Garden Irrigation with a Raspberry Pi!

LINUX MAGAZINE

ISSUE 267 – FEBRUARY 2023

Backup

Stay safe and maximize convenience with the perfect backup tool

Xonsh: Shell programming in Python

Nala: Keep your packages in order with this appealing Apt front end

YunoHost: The easy way to manage a Linux server

Secrets of Natural Language Processing

10 FORMIDABLE FOSS TOOLS

LINUX NEW MEDIA
The Pulse of Open Source

WWW.LINUX-MAGAZINE.COM

CODE CARE

Dear Reader,

After all these years in publishing, it is really hard to grab my attention with a random headline, but I have to admit this one caught my eye: "Poor Software Quality Costs the US $2.4 Trillion." At first I thought it was a typo. (I was thinking maybe it should be "Billion" – I had no illusions that it was going to be "Million.") But with further reading, it appears that "Trillion" really was what they had in mind.

All the articles that appeared in the press with some variation of this headline pointed to a report by the Consortium for Information on Software Quality (CISQ) [1]. CISQ is an organization founded by the Object Management Group (OMG) standards development organization and the Software Engineering Institute at Carnegie Mellon University. Partners include security organizations and consultants, including MITRE and Gartner Group.

The report lists three main reasons for this astronomical loss:

- Cybercrime
- Software supply chain problems with underlying third-party components
- Technical debt

Cybercrime receives the most attention, but, according to the report, it is actually the *least* costly of the three categories. Much of the discussion of the supply chain problems points a finger at open source software. In this case, though, the argument doesn't seem to be so much about the fact that the code is open source as the fact that the same few free components are used so extensively. According to the report, "A medium-sized application (less than 1 million lines of code) carries 200 to 300 third-party components on average." Some of the most popular open source components are built into thousands of applications, so if a problem is found in one of these components, it is echoed thousands of times. Also, as the report points out, the free availability of open source components means the programmers oversee a smaller percentage of the code in the final application and a larger percent of the code comes from sources beyond the programmer's control. Many companies simply don't rise to the challenge of keeping up with all those moving parts. The report quotes a Black Duck study that found that 99 percent of audited codebases contained open source components, and 81 percent of those components were out of date.

The discussion of *technical debt* was perhaps the least familiar for the average user. Technical debt is a concept from the field of economics that doesn't receive a lot of attention in the technology industry. Wikipedia refers to technical debt as "…the implied cost of additional rework caused by choosing an easy (limited) solution now instead of using a better approach that would take longer." The idea is that, by applying a shortcut or quick fix to a problem (or by ignoring a problem, which also happens), you incur a debt that you will have to pay back in the future. The degree to which the problem will be more difficult to fix later is a form of "interest" on that debt.

Economists and venture capitalists need a concept like technical debt to estimate the value of a company. Suppose you have two companies that start with the same software. Company A has a team of developers continuously maintaining and updating the code, which is good development practice, but it also exacts a cost in overhead. Company B doesn't bother with maintaining this level of investment. They might have one coder applying security fixes, but the underlying problems accumulate. Company B minimizes overhead, and might even claim to have a greater "profit" for the year, but all they are really doing is deferring the expense to a later date. Technical debt is a measure of those deferred expenses.

You might be wondering why all this matters. Numbers like $2.4 trillion are intended to shock. Doesn't every industry have to plan ahead for some form of waste? Perhaps, but you can also bet that every successful company is focused on how to keep that number as low as possible. Some of the solutions offered in the report are remedies you've heard about for years: better quality standards and better tools for finding deficiencies. One of solutions is relatively new for this kind of practical discussion, though it is an ongoing dream of futurists: expanded use of AI and machine learning for software development.

Cybercrime and security issues dominate the news cycle. Other code quality concerns are discussed within academia, and to some extent, within the programming profession, but they receive far less attention in the press. The real value of documents like the CISQ report is to bring these issues to the surface. Perhaps the open source community really does need to spend a little more time considering what else can be done to support those core components – including educating developers on keeping them up to date. And the next time you invest in a software company, don't just look at the revenue: Take some time to consider the company's commitment to their code.

Joe

Joe Casad,
Editor in Chief

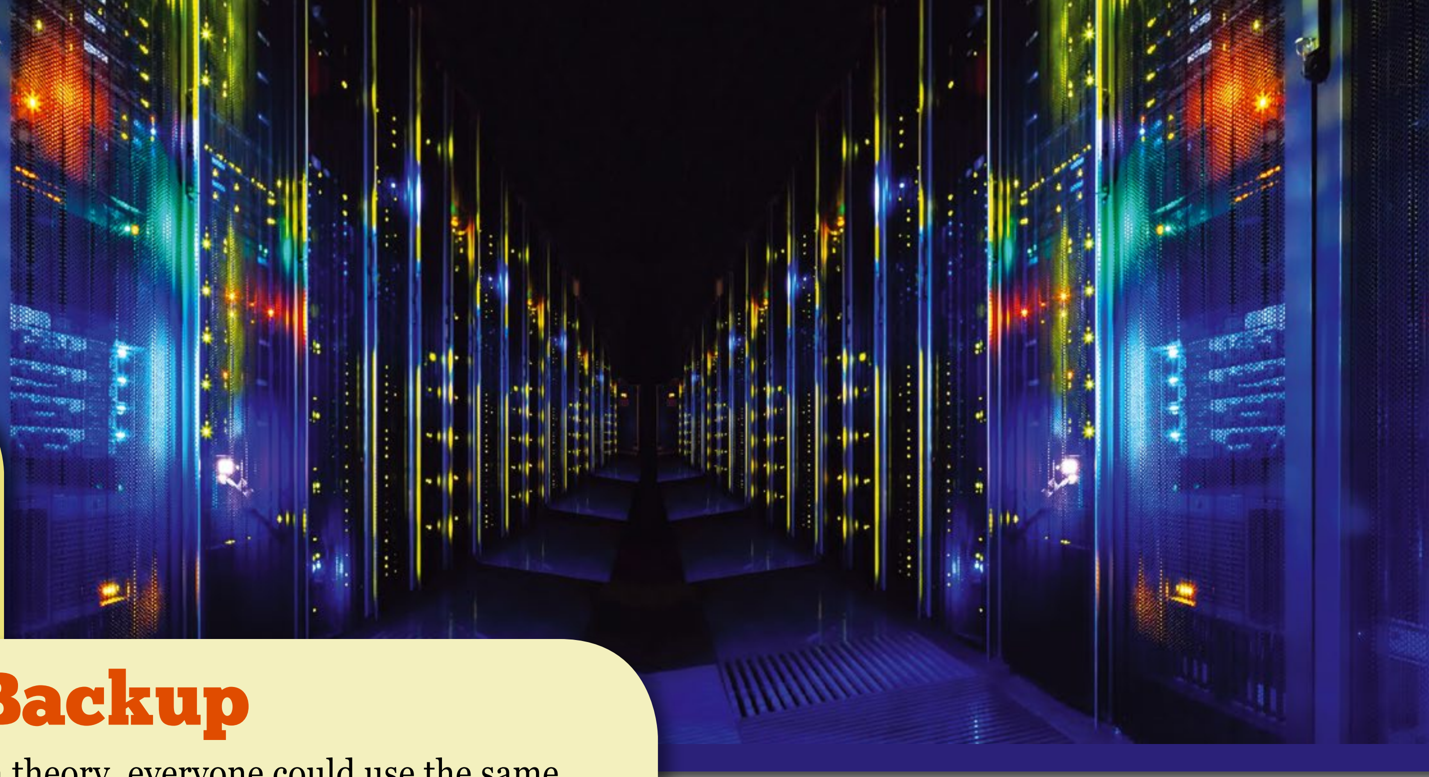
Info

[1] The Cost of Poor Software Quality in the US: A 2022 Report: *https://www.it-cisq.org/the-cost-of-poor-quality-software-in-the-us-a-2022-report/* (requires registration)

LINUX MAGAZINE

FEBRUARY 2023

ON THE COVER



Backup

In theory, everyone could use the same backup tool, but the best way to build regular and systematic backups into your life is to find a solution that fits with your own habits and methods. This month, we preview some popular backup apps in the Linux space so you can find one that works for you.

@linux_pro
@linuxpromagazine
Linux Magazine
@linuxmagazine

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

Linux Mint 21 Cinnamon and Kali Linux 2022.4

Two Terrific Distros on a Double-Sided DVD!

Linux Mint 21 Cinnamon
64-bit

Linux Mint needs no introduction for many users. Based on long-term support (LTS) editions of Ubuntu, Linux Mint is one of the top three Debian derivatives, known for its innovation and its close consultation with users. Cinnamon is the desktop developed by Mint, as opposed to MATE, Linux Mint's Gnome 2 fork.

The latest release is Linux Mint 21, codenamed Vanessa. Its version of Cinnamon includes enhanced Bluetooth support, with the replacement of Blueberry with Blueman; the addition of thumbnails for more graphic formats; and a process manager to help keep track of processes such as updates and system snapshots running in the background. Minor tweaks are also made to Sticky Notes and Timeshift, which takes snapshots of the installed system for easy backups. The major change in this release is that Muffin, Mint's version of the Mutter window manager, has been updated to bring it closer to Mutter after over a decade of independent development. The results include improved anti-aliasing and animation on the desktop.

As usual, Linux Mint's Cinnamon edition offers a user-friendly experience for all levels of users. News users can quickly learn the desktop, and veterans can enjoy one of the most stable, feature-rich environments available on Linux.

Kali Linux 2022.4
64-bit

Developed by Offensive Security, Kali is a leading forensic distribution. A revisioning of Offensive Security's BackTrack, Kali is primarily based on Debian Testing, although Offensive Security also provides packages for the tools it develops.

The latest version of Kali includes over 600 security tools, including:

- Armitage, a graphical cyber attack management tool
- Nmap, a port scanner
- Wireshark, a packet analyzer
- Metasploit, a penetration testing framework
- John the Ripper, a password cracker
- Sqlmap, an automatic SQL injection and database takeover tool
- Aircrack-ng, a software suite for penetration testing wireless LANs
- Burp Suite, an integrated security testing platform for web apps
- OWASP Zed Attack Proxy (ZAP), a penetration tool for web apps

Kali is aimed specifically at expert users. However, for all users, Kali Linux remains a major one-stop distribution for security troubleshooting.

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

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

March 9-12, 2023 | Pasadena, CA

SCaLE

Southern California Linux Expo

Aerospace | Cloud Native | Developer | Embedded | Home Automation | IoT
Kernel | Low Level Systems | Observability | Open Data | Open Government
Open Medical | Security | Systems and Infrastructure | HAM Radio and Linux

Don't miss the North America's largest annual community organized Open Source gathering.

Register Now!

Use code ***LM50*** for an exclusive

50% OFF

socallinuxexpo.org

NEWS

Updates on technologies, trends, and tools

THIS MONTH'S NEWS

Initial Support for Rust in the Linux Kernel is Finally Released

Linus Torvalds finally announced that the latest LTS version of the Linux kernel (version 6.1) includes the initial support for the Rust programming language that recently has been hyped (*https://lkml.org/lkml/2022/12/11/206*). This, of course, comes on the heels of what Torvalds calls the "merge window from hell," thanks to the holidays and his pre-holiday travel.

On this, Torvalds announced he would be very strict with the merge window rules, saying "The rules are that the pull requests sent to me during the merge window should have been ready _before_ the merge window and have seen some time in linux-next. No last-minute batch of experimental new development that hasn't been seen by our test automation."

Of course, the big news for Linux 6.1 is the inclusion of experimental Rust support. It's important to know that this is very much still in the early phase of development, so the implications aren't nearly as game-changing as you might think. This is the very basic implementation of Rust in the kernel, so don't expect to find new and improved Rust-built drivers functioning within the kernel.

Much of the other work found in kernel 6.1 is dominated by drivers for media, Bluetooth, HID, GPU, and, more importantly, networking. Make sure to read the announcement to find out everything that's included with the latest LTS kernel, which should serve as a very important launch point for the addition of Rust drivers for Linux.

Linux Mint 21.1 Enters Beta Status

The developers of Linux Mint have officially announced the upcoming release is in beta (*https://blog.linuxmint.com/?p=4442*). What's big about this news is that 21.1 will include the latest release of the Cinnamon desktop.

The new version of Cinnamon has been tidied up to look cleaner and more modern. You'll find more vibrant colors, a new mouse pointer, new system sounds, and some of the desktop icons have been removed and are now pinned to the panel or opened from the main menu.

In order to achieve more vibrant colors, the developers had to use fewer accents across the user interface (UI). This change also required a number of other minor theme tweaks.

The Linux Mint Driver Manager also received a number of improvements for the user interface and includes a more robust driver installation. Even drivers for Broadcom wireless devices are easier than ever.

Other improvements include Flatpak integration into the Update Manager, a refreshed UI for the Software Manager, an easier way to verify ISO images (via a right-click menu in the file manager), and numerous improvements to the Nemo file manager. Linux Mint 21.1 features Linux kernel 5.15, Cinnamon 5.6, and is based on Ubuntu 22.04.

Learn more about the upcoming release (codenamed Vera) in the official release notes (*https://www.linuxmint.com/rel_vera_cinnamon_whatsnew.php*).

4MLinux 41.0 Now Stable and Ready for Use

4MLinux 41 is now available for general use and includes plenty of updates. There are new applications to be had, including FileZilla, XPaint, and GNU Paint, a command-line tool for managing NVM Express (NVMe) partitions, a small collection of games, LibreOffice 7.4.3, Gnome Office (AbiWord, Gimp, Gnumeric), Dropbox, Firefox, Chromium, Thunderbird, the Audacious music player, VLC, SMPlayer, Wine 7.18, and more.

With the release of 4MLinux 41.0, it is now possible to install on a Btrfs partition, with the help of Syslinux acting as a boot manager.

This lightweight Linux distribution ships with kernel 6.0.6 and Mesa 22.1.4, and uses Joe's Window Manager (JWM) as the desktop interface.

There are three different versions that can be downloaded and used: Full, Core, and Server. The Server edition makes it easy to set up a full LAMP stack with Apache 2.4.54, MariaDB 10.6.11, PHP 5.6.40/7.4.33, Python 2.7.18/3.10.6, Perl 5.36.0, and Ruby 3.1.2.

Download your copy of 4MLinux 41.0 desktop (*https://downloads.sourceforge.net/project/linux4m/41.0/livecd/4MLinux-41.0-64bit.iso*) or Server (*https://downloads.sourceforge.net/project/linux4m/41.0/livecd/4MServer-41.0-64bit.iso*) editions now and read the full list of installed packages here (*http://4mlinux.com/addons-41.0.txt*).

Xfce 4.18 Coming Soon and Offers Subtle Improvements

When the next version of Xfce is released, you might not be blown away by an array of new and game-changing features. Instead, what you'll find are plenty of subtle new features and fixes that make the open source desktop better than ever – all the while remaining very familiar, stable, and easy to use.

The new features include a number of improvements to the Thunar file manager – such as a new bookmark menu, recent sidebar entry, customizable keyboard shortcuts, option to show full directory path, recursive search, undo/redo options, a split view, support for drag & drop items in the view panels, and the ability to execute shell scripts from within a directory.

As far as the desktop, you'll find the panel length is now configured in pixels (as opposed to a percentage), a new *Keep panel above windows* option, more font options for the Xfce clock applet, and header bars that can be disabled in dialogs.

Also included are a new default multi-monitor behavior that can be configured before you attach a second display, fixes for move-to-monitor, and 1.25/1.75 scale ratios.

You can find out more – and learn how to get the first prerelease – from this official announcement from the team (*https://www.mail-archive.com/xfce-announce@xfce.org/msg00715.html*).

Orange Pi Board Has Arch-Based Linux Distribution in the Works

The developers of the Orange Pi board have made available four operating systems supported for their hardware: Orange Pi OS, Ubuntu, Debian, and Manjaro. Soon, they will be adding another distribution into the mix, one based on Arch Linux.

This version of Arch Linux will be user-friendly and highly compatible with open source drivers. Orange Pi OS (Arch) will ship with LibreOffice and will support most of the major Linux desktops, such as Gnome, KDE, and Xfce.

According to the developers, Orange Pi OS (Arch) (*http://www.orangepi.org/html/softWare/orangePiOS/arch.html*) will be easy to install, stable, highly secure, support multi-framework systems (such as x86, Arm, and RISC-V), and will support a wide number of applications (including Code::Blocks, Gnome-disks, Inkscape, Thunderbird, VLC, VS Code, NeoChat, Remmina, and more).

Orange Pi OS (Arch) will also include support for most of the popular multimedia forms, multicore CPUs, command-line and GUI installation, and will be privacy and security conscious.

For more information on Orange Pi OS (Arch), check the official site for the operating system (*http://www.orangepi.org/html/softWare/orangePiOS/arch.html*), where the team will continue to add updates and news as they happen.

Alpine Linux 3.17 Now Available to the General Public

The first Alpine Linux release in the 3.17 stable series is now available for download and finally adds Rust on all supported platforms. The distribution ships with either Gnome 43 or KDE Plasma 5.26 and enjoys all of the new features and fixes found in both of those desktop environments.

As for what's new in Alpine Linux itself, the list includes Bash 5.2, GCC 12, Kea 2.2, OpenSSL 3.0, Perl 5.36, PostgreSQL 15, Node.js 18.12, Ceph 17.2, Go 1.19, Rust 1.64, and .NET 7.0.100. OpenSSL also is available with the *openssl1.1-compat* package.

It should be noted that PHP 8.0 has been officially deprecated and ISC Kea was moved to the main repository for long time support, whereas ISC DHCP was moved to the community repository. With this move, users are now encouraged to make the switch from DHCPD to Kea.

You can read more about Alpine Linux 3.17 in the official release notes (*https://alpinelinux.org/posts/Alpine-3.17.0-released.html*) and download an ISO for installation from the download page (*https://alpinelinux.org/downloads/*) for the following architectures: 64-bit (x86_64), AArch64 (ARM64), ARMv7, 32-bit (x86), PowerPC 64-bit Little Endian (ppc64le), and IBM System Z (s390x).

The New StarFighter Linux Laptop Now Available for Preorder

Star Labs has been creating Linux laptops for some time now and recently they announced a new addition to their fleet of options, the StartFighter custom laptop (*https://us.starlabs.systems/pages/starfighter*). This beautiful piece of technology features a true matte display that uses a protective coating to defuse ambient light so colors can shine brighter. The display offers up to 3840x2400 4K resolution, a 16:10 aspect ratio, and 600cd/m² of brightness at a 165Hz refresh rate and 178 degrees of viewing.

Other features found on the StarFighter include a removable webcam with built-in storage, a kill switch to shut off wireless when needed, a backlit keyboard with media keys, international layouts, and LED indicators. The StarFighter also includes a haptic trackpad, and a plasma electrolytic oxidation coaching for a textured finish that is stronger than steel and fingerprint resistant.

You'll find WiFI 6E and Bluetooth 5.3, HDMI, USB-C/A, microSD, and an audio combo jack. The firmware on the StarFighter is the open-course coreboot and the battery is charged by a gallium nitride charger that provides 65 watts over USB-C.

As far as components, the base model comes with a 12th generation Intel i3 CPU but can be upgraded to an AMD Ryzen 7 (6800H) or an Intel i9. The StarFighter can support up to 64GB of RAM and the storage can range from 240GB to 2TB.

Configure and preorder your StarFighter laptop today (*https://us.starlabs.systems/products/starfighter?variant=43623053000958*). The base model starts at $1,509.30. Keep in mind, however, the laptops may not ship for four to five months from now.

Critical Escalation Vulnerability Found in the Linux Kernel

Red Hat added a new CVE code, listed as 2022-3977 (*https://access.redhat.com/security/cve/CVE-2022-3977*), which is described as a use-after-free flaw. A use-after flaw can occur when a program attempts to use memory that has been released.

CVE 2022-3977 resides in the Linux kernel Management Component Transport Protocol (MCTP). How this vulnerability works is after a user simultaneously calls `DROPTAG ioctl` at the same time a socket close occurs. When this happens, the vulnerability can then be used to elevate privileges all the way up to root.

This CVE has been listed as Moderate, with a CVSS v3 base score of 7.0 and the vulnerability was found in the most recent upstream Linux kernel.

It was the Venustech Active Defense Laboratory that originally reported the vulnerability, finding it in kernel v5.18.0 with the commit 63ed1aab3d40aa61aaa66819bdce9377ac7f40fa. Fortunately, with a recent commit, the vulnerability has been patched.

If you have a Linux machine running kernel 5.18, you should immediately run an upgrade to patch the kernel. Most major repositories have most likely added the patch to their standard repositories.

AlmaLinux 8.7 Now Available

The newest release of AlmaLinux, version 8.7, is now available to the general public. This release is a 1:1 binary-compatible replacement for Red Hat Enterprise Linux 8.7 and features plenty of changes and updates.

One of the big changes comes in the ability to build custom images with custom/boot mount-point partitions and sizes. In addition, the latest release includes important security updates, such as all Network Security Service (NSS) libraries have increased the minimum key size for all RSA operations from 128 to 1023 bits. Other updates include, *scap-security-guide* is now better aligned with Defense Information Systems Agency technical guides content; a new package, *xmlstarlet*, which can parse, transform, query, validate, and edit XML files; Ruby 3.1; Mercurial 6.2; and Node.js 18.

