerned, at least), but labels for data categories. One example of the last kind is a column that contains years of birth, which in the chart will be used only to classify and distinguish different age classes. If rearranging is not possible, however, don't despair. I will show how to deal with such cases in a moment.

Correct grouping of data in the spreadsheet makes chart creation faster but may still tell little or nothing about which chart type should be chosen. The chart type to use depends on what kind of relationship there is between the data.

To understand this, consider a table with just two columns. If the relationship between the data in the two columns is biunivocal – that is (real mathematicians, forgive me for this approximation!), if each value in the first column corresponds to one and only one unique value in the second column (for example, the maximum temperature on each day of the current year) – then a line chart may very well be the best choice. If, instead, there is more than one value on the y-axis for any given x-axis value, a scatter graph would probably be much better.

This said, I strongly suggest only playing with line charts until you have become familiar with all the settings of the chart wizard. Afterwards, it will be much easier to master the other chart types and, above all, to make charts that communicate well, rather than merely trying to impress friends or colleagues.

The second and third panels of the wizard are where you configure the *Data Range* (Figure 5) and the *Data Series* (Figure 6), respectively, of your chart. The first term indicates the complete set of cells that contains *all* the data to plot or use as labels in the chart. A data series, instead, is one

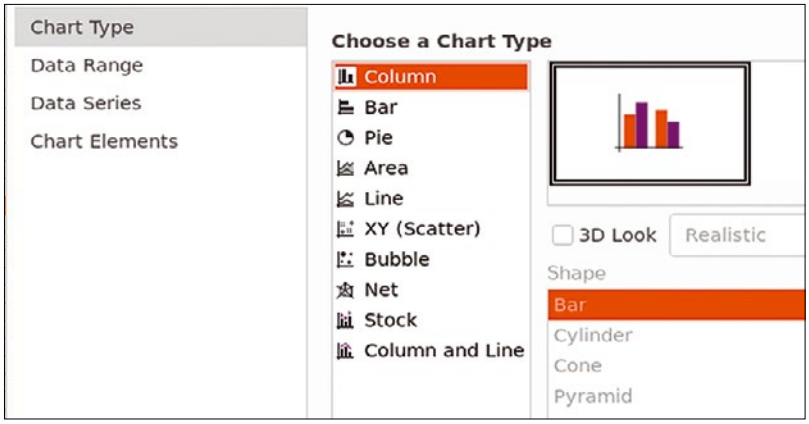


Figure 4: Choosing a chart type looks easy, but it may be the hardest part of the chart creation process.

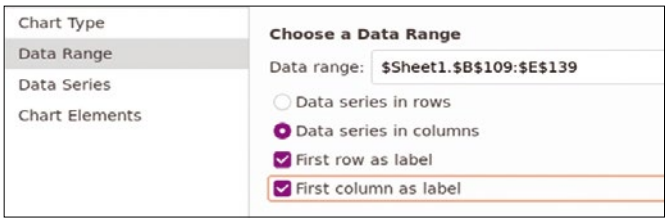


Figure 5: Data ranges are easy to load in the chart wizard, if they were carefully prepared beforehand!

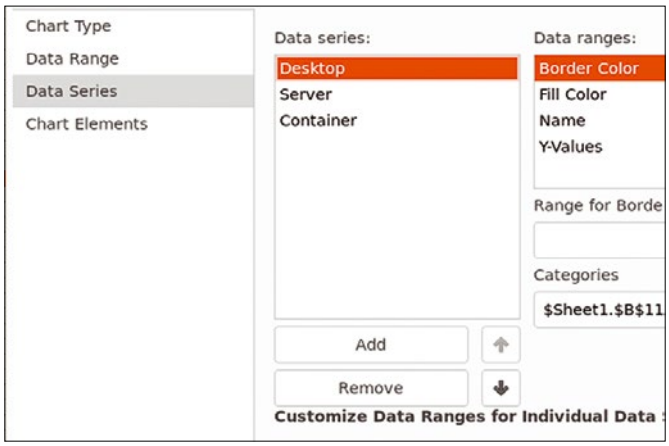


Figure 6: If the data range was prepared properly, the chart wizard recognizes both the data and labels by itself.

single subset of data inside that range, which generates one separate subplot in the chart.