The kernel shipping with AlmaLinux 8.7 is 4.18.0-423.el8 and the operating system includes support for x86_64, AArch64, ppc64le, and s390x architectures.

This new release is ready for production and, according to Benny Vasquez, chair of the AlmaLinux OS Foundation board, "We aim to deliver the quality and timeliness end users require from the leading CentOS successor, and to provide a free and open, community-owned and governed, enterprise-grade Linux operating system."

Download a copy of AlmaLinux 8.7 (*https://mirrors.almalinux.org/isos.html*).

Zack's Kernel News

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

By Zack Brown

Vulnerabilities using a 32-Bit Kernel on a 64-Bit CPU

Naresh Kamboju of Linaro reported that the kernel was giving some serious Spectre and Retbleed security warnings on the Linaro test farm. He said they were using the i386 kernel on the 64-bit Intel Skylake CPU. Or at least, he posted some debugging output that identified the specific CPU.

Greg Kroah-Hartman remarked that this particular combination – a 32-bit kernel running on a 64-bit architecture – was sort of pathological. People should really use a faster and more efficient 64-bit kernel if their architecture supports it. Greg said this particular environment might not be something the kernel developers needed to care about. He suggested that Pawan Gupta from Intel might be the person to ask.

Meanwhile, Peter Zijlstra remarked, "Yeah, so far nobody cared to fix 32bit. If someone *realllllly* cares and wants to put the effort in I suppose I'll review the patches, but seriously, you shouldn't be running 32bit kernels on Skylake / Zen based systems, that's just silly."

Rawan did reply to Greg, saying, "Intel is not aware of production environments that use 32-bit mode on Skylake-gen CPUs. So this should not be a concern."

However, Adam Borowski gave the following evaluation of the plight of potentially many users:

"Alas, some people still run [32-bit kernels on 64-bit CPUs] because of not knowing any better. Until not so long ago, they were proposed with two install media, '32-bit' and '64-bit', but no explanation. Upgrades keep working, crossgrades are still only for the brave of the heart, and reinstalling might not appear to have a reason compelling enough. And for quite some tasks, halved word size (thus ~2/3 memory usage) can overcome register starvation and win benchmarks.

"Thus I wonder: perhaps such combinations we consider to be invalid should refuse to boot unless given a cmdline parameter?"

Jan Engelhardt wryly remarked, "So how many benchmarks does a 32-bit userspace with a 32-bit kernel win over 32-bit userspace with a 64-bit kernel?" To which Adam replied, "Likely none or almost none."

However, Adam went on to say:

"What we want is for people to run 64-bit kernel, there are no real issues with userland.

"Valid uses to run 32-bit kernel:

- *ancient hardware (so much more prevalent than m68k we support!; non-hobbyists should upgrade to reduce power costs)*
- *hardware to run that 100$k-1M ISA industrial control/medical imaging card (which, having ISA, is necessarily ancient too)*
- *us devs testing the above*

"Only the last case will have a modern CPU, thus requiring an explicit override won't hurt less educated users – while telling the latter to grab a 64-bit kernel if their hardware isn't ancient would have other benefits for them beside just vulnerabilities."

It's fascinating for me to see this consideration of which kernels to care about on which architectures. It's certainly true that some combinations don't make practical sense because an alternative would run much faster and safer. And it's also true that a lot of people are not necessarily aware of the ins and outs of all those choices and distinctions and aren't aware that their decision might affect the security of their system. To some extent user ignorance may not be the problem of kernel developers, and it may be that distribution maintainers would be more interested in helping guard against certain risky user errors. But this distinction and division of labor is a fascinating one for me, and I always enjoy watching how the developers come down on one side or the other of such questions.

Working Around Hardware Security Vulnerabilities

Recently on the Linux kernel mailing list, Thomas Gleixner remarked, "Back

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**.

in the good old spectre v2 days (2018) we decided to not use IBRS. In hindsight this might have been the wrong decision because it did not force people to come up with alternative approaches."

Indirect Branch Restricted Speculation (IBRS) is a hardware feature built into some Intel CPUs in response to some of the security vulnerabilities like Spectre that have been identified in recent years. As a hardware feature, IBRS transparently blocks the security exploits of some of those vulnerabilities, without needing to be "enabled" by the kernel.

But like all such attempts to bypass these tough security vulnerabilities, there are trade-offs: For example, one security solution might be to avoid a large swath of CPU features, which in turn means that those features must be implemented in the kernel itself, less efficiently. So it's never an obvious choice to simply accept the hardware security features Intel provides. The kernel developers must evaluate the pros and cons in each case.

For example, as Thomas expressed this time, "It was already discussed back then to try software based call depth accounting […] to avoid the insane overhead of IBRS."

Thomas proceeded to explore the specific aspects of IBRS overhead, including increased memory usage and accounting errors, which he said themselves could be exploited by an attacker.

He went on to say, "As IBRS is a performance horror show, Peter Zijstra and me revisited the call depth tracking approach and implemented it in a way which is hopefully more palatable and avoids the downsides of the original attempt. We both unsurprisingly hate the result with a passion."

Thomas described his and Peter's implementation approach, which was a nightmare-tentacled god of death involving somehow identifying code that was making certain calls, patching those calls to be different (all while the system was in full operation), and then doing various tasks before sending those calls back to where they thought they were going in the first place.

He added, "This does not need a new compiler and avoids almost all overhead for non-affected machines." But Thomas went on to say that in their solution, "The memory consumption is impressive. On a affected server with a Debian config this results in about 1.8MB call thunk memory and 2MB btree memory to keep track of the thunks. The call thunk memory is overcommitted due to the way how objtool collects symbols. This probably could be cut in half, but we need to allocate a 2MB region anyway."

Thomas went into some detail on the scholarly basis for how IBRS (and his and Peter's code) would defeat the various security exploits. Although he also added, "There is obviously no scientific proof that this will withstand future research progress, but all we can do right now is to speculate about that."

He posted a batch of performance benchmarks, showing that their software-based solution was at least never slower than Intel's hardware-based IBRS solution. Of course, one benefit of a software-based solution is that it can be revised and improved by developers, while hardware is forever. Thomas and Peter's work turned out to be a case in point, as the best was most definitely yet to come.

Thomas and David Laight discussed some of the technical details, including ways of improving efficiency and reducing overhead. In fact, part of the potential improvements also involved patches to the GNU C Compiler (GCC). Thomas summarized these changes as "Let the compiler add a 16 byte padding in front of each function entry point and put the call depth accounting there. That avoids calling out into the module area and reduces ITLB pressure."

At this point Linus Torvalds joined the discussion, saying:

"Ooh.

"I actually like this a lot better.

"Could we just say 'use this instead if you have SKL and care about the issue?'

*"I don't hate your module thunk trick, but this does seem *so* much simpler, and if it performs better anyway, it really does seem like the better approach.*

"And people and distros who care would have an easy time adding that simple compiler patch instead."

To which Thomas replied, "Yes, Peter and I came from avoiding a new compiler and the overhead for everyone when putting the padding into the code. We realized only when staring at the perf data that this padding in front of the function might be an acceptable solution. I did some more tests today on different machines with mitigations = off with kernels compiled with and without that padding. I couldn't find a single test case where the result was outside of the usual noise. But then my tests are definitely incomplete."

Peter joined the discussion at this point, as well, along with Joao Moreira and others, and the bunch of them tried to figure out the best form for the GCC patch, along with other implementation details for the rest of the fix. The discussion delved pretty deep into technical details and ranged all over the place, with pretty much everyone being more or less confused at least some of the time.

At one point, Tim Chen of Intel posted some performance benchmarks using the new "padding" approach from Thomas and Peter, and he said, "Padding improves performance significantly."

Linus looked at those numbers and replied, "That certainly looks oh-so-much better than those disgusting ibrs numbers."

The discussion ended roughly around there, with people seeming very enthusiastic about the prospect of a much more efficient approach to Retbleed security problems.

It's an ongoing process – the latest improvement is merely the new worst-case that everyone will love to improve upon. And it's amazing to know that in the open source world, we can watch the developers struggle in real time to protect everyone from attackers that themselves work night and day to crack into all of our systems.

When It's OK to Panic

David Hildenbrand submitted some security code updates that ran into resistance from Linus Torvalds. Among David's various changes was the use of the `VM_BUG_ON()` function. This and its various sibling functions check the state of the running system and bring it to a halt if that state seems to have become pathological. This seems sensible: If your system is horked, you may not want to blithely continue whatever it is you're doing.

However, in response to that bit of code, Linus replied, "STOP DOING THIS."

He went on to explain:

"Using BUG_ON() for debugging is simply not ok.

*"And saying 'but it's just a VM_BUG_ON()' does not change *anything*. At least Fedora enables that unconditionally for normal people, it is not some kind of 'only VM people do this'.*

"Really. BUG_ON() IS NOT FOR DEBUGGING.

"Stop it. Now.

"If you have a condition that must not happen, you either write that condition into the code, or – if you are convinced it cannot happen – you make it a WARN_ON_ONCE() so that people can report it to you.

"The BUG_ON() will just make the machine die.

"And for the facebooks and googles of the world, the WARN_ON() will be sufficient."

David, unflustered, replied, "I totally agree with BUG_ON … but if I get talked to in all-caps on a Thursday evening and feel like I just touched the forbidden fruit, I have to ask for details."

David continued:

*"VM_BUG_ON is only active with CONFIG_DEBUG_VM. … which indicated some kind of debugging at least to me. I *know* that Fedora enables it and I *know* that this will make Fedora crash.*

"I know why Fedora enables this debug option, but it somewhat destroys the whole purpose of VM_BUG_ON kind of nowadays?

"For this case, this condition will never trigger and I consider it much more a hint to the reader that we can rest assured that this condition holds. And on production systems, it will get optimized out.

"Should we forbid any new usage of VM_BUG_ON just like we mostly do with BUG_ON?"

Linus explained:

"VM_BUG_ON() has the exact same semantics as BUG_ON. It is literally no different, the only difference is 'we can make the code smaller because these are less important'.

"The only possible case where BUG_ON can validly be used is 'I have some fundamental data corruption and cannot possibly return an error'.

"This kind of 'I don't think this can happen' is _never_ an excuse for it.

"Honestly, 99% of our existing BUG_ON() ones are completely bogus, and leftover debug code that wasn't removed because they never triggered. I've several times considered just using a coccinelle script to remove every single BUG_ON() (and VM_BUG_ON()) as simply bogus. Because they are pure noise.

"I just tried to find a valid BUG_ON() that would make me go 'yeah, that's actually worth it', and couldn't really find one. Yeah, there are several ones in the scheduler that make me go 'ok, if that triggers, the machine is dead anyway', so in that sense there are certainly BUG_ON()s that don't _hurt_.

"But as a very good approximation, the rule is 'absolutely no new BUG_ON() calls _ever_'. Because I really cannot see a single case where 'proper error handling and WARN_ON_ONCE()' isn't the right thing.

"Now, that said, there is one very valid sub-form of BUG_ON(): BUILD_BUG_ON() is absolutely 100% fine."

Jason Gunthorpe remarked that he had heard people make the argument, "Since BUG_ON crashes the machine and Linus says that crashing the machine is bad, WARN_ON will also crash the machine if you set the panic_on_warn parameter, so it is also bad, thus we shouldn't use anything." Jason added that "I've generally maintained that people who set the panic_on_warn *want* these crashes, because that is the entire point of it. So we should use WARN_ON with an error recovery for 'can't happen' assertions like these."

To which Linus offered this fascinating explanation:

"If you set 'panic_on_warn' you get to keep both pieces when something breaks.

*"The thing is, there are people who *do* want to stop immediately when something goes wrong in the kernel.*

"Anybody doing large-scale virtualization presumably has all the infrastructure to get debug info out of the virtual environment.

"And people who run controlled loads in big server machine setups and have a MIS department to manage said machines typically also prefer for a machine to just crash over continuing.

"So in those situations, a dead machine is still a dead machine, but you get the information out, and panic_on_warn is fine, because panic and reboot is fine.

*"And yes, that's actually a fairly common case. Things like syzkaller etc *wants* to abort on the first warning, because that's kind of the point.*

"But while that kind of virtualized automation machinery is very very common, and is a big deal, it's by no means the only deal, and the most important thing to the point where nothing else matters.

*"And if you are *not* in a farm, and if you are *not* using virtualization, a dead machine is literally a useless brick. Nobody has serial lines on individual machines any more. In most cases, the hardware literally doesn't even exist any more.*

"So in that situation, you really cannot afford to take the approach of 'just kill the machine'. If you are on a laptop and are doing power management code, you generally cannot do that in a virtual environment, and you already have enough problems with suspend and resume being hard to debug, without people also going 'oh, let's just BUG_ON() and kill the machine'.

"Because the other side of that 'we have a lot of machine farms doing automated testing' is that those machine farms do not generally find a lot of the exciting cases.

"Almost every single merge window, I end up having to bisect and report an oops or a WARN_ON(), because I actually run on real hardware. And said problem was never seen in linux-next.

"So we have two very different cases: the 'virtual machine with good logging where a dead machine is fine' – use 'panic_on_warn'. And the actual real hardware with real drivers, running real loads by users.

"Both are valid. But the second case means that BUG_ON() is basically _never_ valid."

So there you have it. ■■■

FOSS Backstage
Community, Management & Compliance

20 23 BACKSTAGE FOSS

Join us for two exciting days on all things **governance, collaboration, InnerSource & OSPOs, project leadership, community management, legal** and **economic aspects** of open source software.

GET YOUR TICKETS NOW & USE CODE "LINUXMAG10"

2
days of engaging talks & discussions

40+
experienced speakers

Lots
of possibilities to network & meet great people

150+
online attendees

5
tracks

120+
onsite attendees

2
stages

23.foss-backstage.de

March 13-14, 2023
@ TUECHTIG Berlin & Online

Back up your data with BorgBackup and Vorta

Keep It Safe

BorgBackup and the Vorta graphical front end take the stress out of creating backups.

By Ferdinand Thommes

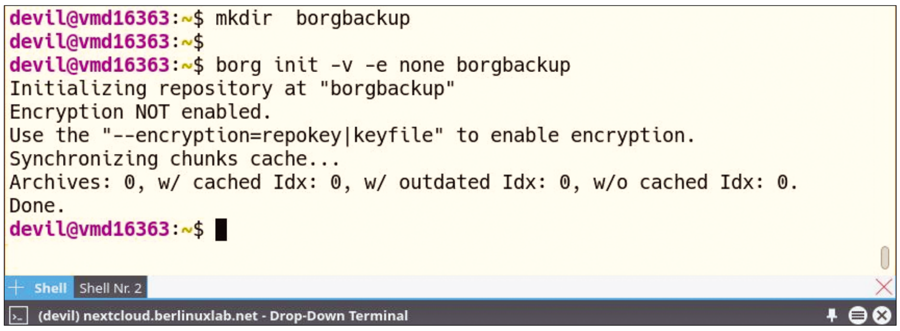
If you want to keep your data safe, it is a good idea to think about a backup strategy and implement it in a consistent way. The more important the data, the more security you need for the backup. Experts recommend that you should back up irreplaceable data twice, with one backup location outside of your own four walls.

Linux offers a variety of backup applications – for both home and work environments. Some are limited to the desktop; others are intended for backing up data on remote servers. One solution that works well in remote backup scenarios is the powerful BorgBackup [1] or Borg for short. Borg can store backups on local drives or remote computers in a space-saving and secure way. Thanks to the intuitive Vorta graphical interface, Borg is suitable for home users as well as professionals.

BorgBackup

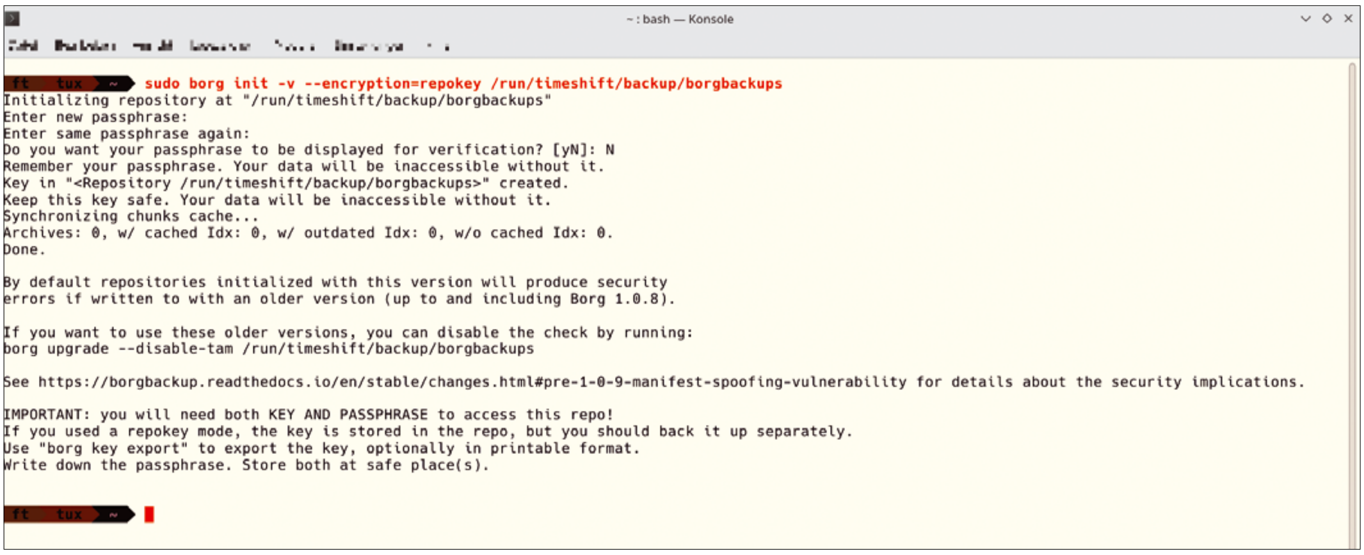
Borg runs at the command line, but you can also control it using the alternative Vorta graphical front end. Written in Python 3, Borg offers features such as deduplication, compression, and authenticated encryption. The data compression supports the LZ4, LZMA, Zlib, and Zstd standards. When it comes to *deduplication*, it is worth pausing to consider what the term actually means in Borg speak.

Deduplication is generally understood as a space-saving technique that breaks data down into blocks of the same size



```
devil@vmd16363:~$ mkdir  borgbackup
devil@vmd16363:~$
devil@vmd16363:~$ borg init -v -e none borgbackup
Initializing repository at "borgbackup"
Encryption NOT enabled.
Use the "--encryption=repokey|keyfile" to enable encryption.
Synchronizing chunks cache...
Archives: 0, w/ cached Idx: 0, w/ outdated Idx: 0, w/o cached Idx: 0.
Done.
devil@vmd16363:~$
```

Figure 1: When using servers connected via SSH, you need to be sure Borg is running on the server as well as the client.



```
ft  tux  ~  sudo borg init -v --encryption=repokey /run/timeshift/backup/borgbackups
Initializing repository at "/run/timeshift/backup/borgbackups"
Enter new passphrase:
Enter same passphrase again:
Do you want your passphrase to be displayed for verification? [yN]: N
Remember your passphrase. Your data will be inaccessible without it.
Key in "<Repository /run/timeshift/backup/borgbackups>" created.
Keep this key safe. Your data will be inaccessible without it.
Synchronizing chunks cache...
Archives: 0, w/ cached Idx: 0, w/ outdated Idx: 0, w/o cached Idx: 0.
Done.

By default repositories initialized with this version will produce security
errors if written to with an older version (up to and including Borg 1.0.8).

If you want to use these older versions, you can disable the check by running:
borg upgrade --disable-tam /run/timeshift/backup/borgbackups

See https://borgbackup.readthedocs.io/en/stable/changes.html#pre-1-0-9-manifest-spoofing-vulnerability for details about the security implications.

IMPORTANT: you will need both KEY AND PASSPHRASE to access this repo!
If you used a repokey mode, the key is stored in the repo, but you should back it up separately.
Use "borg key export" to export the key, optionally in printable format.
Write down the passphrase. Store both at safe place(s).


ft  tux  ~
```

Figure 2: When backing up to an external disk on the home network, Borg only needs to be installed on the source computer. Initializing the repositories is the same as it is for remote machines.

(which Borg refers to as "chunks") and computes a checksum for each block. If a new file has a known checksum, this means an identical version of the file already exists. At this point, the software compares the file contents to make sure they are identical. The app is not restricted to the file level, but it can also save minor changes to large files as chunks, which makes for very effective deduplication. The program even recognizes files if they have been moved or renamed.

Borg is available from the repositories of most mainstream Linux distributions; the package is called either *borg* or *borgbackup*. If you want to set up Borg for client-server operation, you need to install the software on all of the computers involved in the operation. If the Borg version packaged by your choice distribution is too old, you can also install from the source code, use a binary, or fire up `pip` or `git`. The Borg documentation describes the installation procedure in detail [2].

Deduplication

Comparing checksums, which is the preferred method for duplication, saves time over comparing the file contents in each block. The checksum method also has lower memory requirements because, if two identical blocks occur, it deletes the second block, replacing it with a pointer to the identical first block. This approach makes Borg a good candidate for use cases where working directories are constantly changing – you can back up every hour without generating large volumes of data.

Repositories and Archives

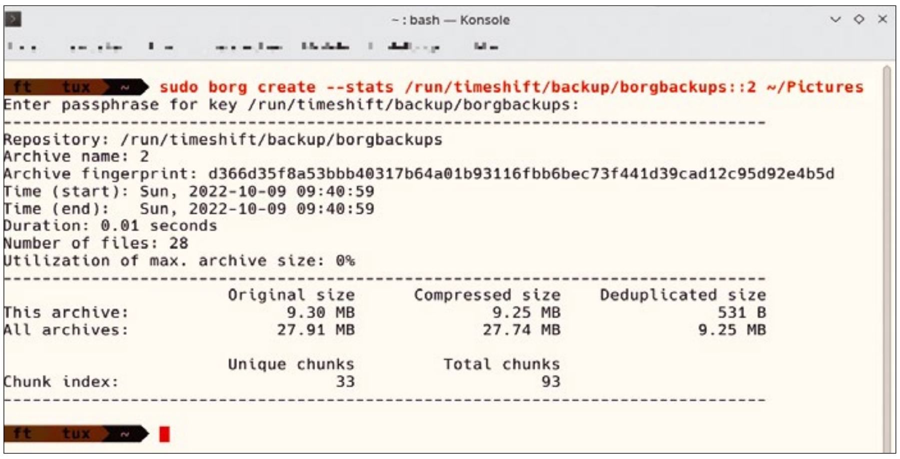
You can back up the directories on a laptop to an external hard disk or, alternatively, transfer the backup to a remote computer using SSH (Figure 1). Borg stores the backups it creates in repositories, and repositories are stored in folders (which Borg refers to as archives). So, you need to create and initialize at least one repository on the target computer or an external hard disk (Figure 2). The first three lines in Listing 1 create the repository.

If you use encryption with the `-e` option when creating the repository, the software prompts you for a password. (Initially, it is useful to use the `-v` option for verbose output.) Borg is then ready for a first backup (Listing 1, line 4). The `borg create` command (Figure 3) creates the backup in the specified path to the repository. The name of the archive where the app stores the backup is 1 in this example; the software automatically creates the matching folder.

The next command lists the folders to be backed up. The `--stats` option takes care of outputting statistics for the newly created archive, such as the volume of data, the deduplication level, and the number of chunks. The command in line 5 of Listing 1 displays all of the archives in the repository (Figure 4).

Listing 1: Preparing and Installing a Repository

```
01 $ cd /media/extern
02 $ sudo mkdir borgbackups
03 $ sudo borg init -v -e repokey media/extern/borgbackups
04 $ sudo borg create --stats /media/extern/backups::1 ~/Pictures ~/Documents
05 $ sudo borg list /media/extern/borgbackups
06 $ sudo borg extract /media/extern/borgbackups::1
```



```
ft  tux  ~  sudo borg create --stats /run/timeshift/backup/borgbackups::2 ~/Pictures
Enter passphrase for key /run/timeshift/backup/borgbackups:
------------------------------------------------------------------------------
Repository: /run/timeshift/backup/borgbackups
Archive name: 2
Archive fingerprint: d366d35f8a53bbb40317b64a01b93116fbb6bec73f441d39cad12c95d92e4b5d
Time (start): Sun, 2022-10-09 09:40:59
Time (end):   Sun, 2022-10-09 09:40:59
Duration: 0.01 seconds
Number of files: 28
Utilization of max. archive size: 0%
------------------------------------------------------------------------------
                       Original size      Compressed size    Deduplicated size
This archive:                9.30 MB              9.25 MB                531 B
All archives:               27.91 MB             27.74 MB              9.25 MB

                       Unique chunks         Total chunks
Chunk index:                      33                   93
------------------------------------------------------------------------------

ft  tux  ~
```

Figure 3: Storing an archive of a folder called `Pictures` with the `borg create` command. The `--stats` option specifies verbose output.

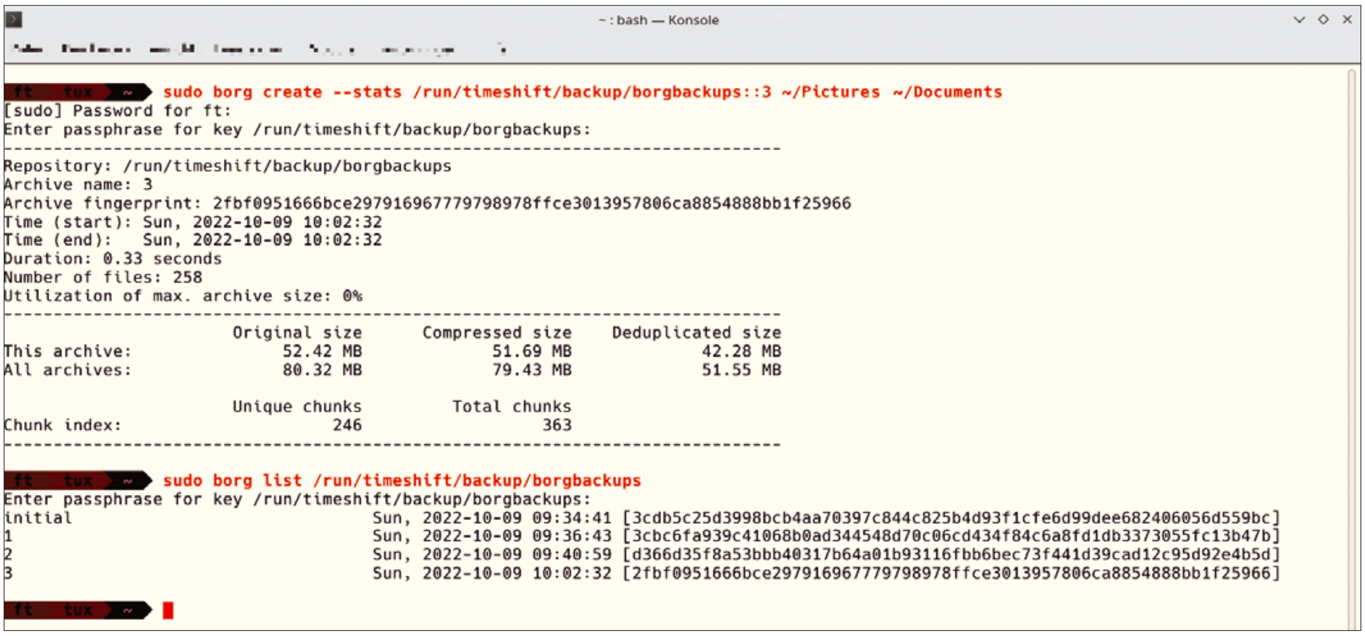

```
 ft  tux  ~  sudo borg create --stats /run/timeshift/backup/borgbackups::3 ~/Pictures ~/Documents
[sudo] Password for ft:
Enter passphrase for key /run/timeshift/backup/borgbackups:
------------------------------------------------------------------------------
Repository: /run/timeshift/backup/borgbackups
Archive name: 3
Archive fingerprint: 2fbf0951666bce2979169677797989 78ffce3013957806ca8854888bb1f25966
Time (start): Sun, 2022-10-09 10:02:32
Time (end):   Sun, 2022-10-09 10:02:32
Duration: 0.33 seconds
Number of files: 258
Utilization of max. archive size: 0%
------------------------------------------------------------------------------
                       Original size      Compressed size    Deduplicated size
This archive:               52.42 MB             51.69 MB             42.28 MB
All archives:               80.32 MB             79.43 MB             51.55 MB

                       Unique chunks         Total chunks
Chunk index:                     246                  363
------------------------------------------------------------------------------

 ft  tux  ~  sudo borg list /run/timeshift/backup/borgbackups
Enter passphrase for key /run/timeshift/backup/borgbackups:
initial                              Sun, 2022-10-09 09:34:41 [3cdb5c25d3998bcb4aa70397c844c825b4d93f1cfe6d99dee682406056d559bc]
1                                    Sun, 2022-10-09 09:36:43 [3cbc6fa939c41068b0ad344548d70c06cd434f84c6a8fd1db3373055fc13b47b]
2                                    Sun, 2022-10-09 09:40:59 [d366d35f8a53bbb40317b64a01b93116fbb6bec73f441d39cad12c95d92e4b5d]
3                                    Sun, 2022-10-09 10:02:32 [2fbf0951666bce2979169677797989 78ffce3013957806ca8854888bb1f25966]

 ft  tux  ~ 
```

Figure 4: **Adding an archive named `~/Documents` to the existing archive with the `~/Pictures` directory. Below that, the `list` option shows the created archives.**

Restore

To restore the archive, use the command from line 6 of Listing 1. You can do a full restore, but you can also mount an archive using FUSE, find the data, and reconstruct the data chunk by chunk. To delete an archive, you just need to replace `extract` with `delete` in the `borg` command. Always make sure there is enough space on the filesystem containing the repository. Once the space is completely used up, it will be difficult to free up space by deleting archives. The backup process can be automated using a script, as described in the Borg documentation.

Vorta offers two approaches to restoring, and both of them start with the *Archives* tab. After selecting the desired archive, you will see the *Extract* and *Mount* options in the right margin. After selecting *Extract*, you will see the backup with the full path in a new window. Highlight the directory to be restored and then press the *Extract*. In the next window, select the point where you want to insert the backup.

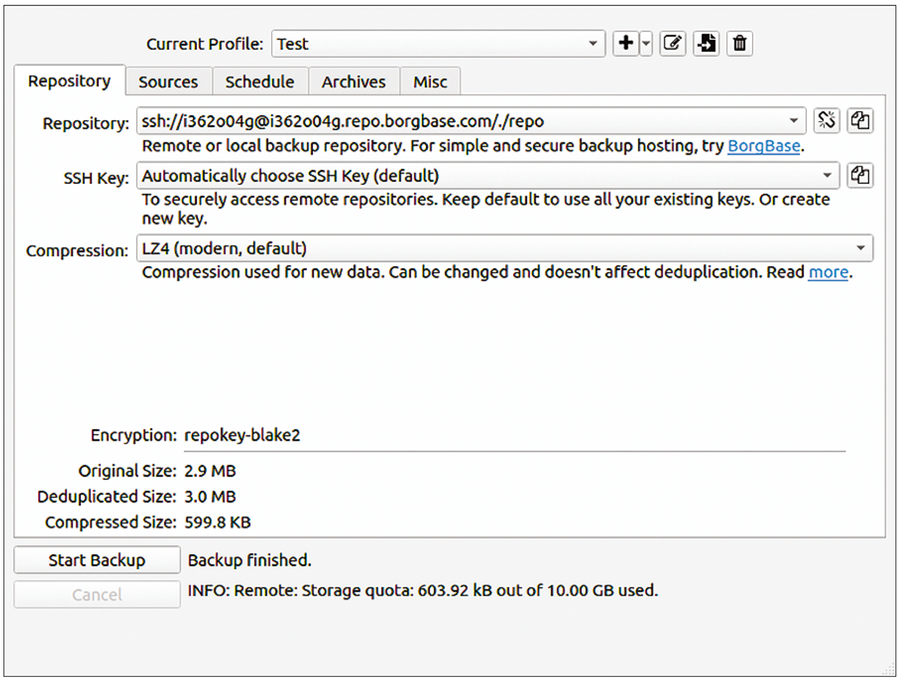
The second route is via the *Mount* option. In the next window, select a mount point, which is the directory that Vorta will use to give you access to the backup. Make sure that the

Vorta

If the commands and scripts become too complicated, you might want to install Vorta (Figure 5), a graphical user interface. Vorta is developed by the BorgBase hosting service (see the box entitled "BorgBase"). You will also find the Vorta software in the package sources of the mainstream distributions.

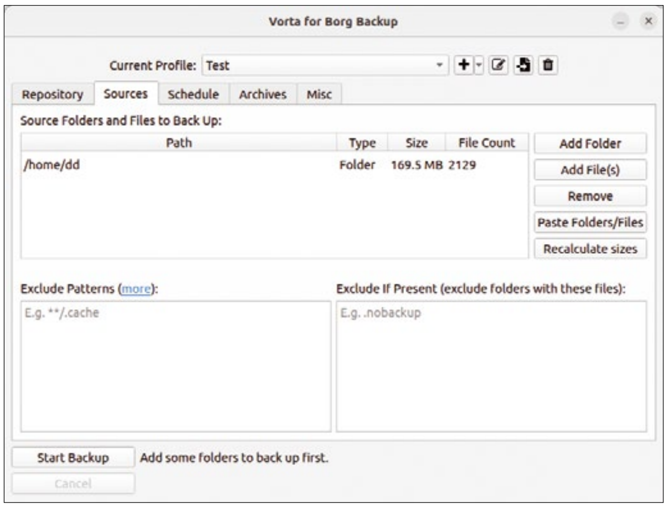
The first step is to create a repository; this repository can exist locally or on a remote server. Alternatively, you can add existing repositories. For authentication on remote computers, the software creates a new SSH key pair or uses an existing key pair. Compared to Back In Time, Borg's backups are about 40 percent smaller.

Then you create new repositories for the various backup tasks. For example, you can back up the entire home directory at night, while a second profile might let you create a copy of a critical directory every hour while you are working. You have the option to exclude specific paths from the backup set (Figure 6).



Repository: ssh://i362o04g@i362o04g.repo.borgbase.com/./repo
Remote or local backup repository. For simple and secure backup hosting, try BorgBase.
SSH Key: Automatically choose SSH Key (default)
To securely access remote repositories. Keep default to use all your existing keys. Or create new key.
Compression: LZ4 (modern, default)
Compression used for new data. Can be changed and doesn't affect deduplication. Read more.
Encryption: repokey-blake2
Original Size: 2.9 MB
Deduplicated Size: 3.0 MB
Compressed Size: 599.8 KB
Backup finished.
INFO: Remote: Storage quota: 603.92 kB out of 10.00 GB used.

Figure 5: **Vorta facilitates the task of controlling Borg and implements the most important commands in a graphical interface. Vorta is a Qt-based application that is similar to some of the well known backup apps, such as Back In Time or Déjà Dup.**



Current Profile: Test

Source Folders and Files to Back Up:

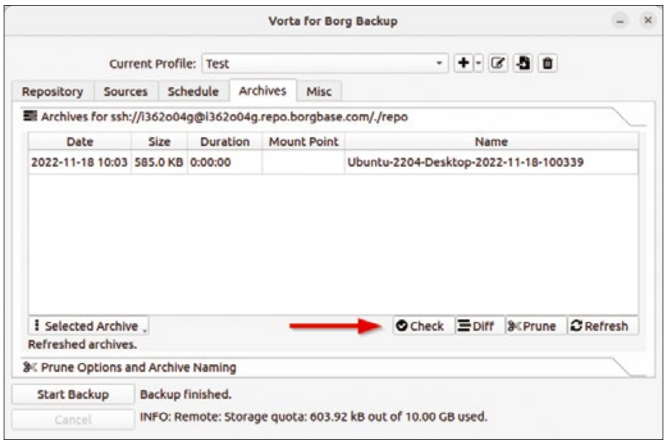
Path	Type	Size	File Count
/home/dd	Folder	169.5 MB	2129

Exclude Patterns (more): E.g. **/.cache

Exclude If Present (exclude folders with these files): E.g. .nobackup

Add some folders to back up first.

Figure 6: To exclude folders and files from the backup, enter the paths in the lower left corner of the mask.



Current Profile: Test

Archives for ssh://i362o04g@i362o04g.repo.borgbase.com/./repo

Date	Size	Duration	Mount Point	Name
2022-11-18 10:03	585.0 KB	0:00:00		Ubuntu-2204-Desktop-2022-11-18-100339

Refreshed archives.

Backup finished.

INFO: Remote: Storage quota: 603.92 kB out of 10.00 GB used.

Figure 7: Vorta can quickly restore Borg archives. The *Extract* and *Mount* options are both available for restore operations. *Mount* is more flexible, because you can use it to mount the archive in a folder on the target filesystem via FUSE.

BorgBase

If you want to store your backups offsite, but don't want to run a server yourself, the BorgBase [3] hosting service is a good choice, offering storage space starting at two euros a month. This investment gives you 100GB storage for up to 10 repositories. BorgBase offers server locations in Europe and the USA. Recently the developers added Restic as an additional backup tool to BorgBase.

After registering with the service and confirming your email address, log in to the website. A free trial account offers up to 10GB of data in a maximum of two repositories with no time limit. First, enter an SSH public key via *SSH Keys* and *Add Key*. If you don't have one, create a new key pair with the command:

```
ssh-keygen -t ed25519 -b 4096
```

You will find the key pair in a hidden directory named `~/.ssh`. Copy the fingerprint of the key with the `.pub` extension to the BorgBase screen and confirm.

Add your first repository and accept or change the default *EU* as the location. The other settings are largely self-explanatory. In *Access*, you need to select the key for full access. *Monitoring* lets you define if and when you want to be notified if a backup process fails. In *Advanced*, it's a good idea to check *Enable Storage Limit*, otherwise the account will automatically switch to a paid plan when the 10GB threshold is exceeded.

Before initiating an initial backup in Vorta, press the button to the left of the repository you just created. This option copies the path to the repository to the clipboard; you need to enter the path in Vorta. Click on the plus sign to the right of the repository address line and select *New Repository*. When you get there, transfer the path you just copied and assign a password. Clicking *Add* will then connect the remote repository to the local Vorta client. You can now create your first backup with BorgBase. If the connection fails, take a look at the troubleshooting tips [4] for your use case. Issues are almost always due to the SSH key and the way it was integrated.

Append-only mode gives you an extra level of security. Each repository can be switched to this mode. *Append-only* means that Borg never overwrites or deletes data transferred in it, nor does it delete the repository as a whole. This mode can currently only be enabled at the command line. The following command:

```
borg config /path_to_repository/ ↩
  repository_name append_only 1
```

enables the mode, and `append_only 0` switches it off.

mount point is an empty folder. After mounting, you will see a confirmation message. You can now access the archive you selected via the mount folder. Using your favorite file manager, you can restore the desired files by copying them directly from the archive and pasting them at the target location. When the process is complete, click *Unmount*.

The *Diff* switch discovers how two archives differ (Figure 7), and *Check* determines the integrity of an archive. Clicking *Prune* removes older archives. How many archives to remove and which ones you want to keep is set in the *Prune Options* settings.

Conclusions

BorgBackup is a very useful open source project supported by the contributions of more than 250 developers on GitHub [5]. The BorgBackup project has passed several peer reviews and has excellent documentation.

The main goal of Borg is to provide an efficient and secure way to backup data. Thanks to its deduplication technique, Borg is suitable for daily or hourly backups. Authenticated encryption enables secure backup to targets even if they are not fully trusted.

Vorta is a GUI by the Borg developers that integrates the most important functions into a graphical user interface. More advanced work is possible at the command line. If you want to control Borg at the prompt only, it is a good idea to check out the borgmatic [6] script, which is based on Borg. BorgBase is a hosting service that leaves the work to the professionals. Fees paid to BorgBase support Borg development. ■■■

Info

[1] BorgBackup packages: *https://repology.org/project/borgbackup/versions*

[2] Installing BorgBackup: *https://borgbackup.readthedocs.io/en/stable/installation.html*

[3] BorgBase: *https://www.borgbase.com*

[4] BorgBackup FAQ: *https://docs.borgbase.com/faq/#all-connections-to-a-borgbase-repo-fail-with-an-error-immediately*

[5] BorgBackup on GitHub: *https://github.com/borgbackup/borg*

[6] borgmatic: *https://torsion.org/borgmatic/*

Preservation

Graphical backup solutions help you protect your data with just a few mouse clicks. We study six popular options.

By Erik Bärwaldt

Many users still perceive the need for regular backups as a chore, but an abundance of graphical backup solutions in the Linux environment means that backup no longer requires complicated command-line input. These individual backup applications focus on different needs and therefore come with different feature sets.

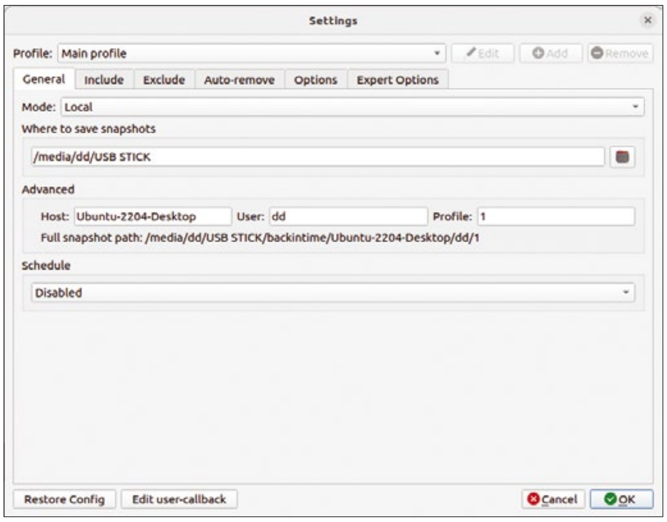
Many (but not all) of these graphical tools are based on Rsync, a command-line program for synchronizing files on different local or remote disks [1]. This article looks at six easy-to-use graphical applications for the Linux desktop. Other articles in this issue examine some other leading backup alternatives.

Back In Time

Back In Time [2] is an Rsync-based backup program that has been under development since 2008. The project's GitHub page says Back In Time is "inspired by FlyBack." (FlyBack is another open source backup tool modeled on Apple's Time Machine.) Both a command-line version and a Qt-based graphical variant are available. The application, written in Python 3, is available in the repositories of all major Linux distributions. Back In Time uses profiles; you can create a profile defining a specific backup scenario, and then conveniently perform the backup at the push of a button.