By default, the data range loaded by the wizard will correspond to all the cells you selected with the mouse before launching the wizard itself. In my example, those cells constitute the rectangle of 31 rows by 4 columns *partially* shown in Figure 2. The wizard identifies that rectangle by listing its top left and bottom right corners, each in the *$SHEET$COLUMN$ROW* format, joined by a colon.

The data range shown in Figure 5 means "everything from cell B109 of Sheet 1 to cell E139 of the same sheet," which, as you can quickly check for yourself, is exactly 31 rows by 4 columns. What really matters here is that you can define the data range as you want by writing it yourself in the *Data range* field in Figure 5, or by clicking on the button to its right (not shown in Figure 5), to select it with the mouse.

Even better, you can group non-contiguous "sub-ranges" (e.g., two separate areas of the spreadsheet) by entering their coordinates in the format above, one at a time. The sub-ranges must be separated by Calc's "appropriate delimiter," the default value of which in almost all locales is a comma. To check or change your range delimiter, go to *Tools | Options | LibreOffice Calc | Formula*, and look in the *Array Column* field.

The other, equally important operation to do in the *Data Range* panel is to specify whether the first row or column of the whole data range should be used in the chart as labels, or independent coordinates, for the other data. If you don't specify, Calc will treat the first row or column as just one more set to plot alongside all others.

Figure 5 shows what happens if you select the data range of Figure 2, with the first row and column each marked as labels for the other data. This makes the wizard recognize the leftmost column (i.e., cells from B109 to B139) as a "category" that must not be plotted by itself but used as the independent coordinate on the x-axis. For the other three columns, marking the top row as labels assigns the names in each of the top cells to, respectively, the columns' three distinct data series.

Figure 6 contains several other important pieces of information. One is that in Calc (confusingly, if you ask me) the numbers inside each data series are called, again, the data range of that series. The good part is that you may enter the actual range of any series in the same way as the whole data range, by typing it in the corresponding text field.

Another important element hidden in plain sight in Figure 6 becomes evident when you compare it with Figure 4: Each data series has different configuration options that depend on the chart type selected in the first panel. In Figure 6, you can set *Border Color*, *Fill Color*, *Name*, and the actual data (*Y-Values*) for each series, because those are the options available for the *Column* chart type that was set in Figure 4. Had I chosen *Line*, instead, Figure 6 would show only the parameters that make sense for a line diagram – the *Y-Values* to plot and the *Name* of the resulting line.

Last but not least, note the *Add*, *Remove*, and arrow buttons in Figure 6. Yes, you can change the order in which the series is placed in the chart, *without* reshuffling the corresponding rows or columns in the spreadsheet. You can also remove whole series (again, just from the chart, *not* from the spreadsheet!), or add entirely new ones, by entering their range as already explained.

The last panel of the wizard (Figure 7) is probably the easiest. This is where you set the chart's descriptive "metadata," that is, title, subtitle, axes labels, and legend. The final result of all this work is shown in Figure 8. The x-coordinates, what Calc calls "category," are unreadable, but no problem! In Calc, you can always double-click on an existing chart to enter an "Edit Mode" in which you can select any element

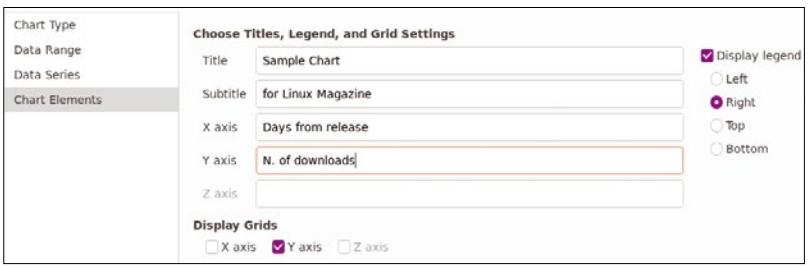


Figure 7: The last panel of the chart wizard lets you enter titles and select legend and grid settings.