After completing the install, you will find two new entries for Back In Time in the menu of your desktop environment. The first entry launches the software with the rights of the logged-in user, and the second entry is for the root user. Launching the application with root privileges allows backups of the drives and directories that a logged-in user might not be able to access, including complete snapshots of the disk.

Back In Time opens to a Settings window (Figure 1), where you create the main profile and configure several other options. In the *General* tab, you first need to specify the software location and choose whether to back up locally or to a remote drive. You can also choose to encrypt the data and define a backup interval. Please note that the target media needs to be formatted with the ext3 or ext4 filesystem. The FAT32 filesystem commonly used for USB memory sticks does not support hard links and is therefore not suitable for backing up Linux filesystems.



Settings

Profile: Main profile

Mode: Local

Where to save snapshots

/media/dd/USB STICK

Advanced

Host: Ubuntu-2204-Desktop User: dd Profile: 1

Full snapshot path: /media/dd/USB STICK/backintime/Ubuntu-2204-Desktop/dd/1

Schedule

Disabled

Figure 1: Back In Time works with profiles that are easy to configure.

You define the datasets to include in the backup using the *Include* and *Exclude* tabs. In *Include*, specify the directories and files to back up, whereas *Exclude* already offers a choice of predefined exclusion patterns. The exclusion patterns could include directories and various file name extensions that the backup action will not take into account. You can also add additional directories, files, or templates. In addition, you can exclude files over a certain size; the default size is 500MB.

The *Auto-remove* tab offers parameters that let you automate the task of removing obsolete datasets. The free space available on the target medium, the age of the files, or the number of free inodes can serve as criteria.

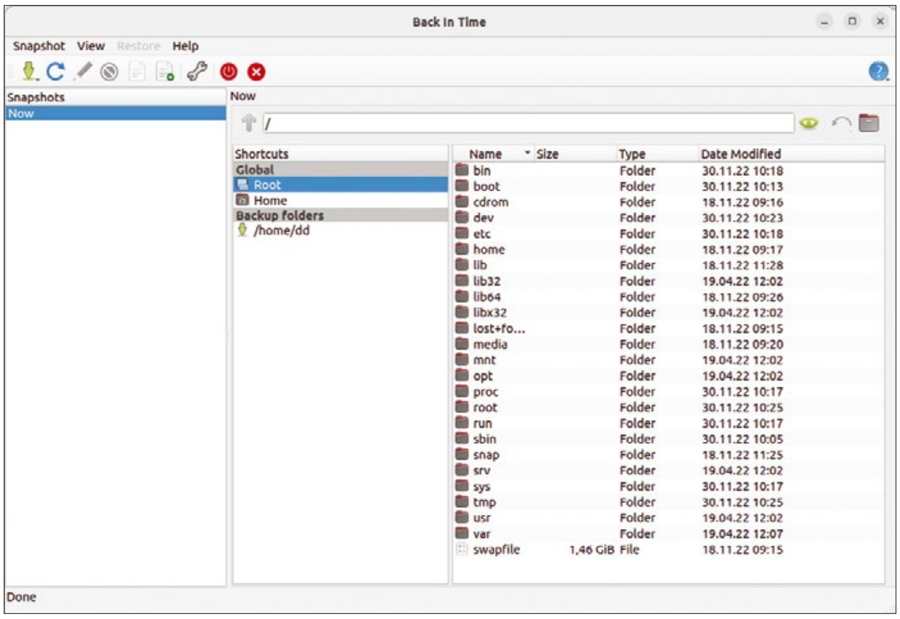
The *Options* tab offers general settings, such as how to handle incomplete snapshots, the level of detail for logs, and whether to display notifications. In the last tab *Expert Options*, you can change various parameters relating to cron jobs and Rsync.

After completing the settings, press *OK*. The software saves the profile and opens the backup window (Figure 2).

The window contains a conventional menu and buttonbar, as well as a snapshot list. Next to it are two areas with a file manager, where you can see the source drive along with all the hidden files.

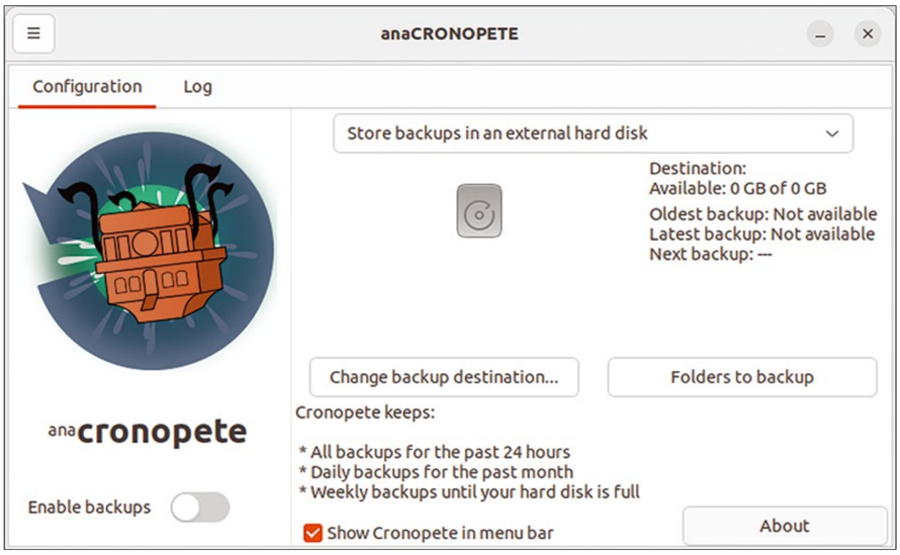
Right-click on files and directories to open a context menu that allows you to exclude a directory or file from the backup. After you have made any further changes, click on the diskette symbol in the buttonbar to start the backup. If you have enabled a schedule in the Settings dialog, Back In Time will create backups automatically in the future. Please note that, depending on the volume of data and the speed of the target drive, the backup can take quite a while to complete.

If you want to restore an existing backup, highlight the desired snapshot in the top left corner of the *Snapshots* table, and then select the *Restore* option in the menubar. In the context menu, select the *Restore* entry if you want to restore the data to the original location. If you want Back In Time to store the data in a location other than the source directory, select the *Restore to* option.



Name	Size	Type	Date Modified
bin		Folder	30.11.22 10:18
boot		Folder	30.11.22 10:13
cdrom		Folder	18.11.22 09:16
dev		Folder	30.11.22 10:23
etc		Folder	30.11.22 10:18
home		Folder	18.11.22 09:17
lib		Folder	18.11.22 11:28
lib32		Folder	19.04.22 12:02
lib64		Folder	18.11.22 09:26
libx32		Folder	19.04.22 12:02
lost+fo...		Folder	18.11.22 09:15
media		Folder	18.11.22 09:20
mnt		Folder	19.04.22 12:02
opt		Folder	19.04.22 12:02
proc		Folder	30.11.22 10:17
root		Folder	30.11.22 10:25
run		Folder	30.11.22 10:17
sbin		Folder	30.11.22 10:05
snap		Folder	18.11.22 11:25
srv		Folder	19.04.22 12:02
sys		Folder	30.11.22 10:17
tmp		Folder	30.11.22 10:25
usr		Folder	19.04.22 12:02
var		Folder	19.04.22 12:07
swapfile	1,46 GiB	File	18.11.22 09:15

Figure 2: The Back In Time backup window.



Store backups in an external hard disk

Destination:
Available: 0 GB of 0 GB
Oldest backup: Not available
Latest backup: Not available
Next backup: —

Cronopete keeps:

* All backups for the past 24 hours
* Daily backups for the past month
* Weekly backups until your hard disk is full

Enable backups

☑ Show Cronopete in menu bar

Figure 3: The Cronopete program window hardly requires any training.

Back In Time then displays a safety prompt and asks you to confirm that you want to restore the data. A log window provides progress information so that you can correct any problems that occur.

Cronopete

Like Back In Time, the free Cronopete [3] is a backup tool modeled on Time Machine that focuses on simple, largely automated backup. Most popular Linux distributions have the software in their repositories. However, the project's website also offers packages of the small program for various Debian, Ubuntu, Fedora, and Arch variants.

After the install, you will find two new entries in your desktop menu. One is used to define backup actions, the other to restore existing backups. On the start screen, first click *Configure now* to enter the configuration dialog. Cronopete then opens an easy-to-use settings window where you define the automatic backups, the datasets, and the target folders (Figure 3).

At the bottom of the window, slide the *Enable backups* slider to the right to enable automatic backup. Then close the window, which prompts Cronopete to drop a control icon into the system tray. Right-click the icon to open a small context menu, and click the *Back Up Now* option. The application creates a backup of the desired data. If you want to make changes to the configuration later, click the *Configure the backups* option in the context menu of the control icon.

Cronopete creates a short log for each new backup action, deleting the

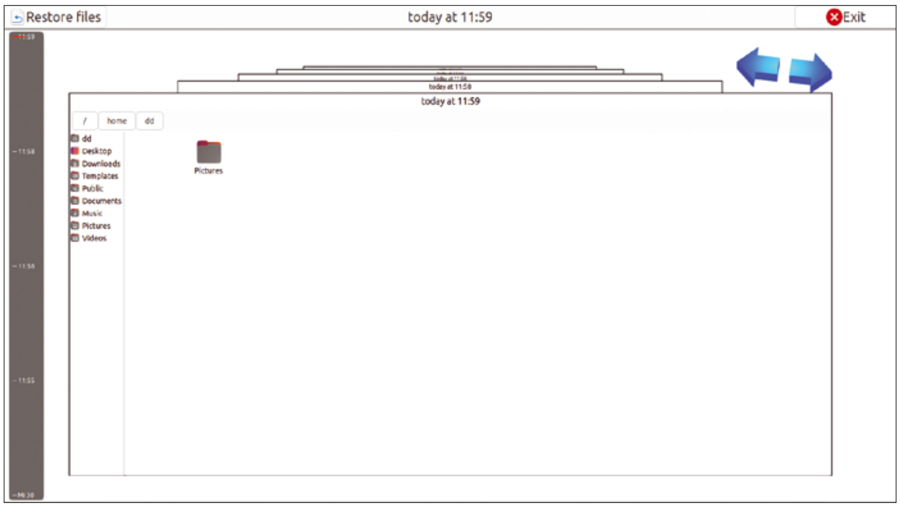


Figure 4: Triggering a restore action in a very unusual looking window.

old logs. You can view the current log in the *Log* tab of the main window.

To restore a backup set, find the launcher in the working environment menu called *Cronopete: restore backup*. You can also click on the Cronopete icon in the system tray and select the *Restore* files option in the context menu.

A very unusual looking window then opens in full screen mode (Figure 4). It shows a vertical timeline on the left, with the existing backups appearing one above the other in their own file windows. Top right on the screen are two blue arrows that you can use to toggle between the individual backups in the file managers. In the timeline, a small red bar shows you the current backup status. This means that you can browse through the individual backups and move freely within the backed up directories in the file managers, although, unlike in conventional file managers, you can't open the files.

Pressing the *Restore files* button in the titlebar lets you restore the backup to its original location. A click on the *Exit* button terminates the program.

Déjà Dup

Déjà Dup [4] is one of the most intuitive programs for local backups. Virtually all of the major Linux distributions have the Gnome-based Déjà Dup in their repositories, which makes it convenient to install. Déjà Dup is actually a graphic front end for the open source duplicity backup tool.

Launching the application takes you to a very plain user interface offering just two options and a vault icon. The Déjà Dup interface follows the traditional Gnome guidelines, doing without menus or buttonbars and with the controls integrated into the titlebar.

First of all, you will need to configure some settings. You can access the configuration dialog via the hamburger menu in the titlebar. Select the *Preferences* option. The dialog that opens has just a few options. Use the *General* tab to define the retention period for the individual backups and configure automatic backup actions. You can only choose between weekly and daily for automated backup frequency.

By default, the software creates the backup in the Google Drive cloud. However, you can specify a computer on your LAN or a local folder instead. A local folder only makes sense if you specify an external drive as the target. After selecting the target drive, specify a target folder that needs to already exist on the drive.

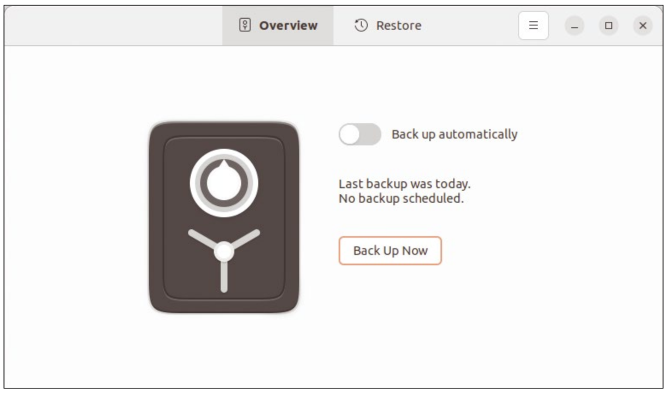
In the *Folders* tab, which you can access from the titlebar in the main window, specify the directories you wish to back up and the ones you wish to exclude from the backup. After you click on the plus symbol, the software opens the integrated file manager; you do not need to type in folder names but can select the directories at the push of a button.

Déjà Dup then lists the folders to be backed up. If you have set up several partitions on your computer's drive, you can back up directories across partitions, provided that the partitions are mounted. Clicking the magnifying glass icon in the corner of the titlebar also lets you quickly access functions by entering search terms.

Once you have completed all the settings, close the window. The application now opens the backup window, where you can enable automated backups. If you want to start a manual backup instead, click *Back Up Now* (Figure 5).

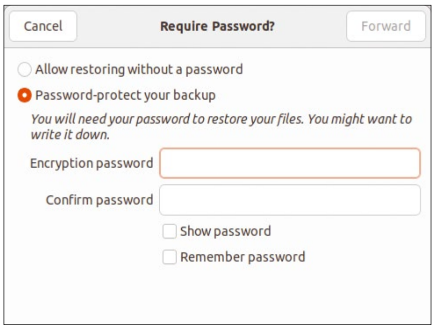
Afterwards, Déjà Dup prompts you to protect the backup against unauthorized access with a password (Figure 6). If you click the radio button to the left of *Allow restoring without a password*, the restore will work without authentication. After clicking *Next*, the backup starts; a progress bar provides information on the status.

To restore the data, click *Restore* in the titlebar. Déjà Dup displays the backed up folders, and you can select the desired directories with a mouse click. Then click the *Restore* button in the bottom left of the program window. In another dialog, you



Last backup was today.
No backup scheduled.

Figure 5: Déjà Dup is definitely a no-frills program.



Allow restoring without a password
[✓ Password-protect your backup]
You will need your password to restore your files. You might want to write it down.
Encryption password
Confirm password
Show password
Remember password

Figure 6: Déjà Dup lets you password protect your data against unauthorized access.

can specify whether you want the application to restore the data to the original location or to an optional directory.

luckyBackup

Another backup solution based on Rsync is luckyBackup [5]. You'll find luckyBackup in the software archives of all the mainstream Linux distributions. The installation routine creates a launcher, which opens a program window with an unusual layout. In addition to a conventional menu and buttonbar, you will find a task list with some controls and an information window that displays notifications and messages from the application.

LuckyBackup relies on profiles for different backup tasks; this means that you don't have to specify the source and target directories manually every time you start the application. You can switch between the individual profiles using a selection box in the buttonbar. At first there is only the *default* profile. To create a new profile, click on *Profiles* in the menubar and select the *New* option. Enter the name of the new profile, which luckyBackup will then add to the profile list.

The next step is to start defining tasks for the new profile. Click the *Add* button in the *Task* column. Enter the name of the task and specify the source and target directories. Once you have added one or more tasks, you can check them for correct operation in a simulation mode (Figure 7). LuckyBackup indicates the end of the test by displaying a message in purple.

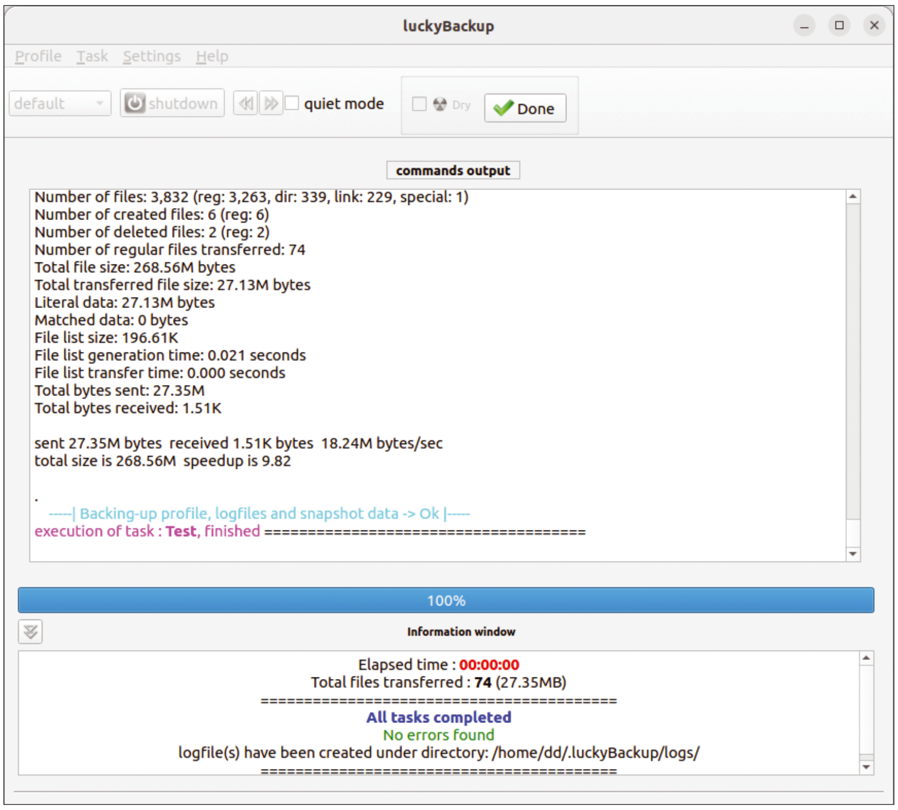
You can modify a task after it has been created. Select the task and then click *Edit*. In the properties dialog, edit the properties, or click *Advanced* to tweak additional settings. Several tabs let you exclude or include directories, define various options for links, and specify a remote computer as the target or source. You can also set up an encrypted SSH connection.

LuckyBackup also supports fully automated, scheduled backups. Click on the clock icon or select *Schedule* from the *Profiles* menu.

In the settings dialog, you will need to create an execution plan for each profile. In addition to the time, you can specify the days of the week, the months, or alternatively, days in the month, on which a backup will be created. Once you have set the dates for the backup, click on *Okay* and then click *CronIT* to save the values.

After completing the configuration, you can start a manual backup by unchecking the *Simulation* option in the application's main window and then pressing *Run*. LuckyBackup displays messages in the command-line output and shows you a progress bar. Depending on the size of the files and folders and the speed of the source and destination media, the backup can take a while to complete.

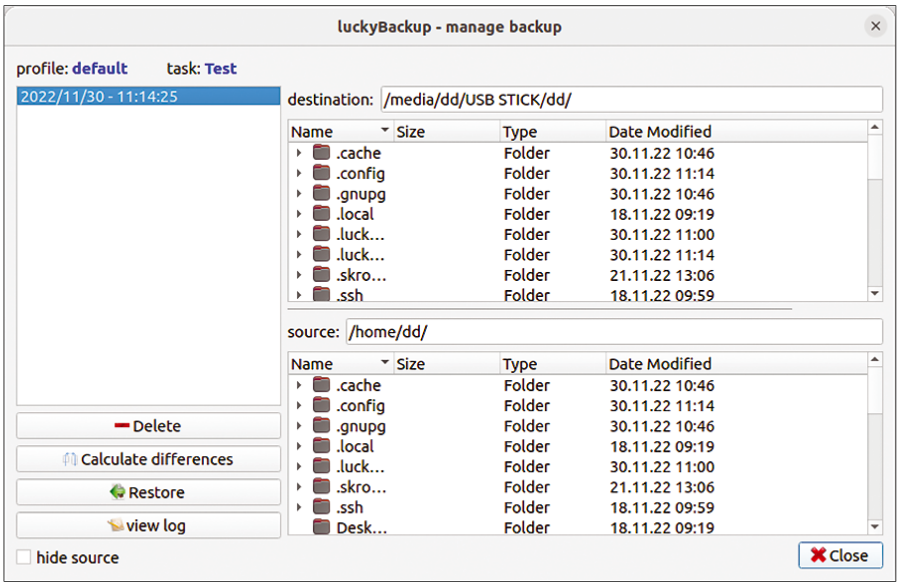
LuckyBackup does not offer an option that is labeled *Restore*. Instead, you use the *Manage Backup* option in the Task menu to restore your data. You will find a list of existing backups, including the profiles and task definitions on the left (Figure 8). The associated files in the source and target directories appear in two larger areas on the right. You can view the corresponding log in a separate window by clicking the *View log* button.



```
Number of files: 3,832 (reg: 3,263, dir: 339, link: 229, special: 1)
Number of created files: 6 (reg: 6)
Number of deleted files: 2 (reg: 2)
Number of regular files transferred: 74
Total file size: 268.56M bytes
Total transferred file size: 27.13M bytes
Literal data: 27.13M bytes
Matched data: 0 bytes
File list size: 196.61K
File list generation time: 0.021 seconds
File list transfer time: 0.000 seconds
Total bytes sent: 27.35M
Total bytes received: 1.51K

sent 27.35M bytes  received 1.51K bytes  18.24M bytes/sec
total size is 268.56M  speedup is 9.82

.
  ----| Backing-up profile, logfiles and snapshot data -> Ok |----
execution of task : Test, finished ====================================
```

```
Elapsed time : 00:00:00
Total files transferred : 74 (27.35MB)
========================================
All tasks completed
No errors found
logfile(s) have been created under directory: /home/dd/.luckyBackup/logs/
========================================
```

Figure 7: You can also perform a dry run in luckyBackup.



profile: default   task: Test

2022/11/30 - 11:14:25

destination: /media/dd/USB STICK/dd/

Name	Size	Type	Date Modified
.cache		Folder	30.11.22 10:46
.config		Folder	30.11.22 11:14
.gnupg		Folder	30.11.22 10:46
.local		Folder	18.11.22 09:19
.luck…		Folder	30.11.22 11:00
.luck…		Folder	30.11.22 11:14
.skro…		Folder	21.11.22 13:06
.ssh		Folder	18.11.22 09:59

source: /home/dd/

Name	Size	Type	Date Modified
.cache		Folder	30.11.22 10:46
.config		Folder	30.11.22 11:14
.gnupg		Folder	30.11.22 10:46
.local		Folder	18.11.22 09:19
.luck…		Folder	30.11.22 11:00
.skro…		Folder	21.11.22 13:06
.ssh		Folder	18.11.22 09:59
Desk…		Folder	18.11.22 09:19

Figure 8: Options for varying the data restore.

For an overview of the differences between the source and target directories, use the *Calculate Differences* option. Instead of the target, a table of differences between the existing dataset and its backup now appears in the top right corner of the window. To restore the highlighted backup, click the *Restore* button. LuckyBackup will now prompt you for the restore path. A checkbox lets you choose to delete existing files in the source directory that are missing from the backup.

If you first want to determine whether the restore will work without trouble, you can simulate the data restore by checking *Simulation*. Then press *Start*. The tool will perform a dry run of the restore, reporting on any errors it encounters.

Timeshift

Timeshift [6] is one of the best-known backup programs for Linux desktops, which is why it crops up in the software repositories of virtually all of the mainstream distributions. Timeshift relies on Rsync to create snapshots, but it can also handle the Btrfs filesystem, which natively supports snapshots.

After launching Timeshift for the first time and authenticating as the system administrator, you will first be asked if you want to select Rsync or Btrfs as the snapshot type. The program then calculates the system size and, on the next screen, displays the capacity and free space of all storage media mounted on the system and the partitions on those media. From this list, you can choose the target location for saving the new snapshots you will be creating.

Timeshift (Figure 9) does not support the filesystems of other operating systems. If you want to use a removable medium such as a USB stick as the target medium, you need to format it with a Linux-compatible filesystem. After completing the settings, press *Next*. You then need to specify how often you want the system to be backed up and how many backups Timeshift will keep each time. The tool saves your configuration as a cron job.

In the next wizard dialog, specify the data you wish to back up. By default, Timeshift excludes the user's home directories from the backup, but you can elect to include them. In addition, you can include just hidden files or all files in the backup.

In the program window, you can quickly access the most important functions via a buttonbar. The main segment of the window lists the existing snapshots, and a status bar at the bottom of the screen shows the number of snapshots and the space available on the target media. Use the *Create* button to create the first snapshot manually. Timeshift backs up the entire system, including the manually added home directories. In a separate window, the application displays the backup progress in real time.

Timeshift creates the snapshots in a directory of the same name on the target disk; the snapshots are stored in different folders depending on the cron job configuration.

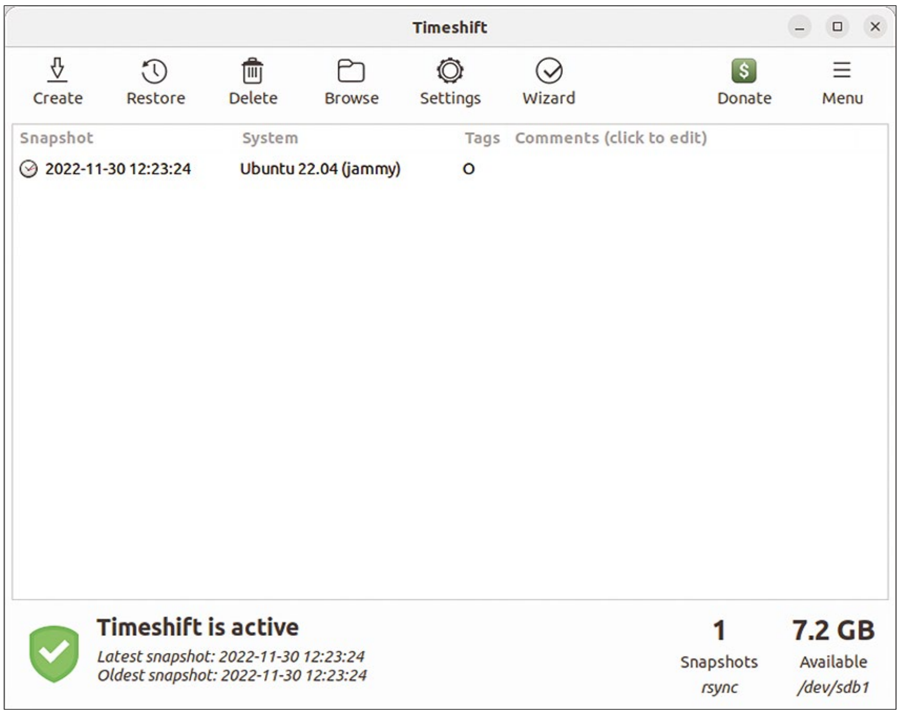
To restore data, first select the desired snapshot from the list and then press *Restore*. A dialog opens, where you can select the devices and paths to restore. The routine restores the entire system. User directories not included in the backup are kept. You can specify directories on all mounted disks as targets (Figure 10). Make sure you have enough free storage capacity.

In the next step, Timeshift first performs a dry run. Then the program displays all the pending actions for restoring, allowing you to confirm. After confirming the prompt, Timeshift switches to text mode and backs up the data. The backup is followed by an automatic reboot of the computer.

Vorta

Vorta [7] acts as a graphical front end for the BorgBackup [8] command-line program (see the article on BorgBackup elsewhere in this issue). The cross-platform application is available in the repositories of many Linux distributions. You can also pick up the source code from the project website, and there is a Flatpak on Flathub.

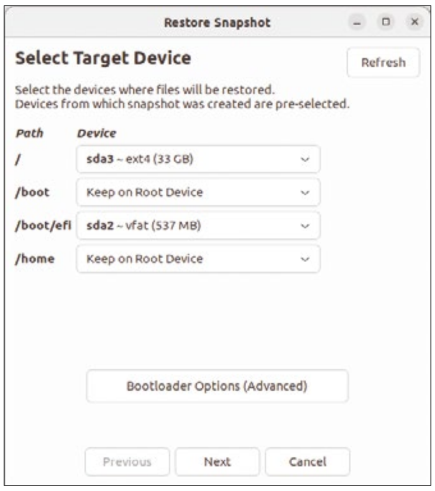
Vorta starts up with a very simple and well-organized program window.



Snapshot	System	Tags	Comments (click to edit)
2022-11-30 12:23:24	Ubuntu 22.04 (jammy)	O	

Timeshift is active
Latest snapshot: 2022-11-30 12:23:24
Oldest snapshot: 2022-11-30 12:23:24
1 Snapshots rsync
7.2 GB Available /dev/sdb1

Figure 9: Timeshift is ready to use after a few mouse clicks.



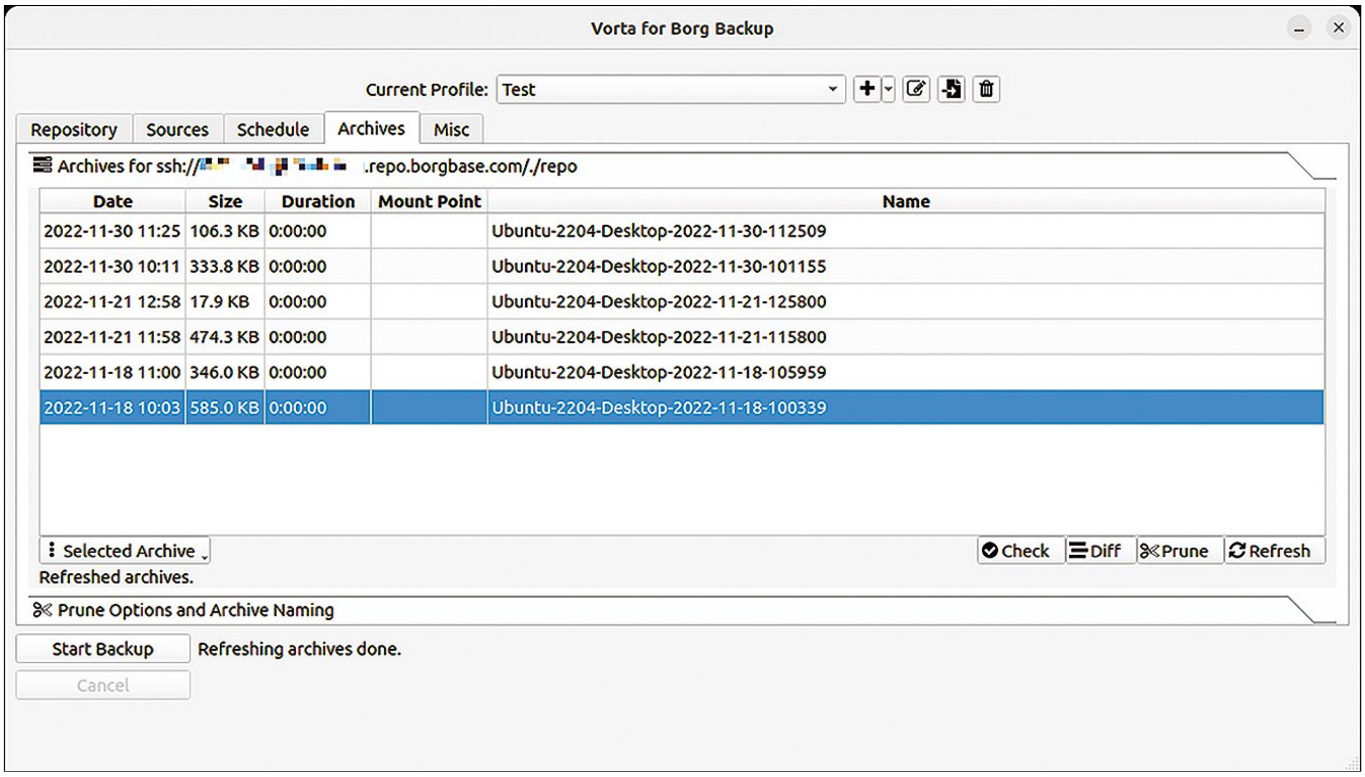
Select Target Device
Select the devices where files will be restored.
Devices from which snapshot was created are pre-selected.

Path	Device
/	sda3 – ext4 (33 GB)
/boot	Keep on Root Device
/boot/efi	sda2 – vfat (537 MB)
/home	Keep on Root Device

Figure 10: Timeshift also lets you select variable target directories for the restore.

When you get there, you can complete the configuration of the tool in several tabs, defining different profiles for different backup tasks. You can add a new profile, in addition to the current *Default* profile, by pressing the button with the plus sign next to the profile selection field. The new profile is then displayed, allowing you to configure the settings (Figure 11).

In the *Repository* tab, first specify the archive where Vorta will store the backups. You can also use online storage. Then specify the access method for the selected repository; data transfer always takes place via a secure SSH connection. You can also define a compression method in this dialog to reduce the transfer volume. You have a choice of several compression methods, or you can save the data without compression. In the



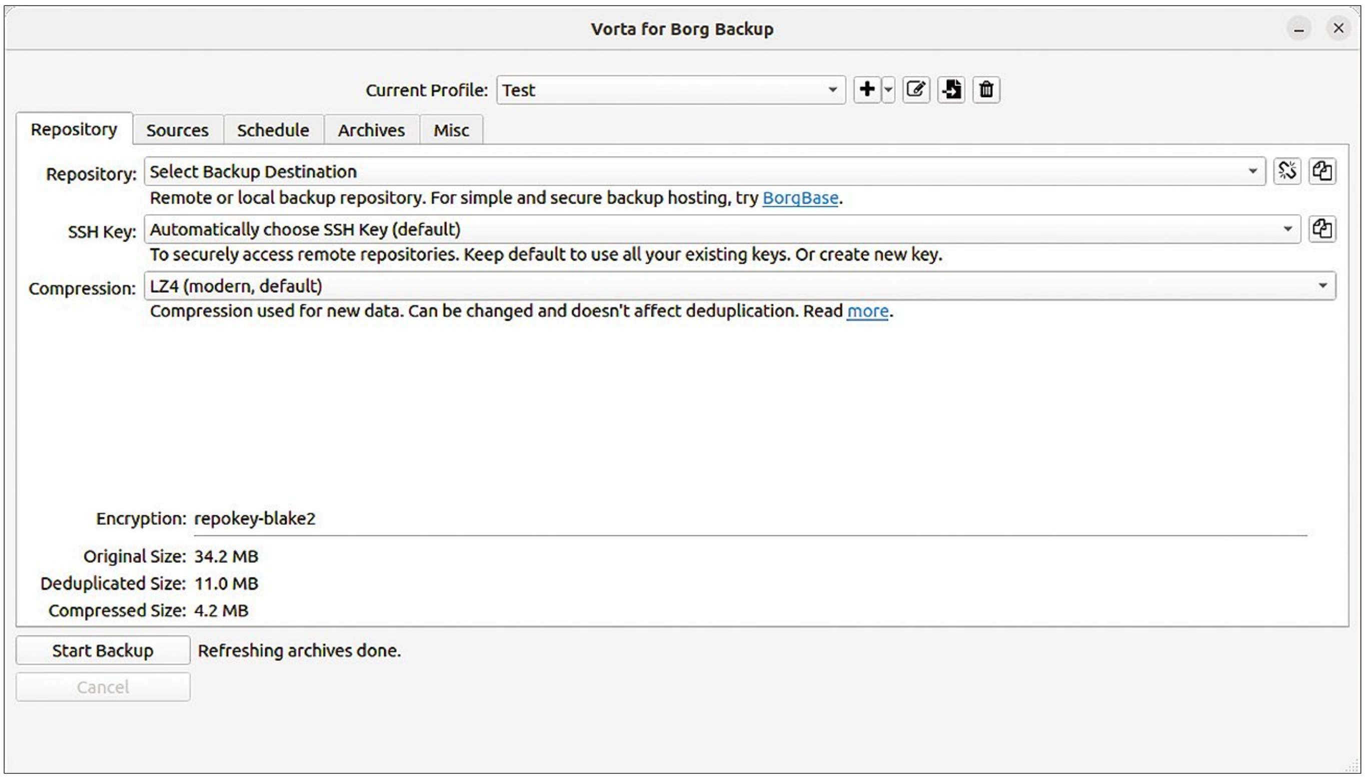
Archives for ssh://[redacted].repo.borgbase.com/./repo

Date	Size	Duration	Mount Point	Name
2022-11-30 11:25	106.3 KB	0:00:00		Ubuntu-2204-Desktop-2022-11-30-112509
2022-11-30 10:11	333.8 KB	0:00:00		Ubuntu-2204-Desktop-2022-11-30-101155
2022-11-21 12:58	17.9 KB	0:00:00		Ubuntu-2204-Desktop-2022-11-21-125800
2022-11-21 11:58	474.3 KB	0:00:00		Ubuntu-2204-Desktop-2022-11-21-115800
2022-11-18 11:00	346.0 KB	0:00:00		Ubuntu-2204-Desktop-2022-11-18-105959
2022-11-18 10:03	585.0 KB	0:00:00		Ubuntu-2204-Desktop-2022-11-18-100339

Refreshed archives.

Refreshing archives done.

Figure 11: In Vorta, you first need to create a profile in the wizard.



Repository: Select Backup Destination
Remote or local backup repository. For simple and secure backup hosting, try BorgBase.
SSH Key: Automatically choose SSH Key (default)
To securely access remote repositories. Keep default to use all your existing keys. Or create new key.
Compression: LZ4 (modern, default)
Compression used for new data. Can be changed and doesn't affect deduplication. Read more.

Encryption: repokey-blake2
Original Size: 34.2 MB
Deduplicated Size: 11.0 MB
Compressed Size: 4.2 MB

Refreshing archives done.

Figure 12: Vorta can also check the integrity of file archives on demand.

Sources tab, specify the datasets you want the tool to back up. You can include or exclude files and directories. Wildcards are supported.

The *Schedule* tab lets you schedule backups, which means that you can create automatic backups at specified times. You can also specify a period of time during which Vorta will validate the existing repository. This validation prevents the data from accidentally ending up in a corrupt archive and being unrecoverable.

The *Archive* and *Misc* tabs grant access to some general settings but do not initially require any input or modifications. After completing the configuration, make an initial backup by clicking *Start Backup* in the program window. In the *Repository* tab, Vorta displays various statistics during the backup. After the backup is complete, you can see the original and compressed size of the backup set. If necessary, verify the saved data in the *Archives* tab by selecting the archive in the list view and then clicking the *Check* button. The tool checks the archive and displays the results (Figure 12).

Vorta does not create a full backup each time a profile is used but instead only saves the differences from the original in the subsequent backups. To find the differences between individual backups, select the desired backups in the archive list and then click the *Diff* button. The application will show you the root directories where it has detected differences. Clicking on the plus symbol expands the directory tree, with the differing files appearing in green.

To keep the list of existing archives from getting out of hand, you can thin them out. Click the *Prune Options and Archive Naming* button at the bottom of the *Archives* tab. In the settings dialog, specify the number of archives you want to keep by defining the number for various age categories. The selectable intervals range from hourly archives to yearly backups. In addition, you can change the naming conventions if necessary.

During the install, Vorta creates a small disk icon in the system tray on the desktop. Left-click on the icon to open the application window; right-click to close the app. Also in the context menu, you can select a profile or create a manual backup.

To access the backed up data, mount the archives like conventional drives on your Linux system. To mount the archive, select it in the *Archives* tab, and then click the small triangle to the right of the *Selected Archive* button. In the context menu, click on the *Mount* option and specify the mount path.

To restore an archive, click *Extract* in the same context menu. Vorta will prompt you for the components of the archive. You can define the path for unpacking the archive or the selected components in a small file manager. After you click *Choose*, the software unpacks the archive or its selected components in the target directory.

Conclusions

There is no shortage of graphical front ends for backing up a Linux desktop environment. The choice ranges from applications for making complete system snapshots like Timeshift to very compact applications for backing up personal data like Déjà Dup. Some of the tools can even store backups on remote computers, and all of them support external disks connected to the local system. Table 1 will help you sort through the various options as you shop for a graphical backup solution. ■■■

Info

[1] Rsync: *https://rsync.samba.org*

[2] Back In Time: *https://github.com/bit-team/backintime*

[3] Cronopete: *https://www.rastersoft.com/programas/cronopete.html*

[4] Déjà Dup: *https://wiki.gnome.org/Apps/DejaDup*

[5] luckyBackup: *https://luckybackup.sourceforge.net*

[6] Timeshift: *https://teejeetech.com/timeshift/*

[7] Vorta: *https://vorta.borgbase.com*

[8] BorgBackup: *https://www.borgbackup.org*

Table 1: Desktop Backup Applications at a Glance

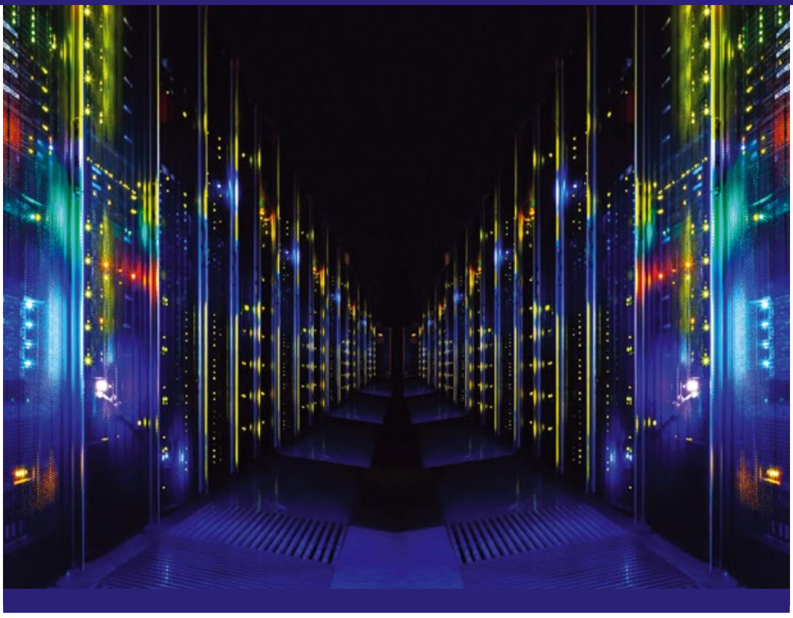
	Back In Time	Cronopete	Déjà Dup	luckyBackup	Timeshift	Vorta
License	GPLv2	GPLv3	GPLv3	GPLv3	LGPLv3	GPLv3
Profiles	yes	no	no	yes	no	yes
Scheduled backups	yes	yes[1]	yes[1]	yes	yes	yes
System snapshots	yes	no	no	yes	yes	yes
Remote backup	yes	no	yes	yes[1]	yes	yes
Encrypted backups	yes	no	yes[1]	yes	yes	yes
Manual inclusion and exclusion of data	yes	no	yes[1]	yes	yes	yes
Maximum file size for backup adjustable	yes	no	no	no	no	no
File compression adjustable	no	no	no	no	no	yes
Scheduled removal of obsolete backups	yes	no	yes	yes	yes	yes
Different filesystems on target media	yes[1]	yes	yes	yes	yes[1]	yes
File restore to different locations	yes	no	yes	yes	yes	yes
Simulation	no	no	no	yes	no	no
Log function	yes	yes	no	yes	yes	yes

[1] **Restrictions apply**

Synchronize your data across multiple devices with Syncthing

Sync or Swim

If you need a cloudless solution for syncing data across multiple devices, Syncthing could be just the thing. *By Erik Bärwaldt*



You are probably familiar with the problems of using more than just one desktop computer. You also might have a laptop, a tablet, or other computer systems. If you use these devices regularly, large datasets will quickly accumulate, causing the need to keep data synchronized across the various devices.