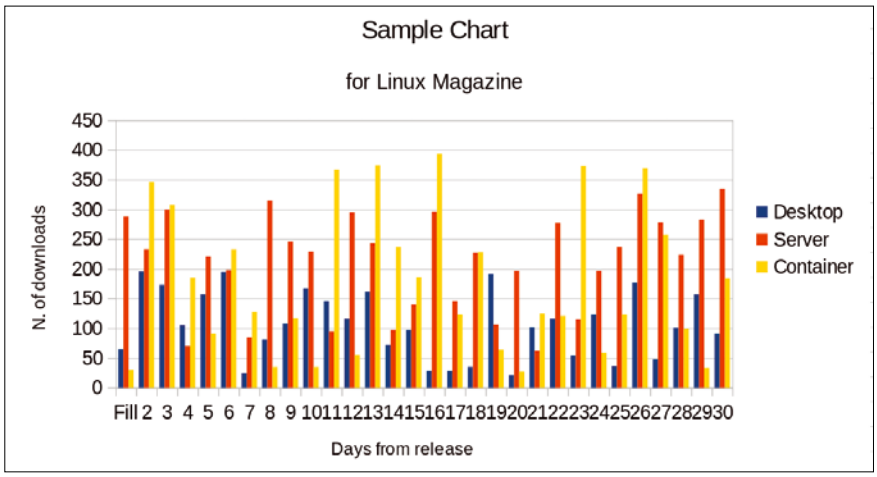


Figure 8: The histogram resulting from the data and settings shown in the previous figures still needs some work.

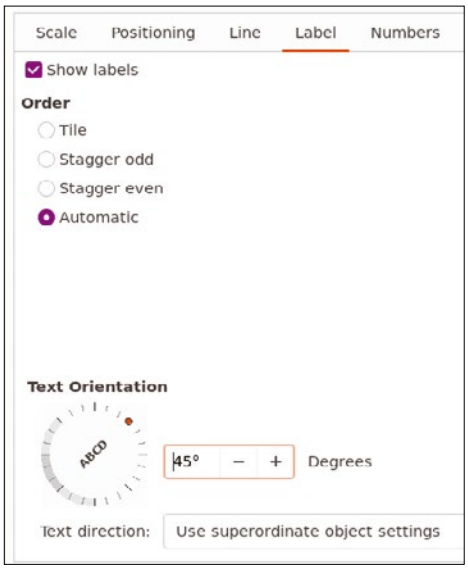


Figure 9: If the labels on the axes are too crowded, rotate them!

and then right-click on it (or choose *Format* in the top menu) to reopen the corresponding section of the wizard. Doing just that on the x-axis opened the panel in Figure 9, in which I rotated the text to make it readable, as shown in Figure 10!

Before continuing, let's go back to Figure 8 to learn a little more chart terminology. The main rectangle that contains everything else is the chart wall, and whatever you put as background is the chart area. Three-dimensional (3D) charts will also have a chart floor. You may load any image you want for the chart area, including photographs, though this is almost never a good idea. Unless you have a really good reason to do it in the first place and choose very carefully both the image and the colors of the diagram, the result will often be much harder to read. The same consideration applies to many of the most esoteric options of Calc charts. More often than not, resisting the urge to use them will make your chart better.

That said, there are other Calc functions that are important to know because they may make many of your charts better. The first of such functions is a must whenever you need to show, within the same chart, two data series that have different units or very different scales. What if, for example, one series varied between 0 and 1,000 and the other only between 0 and 30? Plotting them with just one vertical axis would make the second series almost invisible!

The solution, applicable only after the chart has been created, is to double-click on the chart, click on one of the two series that appears, and then go to the *Options* tab shown in Figure 11. Then, to align that series, select *Secondary Y axis* to obtain a result

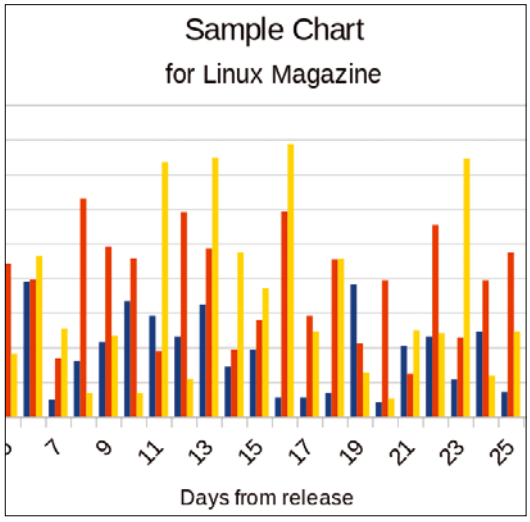


Figure 10: The same labels seen in Figure 8 are much more readable now.

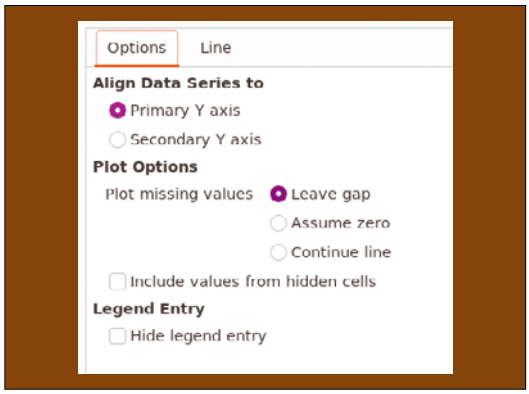


Figure 11: Another option that makes charts much more readable is the addition of a secondary y-axis.

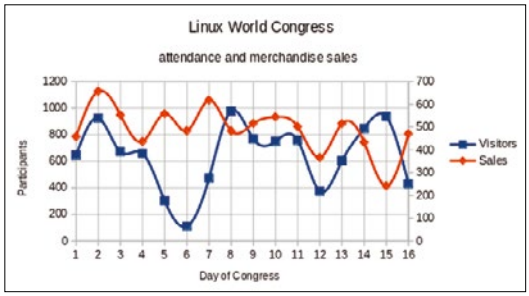


Figure 12: The extra y-axis on the right gives both lines a comparable scale, thus making both of them more readable.

Figure 13: A welcome addition in LibreOffice Calc 7.2 is more options for the moving average trend line.

like shown in Figure 12, which shows two lines in full detail thanks to the fact that each of them has its own axis adapted to its range.

Another area that can greatly increase the clarity of a chart is coloring, a term I use here to indicate three distinct options: To begin with, you can change the color that Calc assigns to each data series by selecting the color inside the chart, right-clicking on it, and choosing *Format Data Series*. To change the default colors for all your charts, instead, go to *Tools | Options | Charts | Default Colors*. The most interesting coloring you might do in LibreOffice charts, however, is based on conditional formatting. This means, for example, giving each bar in a histogram a different color depending on whether the corresponding numeric value meets certain conditions. For a very detailed explanation of this technique, see the "Creating Charts with Conditional Formatting" tutorial [3].

LibreOffice v7.2 has improved a function that, while useless for many users, is a must-have for users who do statistical analyses. The panel shown in Figure 13 – which opens when you select a data series in the chart and click on *Insert Trend Line* – now supports more types of moving average (Figure 14).

The last function for making great Calc charts (like using pictures for chart walls) is very easy to misuse: 3D charts. The readability of a 3D chart depends a lot on proper lighting and orientation, and Calc makes it very easy to optimize these. Once the 3D chart has been created, double-click on it and select *Format | 3D View* to open an interactive panel with three very intuitive tabs in which you can easily try and set different values for those parameters (Figure 15).