Permanent synchronization of personal data across multiple terminal devices was previously the domain of cloud services or dedicated server instances. With Syncthing, you can keep all your devices up to date without the hassle of setting up your own server or sharing your data with public cloud services. Syncthing [1] syncs the data directly between the systems, although the folder structures can differ. The two devices only need to be switched on and accessible on the local network.

Getting Started

The cross-platform Syncthing software is available from the repositories of all the major distributions. In addition to the 32- and 64-bit versions for the x86 system architecture, you will also find packages for the ARM platform. And you can pick up an app for Android v4.1 or newer from the F-Droid store.

If you are setting up from the repositories, the installation creates two starters in the desktop environment menu structure. If you use the software offered on the project page [2], you will need to add it to your system's menu structure manually after unpacking the tarball.

When you launch the application, Syncthing opens the comparison tool in the background and on your web browser. On first launch, the program also generates the keys and certificates that you need for secure data transfer. The dashboard appears in the browser window; you can use it to customize the program to suit your needs.

The splash page prompts you to create a username and password on first launch. To do so, press *Actions* in the top right corner of the browser window and select *Settings* from the context menu. In the dialog box that appears, switch to the *GUI* tab and enter the desired authentication data in the *GUI Authentication User* and *GUI Authentication Password* fields. Then press *Save*. After that, the window will close and you will need to authenticate when you open the web interface again.

Interface

Syncthing's web interface consists of three large panels. On the left side of the window, you will find the *Folders* table, which contains the local directories that you have included for synchronization. In the upper right corner is a *This Device* info area, which shows you some statistics for the local computer system. Below that, the *Remote Devices* panel shows you the connected systems with which the software can synchronize.

Right at the top of the browser window, you will find three shortcuts on the left. *Actions* shuts down or restarts the software, displays the logs, or opens the configuration menu. Use the *General* tab to specify the minimum free disk space on the local machine required for the index database. In addition to a percentage value, you can also specify absolute values in the KB to TB ranges.

To add external devices for data synchronization, press *Add device* in the *Remote Devices* section of the browser window. The software automatically searches the local network for other devices with retrievable identifiers and displays them in an overlapping window (Figure 1). To integrate new terminal devices, you need to make sure they are switched on and that a Syncthing instance is also running on them. Clicking on the identifier integrates the remote devices with the local system. Since the device name is not shown, it makes sense to assign a meaningful device name in the input line for ease of identification. Otherwise, the software will show you the device name that it uses on the terminal device, usually the hostname.

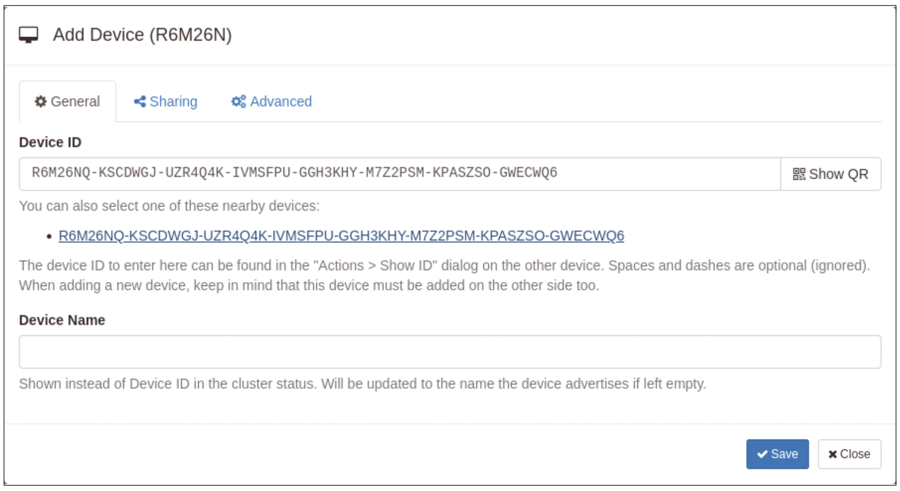
The *Sharing* and *Advanced* tabs contain additional settings. For example, you can opt for unidirectional or bidirectional data synchronization by integrating the remote systems into the local installation as distribution devices. In the *Advanced* tab, you can also specify whether Syncthing compresses the data and what transfer limits apply to incoming and outgoing connections.

After completing the settings, press *Save* bottom right. On the remote device, a message will appear in the Syncthing browser window pointing to a connection attempt and prompting you to add the system by pressing *Add Device*. A short time after accepting the connection, both devices will show the newly connected system, with the device name and connection status.

Sharing Folders

To add local folders to the system for synchronization, click on *Add Folder* in the *Folders* area of the browser window. In the dialog, enter a name for the folder to be synchronized and specify its path. In the *Sharing* tab, you can then specify the end devices on the LAN with which you want to share the folder. The connected devices are listed in a table. To enable one of them, check the box to the left of its name.

In the *File Versioning* tab, you can specify whether or not you want to create multiple chronologically ordered versions of the same folder. The *File Versioning* field gives you a choice between different versioning modes, and you can choose the configuration for this in a supplementary dialog. The *Ignore Pattern* tab lets you use placeholders to specify which file


Add Device (R6M26N)

General | Sharing | Advanced

Device ID

R6M26NQ-KSCDWGJ-UZR4Q4K-IVMSFPU-GGH3KHY-M7Z2PSM-KPASZSO-GWECWQ6 [Show QR]

You can also select one of these nearby devices:

- R6M26NQ-KSCDWGJ-UZR4Q4K-IVMSFPU-GGH3KHY-M7Z2PSM-KPASZSO-GWECWQ6

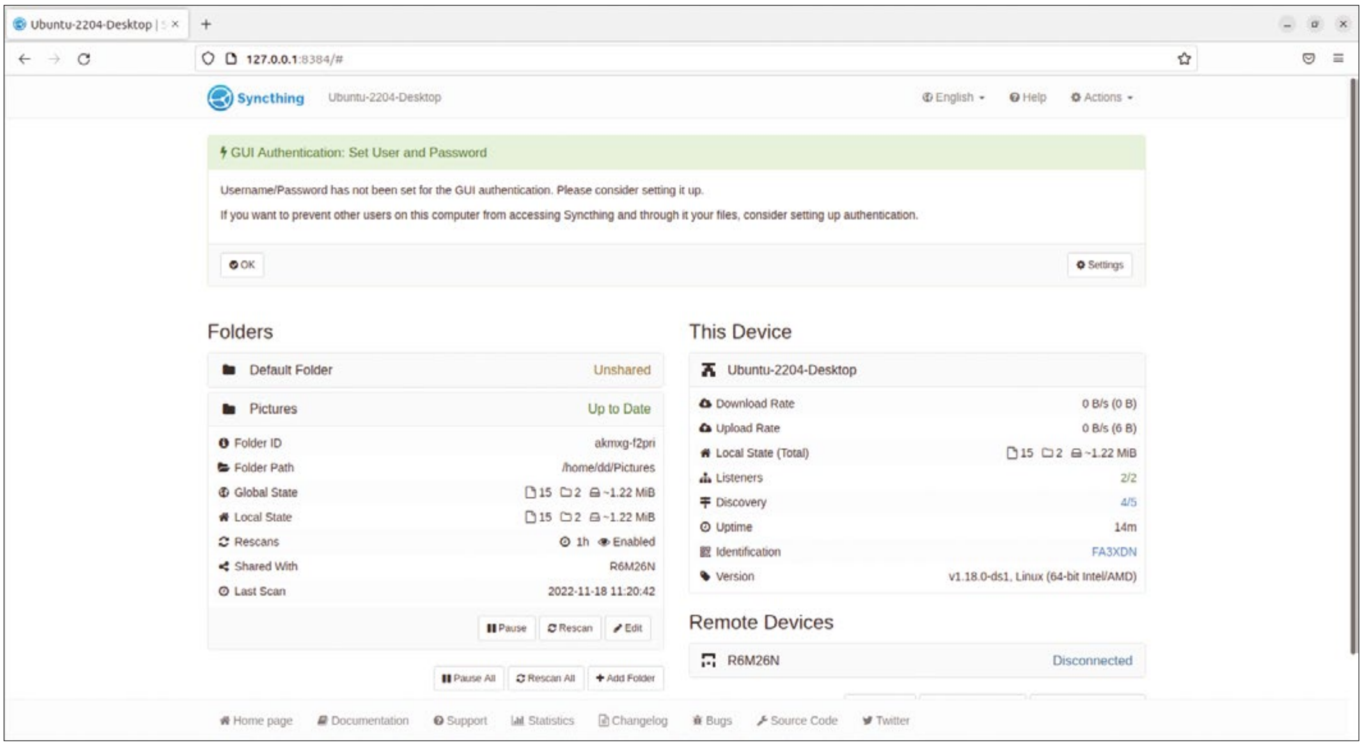
The device ID to enter here can be found in the "Actions > Show ID" dialog on the other device. Spaces and dashes are optional (ignored).
When adding a new device, keep in mind that this device must be added on the other side too.

Device Name

Shown instead of Device ID in the cluster status. Will be updated to the name the device advertises if left empty.

Save | Close

Figure 1: Syncthing's setup dialog will automatically find other devices on the local network.


Ubuntu-2204-Desktop

127.0.0.1:8384/#

Syncthing Ubuntu-2204-Desktop English Help Actions

GUI Authentication: Set User and Password

Username/Password has not been set for the GUI authentication. Please consider setting it up.

If you want to prevent other users on this computer from accessing Syncthing and through it your files, consider setting up authentication.

OK Settings

Folders

Default Folder	Unshared
Pictures	Up to Date
Folder ID	akmxg-f2pri
Folder Path	/home/dd/Pictures
Global State	15 2 ~1.22 MiB
Local State	15 2 ~1.22 MiB
Rescans	1h Enabled
Shared With	R6M26N
Last Scan	2022-11-18 11:20:42

Pause Rescan Edit

Pause All Rescan All Add Folder

This Device

Ubuntu-2204-Desktop	
Download Rate	0 B/s (0 B)
Upload Rate	0 B/s (6 B)
Local State (Total)	15 2 ~1.22 MiB
Listeners	2/2
Discovery	4/5
Uptime	14m
Identification	FA3XDN
Version	v1.18.0-ds1, Linux (64-bit Intel/AMD)

Remote Devices

R6M26N	Disconnected

Home page Documentation Support Statistics Changelog Bugs Source Code Twitter

Figure 2: Syncthing gives users detailed information on the status of devices and file transfers in the browser window if needed.

types you want the software to ignore during synchronization. The final tab, *Advanced*, is where to define the scan intervals.

After completing the configuration, press *Save*. On the terminal device with which you want to synchronize the folder, you are now prompted in the browser whether you want to let Syncthing create the new folder. If so, another dialog box appears and prompts you for the path.

A Word of Caution: Outdated GTK Front End

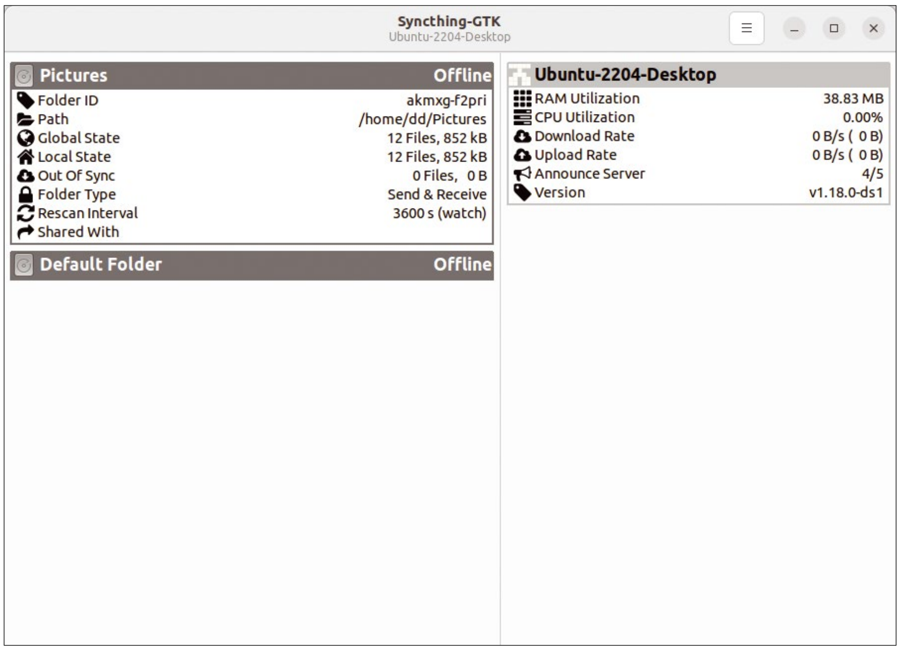
A GTK front end for Syncthing has been around for a long time (Figure 3). You will find it, with approximately the same version number, in the repositories of various distributions. Unfortunately, this front end has not been independently maintained for over three years. Some restrictions now apply to operations due to incompatibility with a number of dependencies. The matching Flatpak can also be installed and started, but without full functionality. My recommendation is to opt for the web interface and avoid this GTK front end.

As soon as data in one of the shared folders changes, the application synchronizes the data between the computers. A click on the directory in the browser window opens a status menu showing you the software's actions. This summary includes the number of files transferred and the total amount of data transferred (Figure 2).

If you connect multiple devices to Syncthing, unavailable systems will also appear in the list and will be shown with the *Disconnected* status. If the software is activated by the system rebooting, it automatically connects to other Syncthing instances on the intranet and synchronizes the data. Syncthing then replicates any changes following your settings. The status indicators in the web browser let you track the tool's activities.

Conclusions

Syncthing helps you keep your personal data in sync across several different devices on your local network. Since the software works in peer-to-peer mode, it does not require a dedicated server or a cloud subscription. The tool works quickly and reliably in the background; you only need the web interface for configuration changes (see the "A Word of Caution: Outdated GTK Front End" box). All told, Syncthing is a great solution for anyone wanting to quickly synchronize data across platforms and devices. ■■■



Pictures	Offline
Folder ID	akmxg-f2pri
Path	/home/dd/Pictures
Global State	12 Files, 852 kB
Local State	12 Files, 852 kB
Out Of Sync	0 Files, 0 B
Folder Type	Send & Receive
Rescan Interval	3600 s (watch)
Shared With	

Default Folder	Offline

Ubuntu-2204-Desktop	
RAM Utilization	38.83 MB
CPU Utilization	0.00%
Download Rate	0 B/s (0 B)
Upload Rate	0 B/s (0 B)
Announce Server	4/5
Version	v1.18.0-ds1

Figure 3: Unusable due to lack of maintenance: the GTK-based front end for Syncthing.

Info

[1] Syncthing: *https://syncthing.net*

[2] Syncthing Downloads: *https://syncthing.net/downloads/*

Author

Erik Bärwaldt is a self-employed IT admin and technical author living in United Kingdom. He writes for several IT magazines.

Simplicity

Restic and the Jarg front end appeal to users who want to see quick results without too much overhead.

By Erik Bärwaldt

Restic [1] is a cross-platform backup tool that is simple, quick, and easy to learn. Versions of Restic run on macOS, Windows, and BSD, as well as Linux. A graphical front end called Jarg [2] extends Restic to users who are more comfortable in a GUI.

Restic focuses on flexibility and ease of handling. Instead of entering long lines of cryptic parameters, you can save data with just a few simple commands. But the tool still keeps all options open for you. You can preserve your data in the cloud and also on local media. If you need to, you can use SFTP to control a file server on the LAN as a storage location. Because Restic is implemented as a command-line tool, you can include it in scripts and time-control jobs with cron.

Restic automatically encrypts the backups in line with the AES-256 standard. It can handle incremental backups where it will only save the changed data after an initial full backup. Restic also offers a deduplication feature to avoid redundant data. The ability to verify backups helps users check that the data is backed up without error.

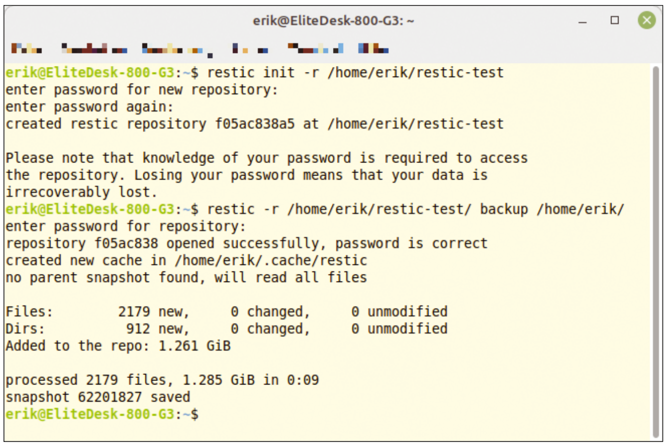
Restic is available in the software archives of all major Linux distributions. You can pick up the software for other architectures and operating systems from the project's GitHub page.

Ready to Start

Restic relies on repositories. Two commands are all it takes to back up data to your local media. The first command (Listing 1, line 2) creates the repository for the backup. Restic automatically generates the matching directory and prompts you for a password to encrypt and decrypt the backups. The second command (line 3) pushes a backup of the specified folder, including all subdirectories, into the new repository. Restic also prompts you for a password before proceeding to create the backup (Figure 1).

Listing 1: Backing Up with Restic

```
01 ### local
02 $ restic init -r /path_to_repository
03 $ restic -r /path_to_repository backup /path_to_data
04 ### remote
05 $ restic init -r sftp:User@Host:/path_to_repository.
06 $ restic -r sftp:User@Host:/path_to_repository. backup /path_to_backup_data.
```



```
erik@EliteDesk-800-G3:~$ restic init -r /home/erik/restic-test
enter password for new repository:
enter password again:
created restic repository f05ac838a5 at /home/erik/restic-test

Please note that knowledge of your password is required to access
the repository. Losing your password means that your data is
irrecoverably lost.
erik@EliteDesk-800-G3:~$ restic -r /home/erik/restic-test/ backup /home/erik/
enter password for repository:
repository f05ac838 opened successfully, password is correct
created new cache in /home/erik/.cache/restic
no parent snapshot found, will read all files

Files:        2179 new,     0 changed,     0 unmodified
Dirs:          912 new,     0 changed,     0 unmodified
Added to the repo: 1.261 GiB

processed 2179 files, 1.285 GiB in 0:09
snapshot 62201827 saved
erik@EliteDesk-800-G3:~$
```

Figure 1: Restic quickly creates the desired backup and provides detailed information.

If necessary, you can extend the command line to include additional parameters. For example, `--verbose` outputs various detailed messages during the backup. To back up multiple directories to different paths in a single pass, you need to enter the paths as a space-separated list. The `--exclude` and `--iexclude` parameters exclude individual directories or entire paths from the backup.

For repetitive, larger backup tasks, you can enter the list of directories or files you wish to include or exclude in a text file. To back up to a server on the LAN, you just enter the same commands as for a local backup, but add the prefixed computer name and the protocol used to the target path (Listing 1, lines 5 and 6).

The `--keep` and `--keep-last` parameters let you define how many of the last backups are kept. Additional parameters for automated removal of backups are listed in the Restic documentation [3].

Detailed

For encrypted backups, you can't view the data inside the destination folder directly. To look inside, you need to mount the repository like a normal drive. First create a separate mount folder, and then run the command from the line 2 of Listing 2.

Enter the password at the prompt for Restic to mount the backup repository and decrypt the data files in the target directory. You can now view the data in the `snapshots/` subdirectory of the mount folder, but editing is not allowed. Restic remains active in the terminal. The keyboard shortcut Ctrl + C lets you unmount the drive again, and the data will disappear from the target folder.

Comparing

You can also use Restic to compare data sets or check the integrity of the folders. Use the `diff` and `check` parameters. `check` is used to check the index and contents of the specified repository (Listing 2, line 4).

To compare two or more snapshots, you need their identification numbers, which you can discover with the command from line 5 of Listing 2. The ID numbers appear in a table below each other, along with the timestamp of the backup (Figure 2). You can now identify the snapshots and compare them directly (line 6).

Restoring Data

The `restore` parameter is used to restore data from the backup. First determine the ID number of the snapshot you need (Listing 3, line 1), and then initiate the restore (line 2). You can also specify a new folder that does not yet exist as the target path for the restore. If the target directory does not yet exist, Restic will create it automatically without further prompting.

Graphical Front Ends

Restic has several graphical front ends, but some are no longer under active development or are only sporadically maintained. Jarg is one of the sporadically maintained options. The last Jarg update was in September 2021. The name *Jarg* is an acronym for Just Another Restic GUI.

The Jarg front end covers the basic functionality of Restic and is available for download exclusively in source code or as an AppImage for 64-bit systems. Jarg has not yet found its way into the package sources of the common distributions.

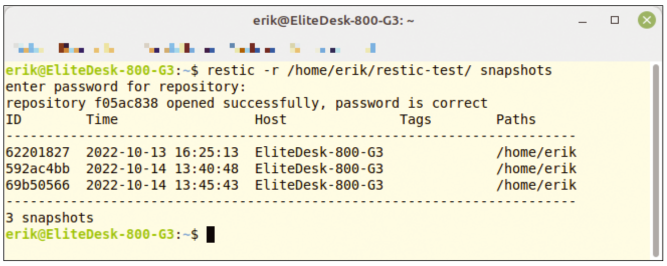
After downloading the Jarg AppImage file, first grant it execution permission with the command:

```
chmod +x Jarg.0.2.0.AppImage
```

Then call the Jarg front end by typing the following at the prompt:

```
./Jarg.0.2.0.AppImage
```

The main window shows some information about the profile and four buttons for operation (Figure 3). Clicking on the pencil icon opens a small settings menu. The settings window is where you specify the path to the repository for the backups, the directories you wish to back up, and the matching



```
erik@EliteDesk-800-G3:~$ restic -r /home/erik/restic-test/ snapshots
enter password for repository:
repository f05ac838 opened successfully, password is correct
ID        Time                 Host                Tags        Paths
-----------------------------------------------------------------------
62201827  2022-10-13 16:25:13  EliteDesk-800-G3                /home/erik
592ac4bb  2022-10-14 13:40:48  EliteDesk-800-G3                /home/erik
69b50566  2022-10-14 13:45:43  EliteDesk-800-G3                /home/erik
-----------------------------------------------------------------------
3 snapshots
erik@EliteDesk-800-G3:~$
```

Figure 2: Restic identifies snapshots with unique IDs.

Listing 2: Checking Backups

```
01 ### Mount repository
02 $ restic -r /path_to_repository mount /path_to_mount_point
03 ### Check and compare repositories
04 $ restic -r /path_to_repository check --read-data
05 $ restic -r /path_to_repository snapshots
06 $ restic -r /path_to_repository diff Snapshot1_ID Snapshot2_ID ...
```

Listing 3: Restoring Data

```
01 $ restic -r /path_to_repository snapshots
02 $ restic -r /path_to_repository restore -t /path_to_restore ID_number
```

password. The repository can be on the local system, or it can reside on a remote server.

Save the settings by pressing the *Save* button, and click *Initialize*. Then trigger the backup by pressing the *Backup* button. Jarg displays Restic's runtime messages on the right side of the gray window segment (Figure 4). The messages help you identify errors immediately so you can modify the profile accordingly.

Clicking on the *List* button brings up a list of existing snapshots, including their ID numbers, so you can easily reconstruct the datasets using the *Restore* dialog.

Profiles

You can create additional profiles for different backup tasks by clicking *default* in the configuration dialog. Choose the *new* entry in the selection menu. Then specify a name for the new profile in the name field, fill out the paths and password fields, and save the new profile. After that, you can select the profile whenever you run the program.

Conclusions

Restic takes the horror out of regular backup runs and can be used at a glance, even without in-depth knowledge of backup concepts and strategies. The tool does its job quickly and reliably. All told, Restic and the Jarg front end offer home users and administrators a practical solution for structured data backups. ■■■

Info

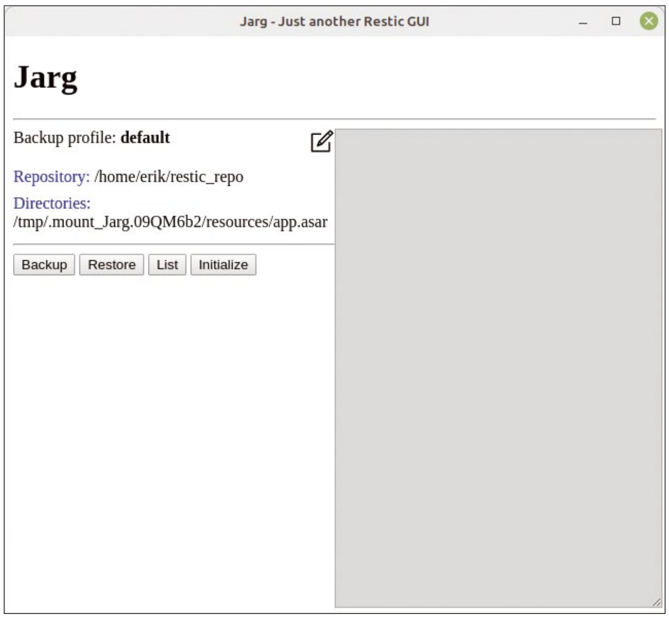
[1] Restic: *https://restic.net*

[2] Jarg: *https://github.com/rgwch/jarg*

[3] Automatic deletion: *https://restic.readthedocs.io/en/stable/060_forget.html#removing-snapshots-according-to-a-policy*

Author

Erik Bärwaldt is a self-employed IT admin and technical author living in United Kingdom. He writes for several IT magazines.

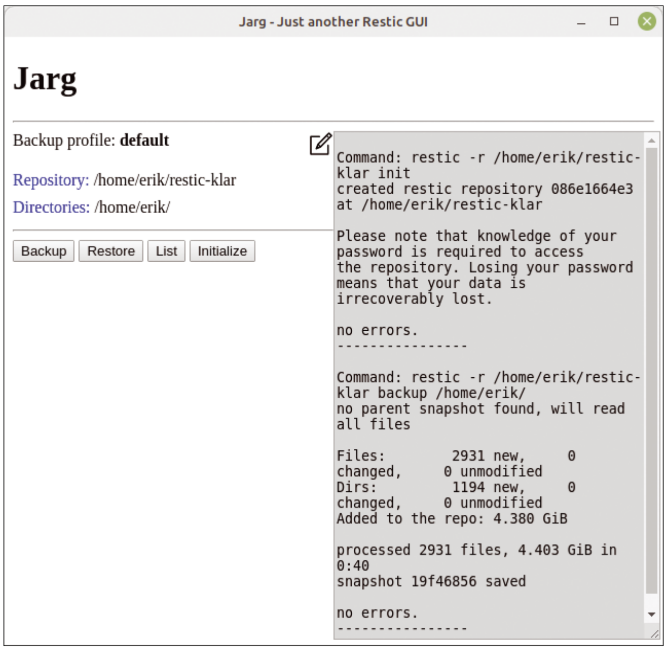


Backup profile: **default**

Repository: /home/erik/restic_repo

Directories: /tmp/.mount_Jarg.09QM6b2/resources/app.asar

Figure 3: Jarg provides a simple graphical front end for Restic.



Backup profile: **default**

Repository: /home/erik/restic-klar

Directories: /home/erik/

```
Command: restic -r /home/erik/restic-
klar init
created restic repository 086e1664e3
at /home/erik/restic-klar

Please note that knowledge of your
password is required to access
the repository. Losing your password
means that your data is
irrecoverably lost.

no errors.
----------------

Command: restic -r /home/erik/restic-
klar backup /home/erik/
no parent snapshot found, will read
all files

Files:        2931 new,     0
changed,     0 unmodified
Dirs:         1194 new,     0
changed,     0 unmodified
Added to the repo: 4.380 GiB

processed 2931 files, 4.403 GiB in
0:40
snapshot 19f46856 saved

no errors.
----------------
```

Figure 4: Restic messages appear on the right in Jarg.

Whatever happened to Mandrake?

OpenMandriva Lx

Mandrake lives on as OpenMandriva Lx. Bruce talks to OpenMandriva Council members to find out more about this innovative distribution. *By Bruce Byfield*

Long-time Linux users may recall the once popular Mandrake Linux, but, in North America, any traces of Mandrake have almost disappeared from public view. However, in Europe, the story is different. The once popular distribution has several descendants. In particular, its direct legal descendant is OpenMandriva Lx [1]. Wanting to learn more, I asked for more information on the OpenMandriva forum. Here is what I learned.

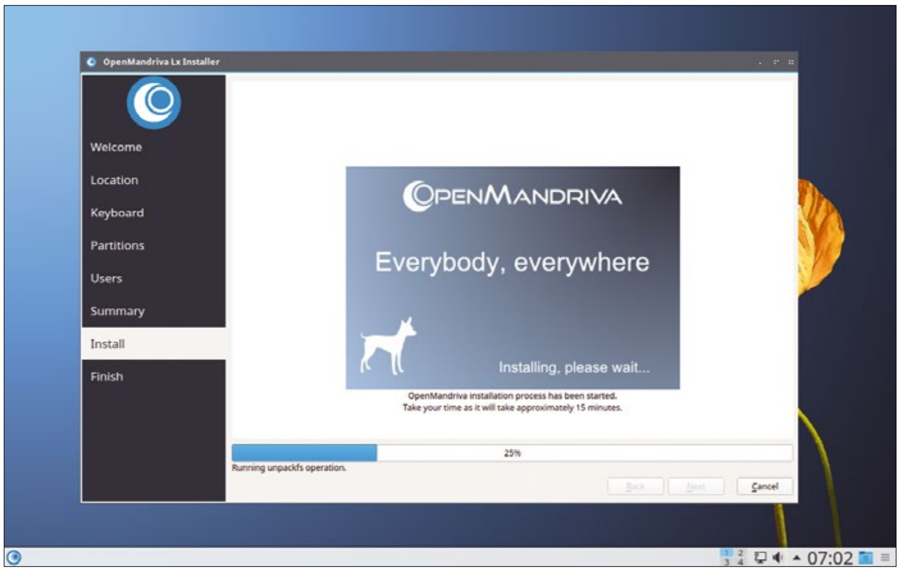
Back Story

OpenMandriva Lx's history is complicated. Around the turn of the millennium, Mandrake Linux was a popular fork of Red Hat Linux. Mandrake quickly became one of the top half dozen commercial distributions, thanks mainly to the fact that it was one of the first to provide desktop utilities. However, Mandrake's name conflicted with that of the *Mandrake the Magician* comic, and in 2005, Mandrake merged with the Connectiva distribution to become Mandriva SA. Mandriva was forked by Mageia Linux and ROSA Linux, but when it went into receivership in 2015, it formally transferred "a non-exclusive and irrevocable worldwide license" [2] of its intellectual property to OpenMandriva SA, an alliance of previous Mandriva contributors and people from related projects, including Unity Linux and Ark Linux. In turn, OpenMandriva became the association that has developed OpenMandriva Lx ever since. As OpenMandriva Chairman Bernhard Rosenkränzer (aka bero) explains, despite sharing common origins, OpenMandriva is completely separate from other forks.

Today, OpenMandriva is governed by its Council that oversees legal issues, public relations, and general organization and the Technical Committee that is responsible for development. Members of both are invited to join, rather than be elected, and decisions are made by consensus whenever possible. Currently OpenMandriva Lx has seven main developers, plus a few irregular ones, as well as two mascots, Chwido and Laska. In the last year, there were 82,350 commits, according to bero.

On the one hand, the OpenMandriva home page [1] states that "our roots are in Mandrake and its traditions," which, according to Council member rugyada, consists of the original source code, part

Photo by corina ardeleanu on Unsplash



OpenMandriva installation process has been started.
Take your time as it will take approximately 15 minutes.

Running unpackfs operation.

Figure 1: **Like its ancestor Mandrake, OpenMandriva Lx uses its own tools for installation and administration.**

of the initial development team, and "the idea of building an operating system that is simple enough for someone who has never seen Linux to get productive with, without dumbing it down to the point that it stops being useful to experts."

On the other hand, OpenMandriva Lx has also evolved in new directions. As soon as the agreement with Mandriva SA was finalized, the distribution began to focus on new technologies. Today, most of the desktop tools have been replaced with newer ones that use different core libraries. As well, the core system has evolved, with the main toolchain being built on Clang and ported to new architectures. Currently, bero says, "we are rethinking the filesystem layout to make working across different CPU architectures easier. With qemu-static and binfmt, it shouldn't be a problem to install x86-specific binaries (think Wine) on AArch64 or vice versa, or to cross-compile even packages that don't support cross-compiling" (Figures 1 and 2).

The Audience

For all users, OpenMandriva Lx offers both a point release (Rock) and a rolling release (ROME). As bero writes, users can choose "if they want the latest and greatest features at all times (while still getting some testing) … if they want a user experience that is essentially guaranteed not to change until they choose to update to a new point release."

Since its early stages in 2011, OpenMandriva's target audience has been desktop users. For these users, OpenMandriva offers the ease of installation and a default installation "with pretty much everything we think a typical desktop user will need. You will also find a system that performs well, even on older hardware," writes bero. Increasingly, though OpenMandriva is also aimed at developers and more experienced users, offering an up-to-date Clang toolchain, current kernels, and cross-compilers such as fiptool and github-cli.

More recently, the distribution has started to focus on server users as well. Bero writes that "we have been working with Ampere to get OMLx [OpenMandriva Lx] running on their AArch64 servers and inside cloud services running on them, such as Oracle Cloud. Here our aim is to prove that stable is spelled with a 'b' in the middle – and doesn't have to imply 'stale'. We believe a system can be stable without relying on technologies that are a few years behind the time. Of course we also support x86 servers." In addition, "server users may well appreciate current versions of the PHP and Java stacks and always up-to-date PostgreSQL and MariaDB, our docker containers, and the performance (booting our Ampere Altra server with the preloaded CentOS took more than seven minutes – which we cut down to less than four by installing an OpenMandriva snapshot)."

Hardware Support

Like most distributions, OpenManriva Lx supports a broad range of hardware. However, out of a wish to remain open source, the distribution recommends avoiding NVIDIA graphics, although the nouveau graphics are available. Among the supported architectures are a variety of AArch64 (ARM) and Raspberry devices, as well as Pinebook Pro or a high-end Ampere Altra server. No UEFI support is required for AArch64 devices, on the grounds that to do so would limit hardware support. Instead, OpenMandriva builds a custom image during installation.

In the near future, OpenMandriva expects some innovative support. Both a port to Pine Phone Pro and support for cheap hardware are in their early stages, and, most important, support for the open hardware RISC-V chips, suspended

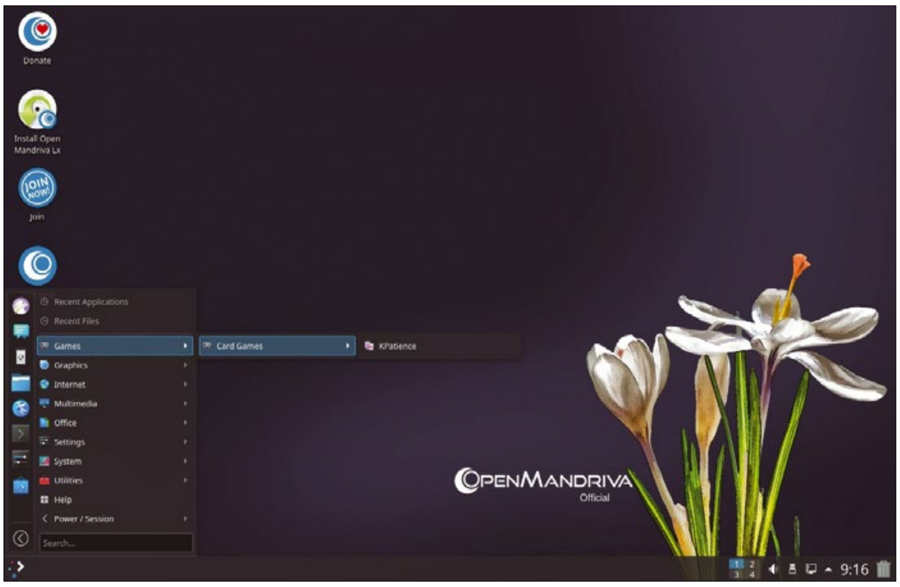


Figure 2: **The OpenMandriva Lx desktop.**

after problems with the first SiFive Unlimited boards, is expected to resume after the next general release using SiFive Unmatched VisionFive boards. "Even though there is currently little real world hardware available," bero says, "RISC-V is a very interesting target to us, primarily because of its open nature." Although no one from OpenMandriva has said so, the distribution may well be the first to support RISC-V chips, and thus the first to permit systems in which both the software and the hardware are open.

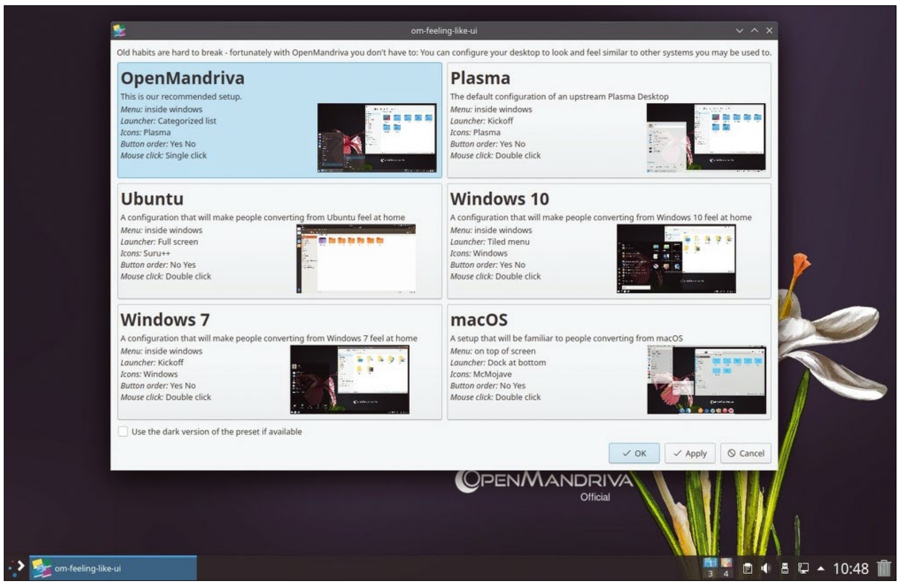
A Tradition of Innovation

Asked what might attract users to OpenMandriva Lx, bero notes that, unlike many distros, it is not a derivative, and that "this gives us more room to be innovative and go new ways."

While OpenMandriva does not hesitate to borrow from other distributions when convenient, it definitely takes full advantage of its independence. For example, in 2016, OpenMandriva was the first distribution to be built with Clang as its main compilers, even before Android. Similarly, "using a different implementation of the `tar` command, we are exploring a more cross-compiler friendly filesystem layout making sysroots more useful than ever, and we have started experimenting with builds using alternative libc implementations. The distribution has also started to use an automated update tool for packages, which helps to ensure that even neglected packages are updated. On the desktop, the custom-built tools include Desktop Presets, a tool that can make OpenMandriva look more like Windows, macOS, or another Linux distribution" (Figure 3).

So, in answer to my original question, what happened to Mandrake? The answer is nothing. In OpenMandriva, Mandrake's spirit lives on, the biggest change being the name. In 2023, the Mandrake branch of Linux is still going its own way and adding innovations to open source.

With thanks to rugyada, raphael, bero, and ben79. ■■■


om-feeling-like-ui

Old habits are hard to break - fortunately with OpenMandriva you don't have to: You can configure your desktop to look and feel similar to other systems you may be used to.

OpenMandriva
This is our recommended setup.
Menu: inside windows
Launcher: Categorized list
Icons: Plasma
Button order: Yes No
Mouse click: Single click

Plasma
The default configuration of an upstream Plasma Desktop
Menu: inside windows
Launcher: Kickoff
Icons: Plasma
Button order: Yes No
Mouse click: Double click

Ubuntu
A configuration that will make people converting from Ubuntu feel at home
Menu: inside windows
Launcher: Full screen
Icons: Suru++
Button order: No Yes
Mouse click: Double click

Windows 10
A configuration that will make people converting from Windows 10 feel at home
Menu: inside windows
Launcher: Tiled menu
Icons: Windows
Button order: Yes No
Mouse click: Double click

Windows 7
A configuration that will make people converting from Windows 7 feel at home
Menu: inside windows
Launcher: Kickoff
Icons: Windows
Button order: Yes No
Mouse click: Double click

macOS
A setup that will be familiar to people converting from macOS
Menu: on top of screen
Launcher: Dock at bottom
Icons: McMojave
Button order: No Yes
Mouse click: Double click

☐ Use the dark version of the preset if available

Figure 3: Like Zorin, OpenMandriva LX can mimic other operating systems as an aid to new users.

Info

[1] OpenMandriva: *https://www.openmandriva.org/*

[2] Transfer of intellectual property: *https://wiki.openmandriva.org/en/policies/oma-tm-licence-en*

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/*.

■■■

Open Source
is more than tech

Find jobs in sales, marketing,
customer support, and more!