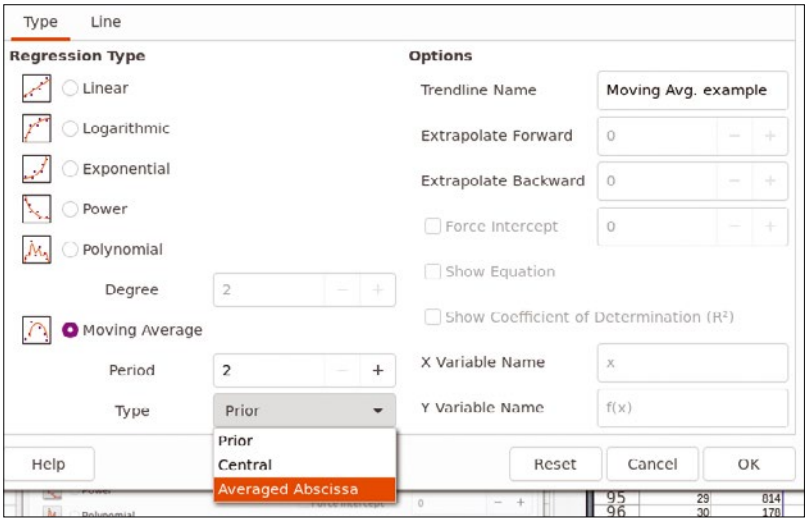


Figure 14: A moving average trend line (blue) has been added to the chart by using the options shown in Figure 13.

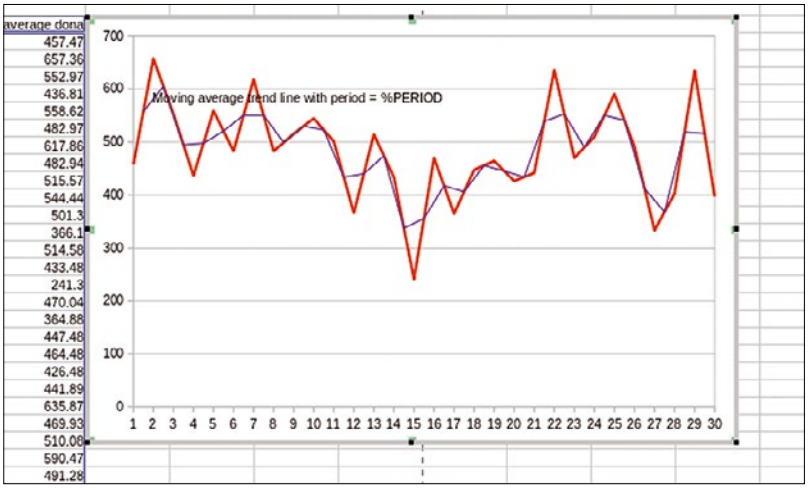
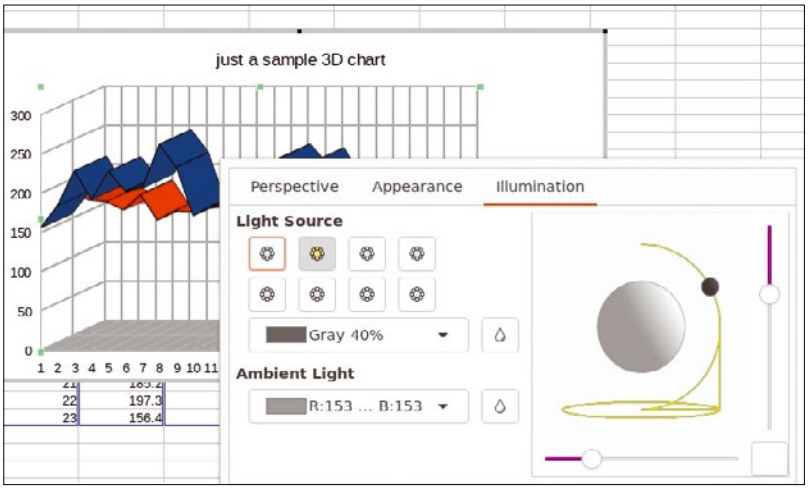


Figure 15: 3D charts can be great, but only if they have the right lighting, orientation, and colors.



Next Steps

This tutorial contains all the basic definitions and concepts you need to get started as a serious chart designer with LibreOffice 7.2 or later. If you make sure you really understand the foundation explained here, then practicing Calc's many charting options will be much faster and easier than if you had started from scratch. Happy charting! ■■■

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 blogs about digital rights at *https://stop.zona-m.net*.

Info

[1] LibreOffice Calc Guide, v7.2: *https://wiki.documentfoundation.org/images/2/2b/CG72-CalcGuide.pdf*

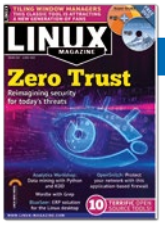
[2] Graphs vs. Charts: *https://www.wallstreetmojo.com/graphs-vs-charts/*

[3] Creating Charts with Conditional Formatting: *https://www.ryananddebi.com/2019/02/02/libreoffice-calc-creating-charts-with-conditional-formatting/*

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.



#259/June 2022

Zero Trust

Twenty Years ago, everyone thought a gateway firewall was all you needed to stay safe from intruders, but recent history has told a different story. Today, the best advice is: Don't trust anyone. Your internal network could be just as dangerous as the Internet.

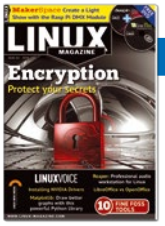
On the DVD: Zorin OS 16.1 Core and Super GRUB2 Disk

#258/May 2022

Clean IT

Most people know you can save energy by changing to more efficient light bulbs, but did you know you can save energy with more efficient software? This month we examine the ongoing efforts to bring sustainability to the IT industry.

On the DVD: Manjaro 21.2 Qonos and DragonFly BSD 6.2.1



#257/April 2022

Encryption

This month, we survey the state of encryption in Linux. We look beyond the basics to explore some of the tools and technologies that underpin the system of secrecy – and we show you what you need to know to ensure your privacy is airtight.

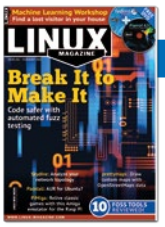
On the DVD: Linux Mint 20.3 Cinnamon Edition and deepin 20.4

#256/March 2022

Facial Recognition

Biometrics got a boost recently with the arrival of Microsoft's Hello technology. Now the open source world is catching up, with an innovative tool appropriately called Howdy. Facial authentication might not be ready for the CIA yet, but we'll help you get started with Howdy and explore the possibilities of authenticating with a glance.

On the DVD: antiX 21 and Haiku R1/ Beta 3



#255/February 2022

Break It to Make It

Fuzz Testing: Ever wonder how attackers discover those "carefully crafted input strings" that crash programs and surrender control? Welcome to the world of fuzz testing. We introduce you to the art of fuzzing and explore some leading fuzz testing techniques.

On the DVD: Parrot OS 4.11 and Fedora Workstation 35

#254/January 2022

Phone Hacks

Eventually phone manufactures just give up on supporting old hardware. If you're not ready to abandon that hardware yourself, you might find a better alternative with LineageOS — a free Android-based system that supports more than 300 phones, including many legacy models that are no longer supported by the vendor. We also explore PostmarketOS, a community-based Linux distribution that runs on several Android devices.

On the DVD: Ubuntu 21.10 and EndeavourOS 2021.08.27

FEATURED EVENTS

Users, developers, and vendors meet at Linux events around the world. We at *Linux Magazine* are proud to sponsor the Featured Events shown here.

For other events near you, check our extensive events calendar online at *https://www.linux-magazine.com/events*.

If you know of another Linux event you would like us to add to our calendar, please send a message with all the details to *info@linux-magazine.com*.

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.

GUADEC 2022

Date: July 20-25, 2022

Location: Guadalajara, Mexico

Website: *https://oscal.openlabs.cc/*

This year marks the 25th anniversary of GNOME. Join us in Guadalajara, Mexico for GUADEC, the GNOME community's largest conference. Celebrate GNOME and free software as we bring together hundreds of users, contributors, community members, and enthusiastic supporters for a week of talks and workshops.

SCaLE 19X

Date: July 28-31, 2022

Location: Los Angeles, California

Website: *https://www.socallinuxexpo.org/scale/19x*

SCaLE is the largest community-run open source and free software conference in North America. It is held annually in the greater Los Angeles area. Join us for four days of sessions, tutorials, and special events.

Events

Storage Developer Conference (SDC EMEA)	June 14	Virtual Conference	https://www.snia.org/events/sdcemea
ODSC Europe 2022	June 15-16	London, UK + Virtual	https://odsc.com/europe/
openSUSE Summit at OSCAL 2022	June 18-19	Tirana, Albania	https://events.opensuse.org/conferences/oSO22
OSCAL 2022	June 18-19	Tirana, Albania	https://oscal.openlabs.cc/
CIO/CISO DACH Summit	June 21	Frankfurt, Germany	https://cisodachsummit.com/
ITEXPO Florida	June 21-24	Fort Lauderdale, Florida	https://www.itexpo.com/east/
Open Source Summit North America	June 21-24	Austin, Texas + Virtual	https://events.linuxfoundation.org/
CDO UK Summit	June 23	London, United Kingdom	https://cdosummit.uk/
Linux Security Summit North America	June 23-24	Austin, Texas and Virtual	https://events.linuxfoundation.org/
Xen Developer & Design Summit	June 28-30	Bucharest, Romania + Virtual	https://events.linuxfoundation.org/xen-summit/
USENIX ATC '22 & OSDI '22	July 11-13	Carlsbad, California	https://www.usenix.org/conference/
The Open Source Infrastructure Conference	July 19-20	Berlin, Germany	https://stackconf.eu/
GUADEC 2022	July 20-25	Guadalajara, Mexico	https://events.gnome.org/event/77/
Icinga Camp Berlin 2022	July 21	Berlin, Germany	https://icinga.com/community/events/icinga-camp-berlin-2022/
SCaLE 19x	July 28-31	Los Angeles, California	https://www.socallinuxexpo.org/scale/19x
USENIX Security '22	August 10-12	Boston, Massachusetts	https://www.usenix.org/conference/usenixsecurity22

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	16, 46
Zack Brown	12
Bruce Byfield	6, 20, 26, 43
Joe Casad	3, 20
Mark Crutch	69
Markus Feilner	48
Marco Fioretti	90
Karsten Günther	72
Jon "maddog" Hall	71
Rubén Llorente	38
Martin Gerhard Loschwitz	28, 48
Vincent Mealing	69
Goran Mladenovic	32
Martin Mohr	60
Graham Morrison	84
Dmitri Popov	52
Thorsten Scherf	30
Mike Schilli	56
Scott Sumner	64
Ferdinand Thommes	78
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
© Pavlo Syvak, 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

Represented in Europe and other territories by: Sparkhaus Media GmbH, Bialasstr. 1a, 85625 Glonn, Germany.

Published monthly as Linux Pro Magazine (ISSN 1752-9050) for the USA and Canada and Linux Magazine (ISSN 1471-5678) for Europe and other territories 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.

Approximate On Sale Date	
UK / Europe	Jul 02
USA / Canada	Jul 29
Australia	Aug 29

Please note: On sale dates are approximate and may be delayed because of logistical issues.

Issue 261 / August 2022

Boot Drive Tools

A bootable CD won't take you far in world where most new computers don't even have a CD drive. Live boot now means booting from a USB stick, and while users have gotten more mobile, the tools have gotten more sophisticated. Carry your OS with you and plug it in – you can even put multiple ISOs on a single stick. Next month we explore a new generation of utilities for USB boot.

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***

The GNOME™ Conference

GUADEC

Guadalajara, Mexico
July 20–25, 2022

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

Join us at GUADEC 2022, in Guadalajara or online, to hear about the latest technical developments, attend talks, participate in workshops, and celebrate the 25th anniversary of GNOME!

Learn More

GUADEC.ORG

HETZNER

LOCATED IN THE USA

CLOUD SERVER

STARTING AT

$4.15

monthly

HETZNER CLOUD SERVER CPX11

- ✔ AMD EPYC™ 2nd Gen
- ✔ 2 vCPU
- ✔ 2 GB RAM
- ✔ 40 GB NVMe SSD
- ✔ 20 TB traffic inclusive
- ✔ Intuitive Cloud Console
- ✔ Located in Germany, Finland or USA

EPYC AMD

HIGH QUALITY - UNBEATABLE PRICES

DEPLOY YOUR HETZNER CLOUD IN UNDER **10 SECONDS!**

MANAGE YOUR CLOUD QUICK AND EASY WITH FEATURES LIKE

LOAD BALANCER, FIREWALLS, ONE CLICK APPS AND MANY MORE!

GET YOUR CLOUD NOW

CLOUD.HETZNER.COM