OpenSource
JOB HUB

opensourcejobhub.com/

An improved front end for Apt

Apt on Steroids

The Nala front end for Debian's Apt package manager combines the reliability of the Apt tools with easier to read output and speedier downloads. *By Ferdinand Thommes*

Debian's package management system for the command line relies on dpkg [1] in the background and the Advanced Package Tool (Apt) [2] with its default front end, `apt`. While `apt` usually does a reliable job, its command-line tools (based on the *libapt-pkg* and *libapt-inst* libraries), such as `apt`, `apt-file`, `apt-policy`, and the like, are not particularly clear-cut or attractive. In addition, the Apt front end is slow.

This, along with other shortcomings in the tools available in Debian, prompted longtime Debian developer, Michael Stapelberg, to announce his retirement in 2019 with some harsh criticism [3]. Stapelberg subsequently created the proof of concept distri distribution, with the goal of optimizing package management in general (*Linux Magazine* previously covered distri in July 2021 [4]).

Nala, which was inspired by Fedora's DNF package manager, steps up to eradicate these shortcomings, offering easier to read output and increased speed thanks to parallel downloading.

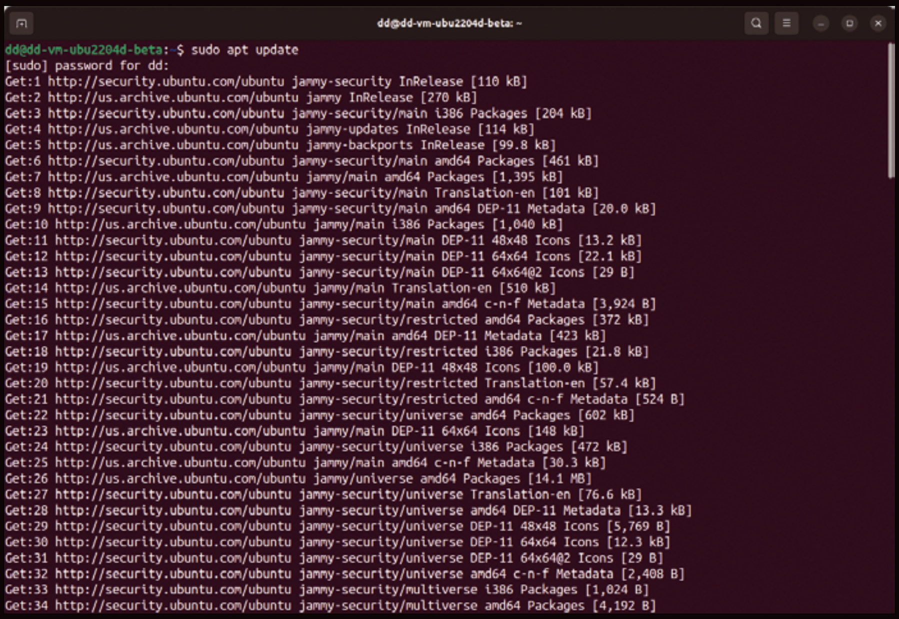
What Apt Does

As part of Apt on Debian, Ubuntu, and their derivatives, `apt` is the interactive command-line utility that takes care of package management. It performs tasks such as installing, uninstalling, or searching for packages and displaying information about them. While `apt` does not offer color-highlighted output (Figure 1), it does at least have a progress bar. You can interact with `apt` through subcommands such as `update`, `upgrade`, `full-upgrade`, `install`, `remove`, and `purge`. The packages installed via `apt` are based on the repositories entered in the source list below `/etc/apt/sources.list.d/`.

Since the beginning of the year, another Apt front end named Nala has been making the rounds. Unlike `apt`, Nala uses the `python-apt` [5] library. This library provides access to almost all the features supported by Apt's underpinnings, the *libapt-pkg* and *libapt-inst* libraries. It's only logical that front-end programs for Apt, such as Nala, can be written in Python.

Although still quite new, Nala can already be found in the repositories of distributions such as Debian Testing and Unstable, Kali Linux, MX Linux, Raspbian Testing, and Ubuntu 22.04 backports. For other Debian derivatives, you need to include the developer's repository as described in the wiki on GitLab [6]. Alternatively, you can download the tool as a DEB package and install it directly.

Currently, Nala version 0.11.1 can be used in the Bash, ZFS, and Fish shells.



```
dd@dd-vm-ubu2204d-beta:~$ sudo apt update
[sudo] password for dd:
Get:1 http://security.ubuntu.com/ubuntu jammy-security InRelease [110 kB]
Get:2 http://us.archive.ubuntu.com/ubuntu jammy InRelease [270 kB]
Get:3 http://security.ubuntu.com/ubuntu jammy-security/main i386 Packages [204 kB]
Get:4 http://us.archive.ubuntu.com/ubuntu jammy-updates InRelease [114 kB]
Get:5 http://us.archive.ubuntu.com/ubuntu jammy-backports InRelease [99.8 kB]
Get:6 http://security.ubuntu.com/ubuntu jammy-security/main amd64 Packages [461 kB]
Get:7 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 Packages [1,395 kB]
Get:8 http://security.ubuntu.com/ubuntu jammy-security/main Translation-en [101 kB]
Get:9 http://security.ubuntu.com/ubuntu jammy-security/main amd64 DEP-11 Metadata [20.0 kB]
Get:10 http://us.archive.ubuntu.com/ubuntu jammy/main i386 Packages [1,040 kB]
Get:11 http://security.ubuntu.com/ubuntu jammy-security/main DEP-11 48x48 Icons [13.2 kB]
Get:12 http://security.ubuntu.com/ubuntu jammy-security/main DEP-11 64x64 Icons [22.1 kB]
Get:13 http://security.ubuntu.com/ubuntu jammy-security/main DEP-11 64x64@2 Icons [29 B]
Get:14 http://us.archive.ubuntu.com/ubuntu jammy/main Translation-en [510 kB]
Get:15 http://security.ubuntu.com/ubuntu jammy-security/main amd64 c-n-f Metadata [3,924 B]
Get:16 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 Packages [372 kB]
Get:17 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 DEP-11 Metadata [423 kB]
Get:18 http://security.ubuntu.com/ubuntu jammy-security/restricted i386 Packages [21.8 kB]
Get:19 http://us.archive.ubuntu.com/ubuntu jammy/main DEP-11 48x48 Icons [100.0 kB]
Get:20 http://security.ubuntu.com/ubuntu jammy-security/restricted Translation-en [57.4 kB]
Get:21 http://security.ubuntu.com/ubuntu jammy-security/restricted amd64 c-n-f Metadata [524 B]
Get:22 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 Packages [602 kB]
Get:23 http://us.archive.ubuntu.com/ubuntu jammy/main DEP-11 64x64 Icons [148 kB]
Get:24 http://security.ubuntu.com/ubuntu jammy-security/universe i386 Packages [472 kB]
Get:25 http://us.archive.ubuntu.com/ubuntu jammy/main amd64 c-n-f Metadata [30.3 kB]
Get:26 http://us.archive.ubuntu.com/ubuntu jammy/universe amd64 Packages [14.1 MB]
Get:27 http://security.ubuntu.com/ubuntu jammy-security/universe Translation-en [76.6 kB]
Get:28 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 DEP-11 Metadata [13.3 kB]
Get:29 http://security.ubuntu.com/ubuntu jammy-security/universe DEP-11 48x48 Icons [5,769 B]
Get:30 http://security.ubuntu.com/ubuntu jammy-security/universe DEP-11 64x64 Icons [12.3 kB]
Get:31 http://security.ubuntu.com/ubuntu jammy-security/universe DEP-11 64x64@2 Icons [29 B]
Get:32 http://security.ubuntu.com/ubuntu jammy-security/universe amd64 c-n-f Metadata [2,408 B]
Get:33 http://security.ubuntu.com/ubuntu jammy-security/multiverse i386 Packages [1,024 B]
Get:34 http://security.ubuntu.com/ubuntu jammy-security/multiverse amd64 Packages [4,192 B]
```

Figure 1: With `apt`, the output for updating the source list is a mess of information without any clear focus on important facts.

For Debian Stable, Ubuntu 21.04, and distributions based on their package set, you need to use the *nala-legacy* package, which provides the same functions and is only adapted to the older package set in terms of dependencies. You cannot use Nala with older distributions, such as Ubuntu 18.04 or Debian 10 (buster).

What Nala Offers

Because of the many changes, it is important to be able to intuitively understand the package manager's output during package installations or updates of the entire installation at the command line – especially if you use Debian Testing or Unstable, or other rolling release distributions based on DEB packages. This desire for clarity is one of the starting points for Nala: The output should be more accessible, detailed where necessary, and unobtrusive where possible. Nala achieves this goal by using more color highlighting and clearer output formatting (Figure 2).

When called, Nala acts a little different from `apt`. The command

```
sudo nala update
```

has the same result as

```
sudo apt update
```

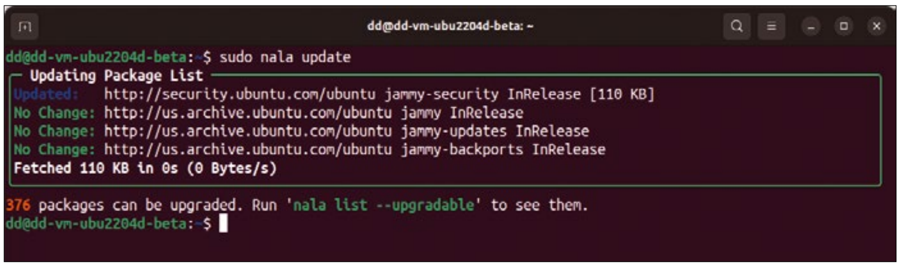
It updates the sources list in `etc/apt/sources.list.d`. On the other hand, if you type

```
sudo nala upgrade
```

Nala will run an

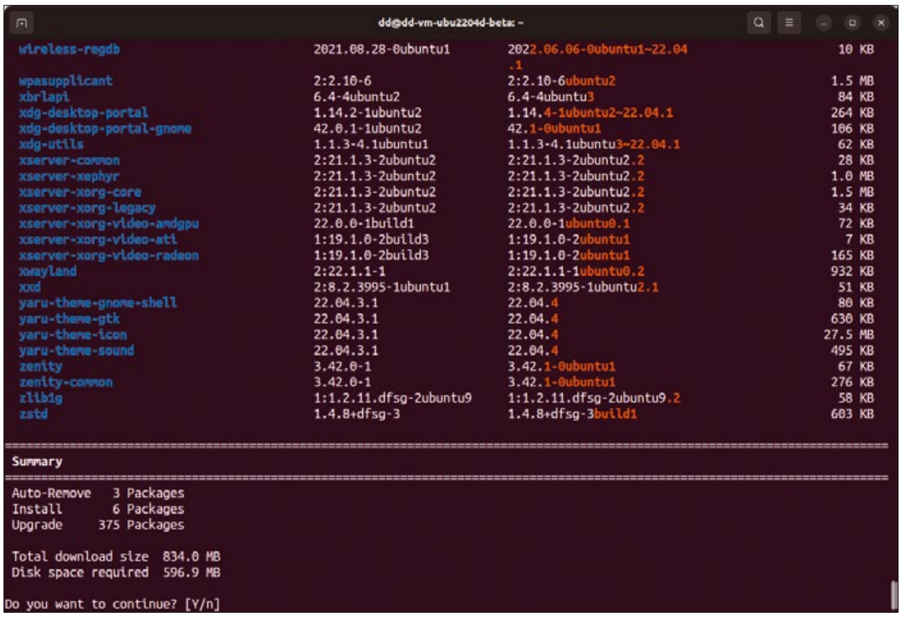
```
apt update && apt full-upgrade
```

as shown in Figure 3. This is because the developers designed Nala with rolling releases in mind. However, the command works just as well on stable distributions.



```
dd@dd-vm-ubu2204d-beta:~$ sudo nala update
 Updating Package List
Updated:    http://security.ubuntu.com/ubuntu jammy-security InRelease [110 KB]
No Change: http://us.archive.ubuntu.com/ubuntu jammy InRelease
No Change: http://us.archive.ubuntu.com/ubuntu jammy-updates InRelease
No Change: http://us.archive.ubuntu.com/ubuntu jammy-backports InRelease
Fetched 110 KB in 0s (0 Bytes/s)

376 packages can be upgraded. Run 'nala list --upgradable' to see them.
dd@dd-vm-ubu2204d-beta:~$ 
```

Figure 2: With Nala, you can immediately see which repositories have been updated since the last update. In addition, the output also highlights ignored sources.



```
 wireless-regdb                 2021.08.28-0ubuntu1        2022.06.06-0ubuntu1~22.04          10 KB
                                                           .1
 wpasupplicant                  2:2.10-6                   2:2.10-6ubuntu2                   1.5 MB
 xbrlapi                        6.4-4ubuntu2               6.4-4ubuntu3                       84 KB
 xdg-desktop-portal             1.14.2-1ubuntu2            1.14.4-1ubuntu2~22.04.1           264 KB
 xdg-desktop-portal-gnome       42.0.1-1ubuntu2            42.1-0ubuntu1                     106 KB
 xdg-utils                      1.1.3-4.1ubuntu1           1.1.3-4.1ubuntu3~22.04.1           62 KB
 xserver-common                 2:21.1.3-2ubuntu2          2:21.1.3-2ubuntu2.2                28 KB
 xserver-xephyr                 2:21.1.3-2ubuntu2          2:21.1.3-2ubuntu2.2               1.0 MB
 xserver-xorg-core              2:21.1.3-2ubuntu2          2:21.1.3-2ubuntu2.2               1.5 MB
 xserver-xorg-legacy            2:21.1.3-2ubuntu2          2:21.1.3-2ubuntu2.2                34 KB
 xserver-xorg-video-amdgpu      22.0.0-1build1             22.0.0-1ubuntu0.1                  72 KB
 xserver-xorg-video-ati         1:19.1.0-2build3           1:19.1.0-2ubuntu1                   7 KB
 xserver-xorg-video-radeon      1:19.1.0-2build3           1:19.1.0-2ubuntu1                 165 KB
 xwayland                       2:22.1.1-1                 2:22.1.1-1ubuntu0.2               932 KB
 xxd                            2:8.2.3995-1ubuntu1        2:8.2.3995-1ubuntu2.1              51 KB
 yaru-theme-gnome-shell         22.04.3.1                  22.04.4                            80 KB
 yaru-theme-gtk                 22.04.3.1                  22.04.4                           630 KB
 yaru-theme-icon                22.04.3.1                  22.04.4                          27.5 MB
 yaru-theme-sound               22.04.3.1                  22.04.4                           495 KB
 zenity                         3.42.0-1                   3.42.1-0ubuntu1                    67 KB
 zenity-common                  3.42.0-1                   3.42.1-0ubuntu1                   276 KB
 zlib1g                         1:1.2.11.dfsg-2ubuntu9     1:1.2.11.dfsg-2ubuntu9.2           58 KB
 zstd                           1.4.8+dfsg-3               1.4.8+dfsg-3build1                603 KB

==================================================================================================
 Summary
==================================================================================================
 Auto-Remove   3 Packages
 Install       6 Packages
 Upgrade     375 Packages

 Total download size  834.0 MB
 Disk space required  596.9 MB

Do you want to continue? [Y/n]
```

Figure 3: Nala highlights the new version of the packages to be upgraded in the right column when upgrading. The summary clearly shows the packages to be installed and upgraded, as well as the scope of the upgrade.

When updating, Nala highlights important aspects of the output in color, which shows at a glance which repositories offer new packages and to which version a package will be updated. In addition, the tool color highlights all of the repositories that it ignores due to an incorrect configuration or a missing key. While updating packages, Nala reliably shows you a progress bar with the remaining time, the completed workload as a percentage, and the number of packages already processed compared to the total update (Figure 4).

One of Nala's less desirable quirks is to additionally integrate

```
sudo nala upgrade
```

with

```
sudo apt autoremove
```

which removes packages that are no longer needed (Figure 5). As experience shows, packages that are actually needed will still disappear from time to time. The recommended approach is to prevent this behavior by calling Nala with the `--no-autoremove` parameter.

Alternatively, you can prevent this behavior permanently in the `/etc/nala/nala.conf` file by setting the second entry to `auto_remove = false`. If you want Nala to clean up the package inventory, use the `--autoremove` parameter. In the configuration file, you will find some additional simplifying settings that Apt does not offer, which includes hooks that automatically perform scripted actions before or after installing a package. A video on Reddit explains this feature and shows an example [7].

If `apt` is indelibly embedded in your finger muscle memory, you can create an alias and keep using the familiar commands while Nala is working in the background.

What Slows Apt Down

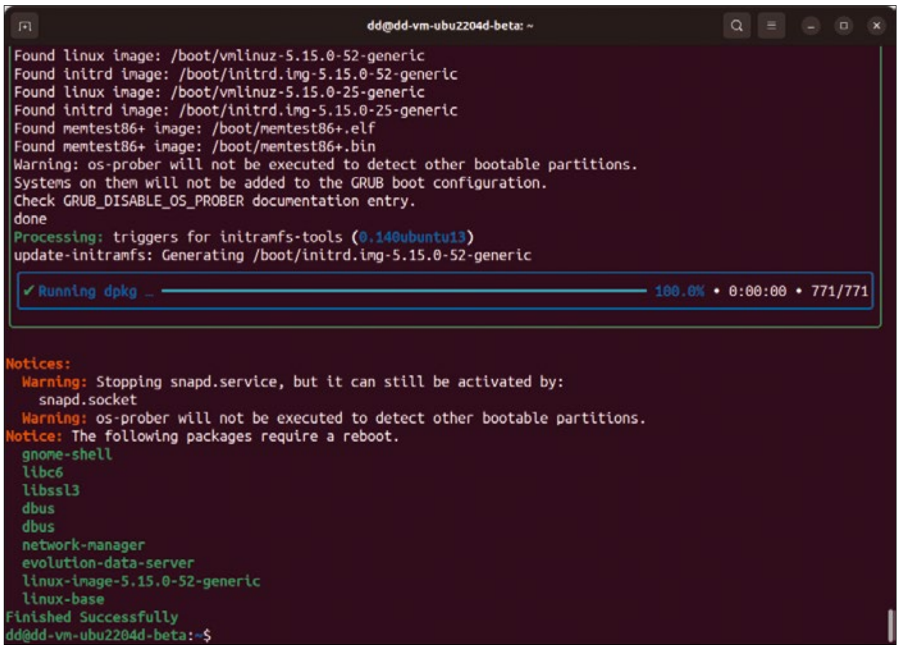
Since 2002, the Debian developers have been resisting allowing parallel downloading from the same server on the grounds that it puts too much load on the servers while offering little gain [8]. Today, in view of HTTP/2 and soon

HTTP/3, this argument needs to at least be verified. Other distributions have long since allowed parallel downloading from the same host. On Debian, you can install the `apt-fast` shell script wrapper from GitHub [9]; it drastically improves `apt`'s download times by downloading packages in parallel, with multiple connections per host.

Nala makes `apt-fast` obsolete: Parallel downloading is already part of Nala's strategy. By default, Nala downloads three packages per unique mirror server in your source list. Also, Nala alternates downloads between available mirrors to boost speed further. If a server fails for any reason, Nala tries the next one until it has processed all defined servers.

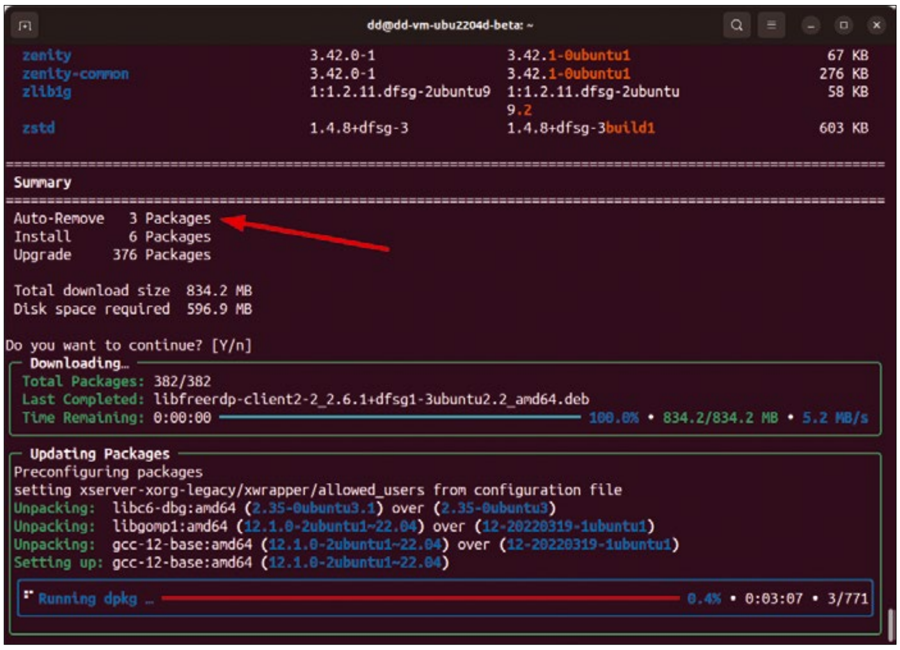
Listing 1: Specifying a Downgrade

```
$ awk '$1 == "2022-08-25" && $3 == "upgrade" {print $4"="$5}' /var/log/dpkg.log
```



```
dd@dd-vm-ubu2204d-beta: ~
Found linux image: /boot/vmlinuz-5.15.0-52-generic
Found initrd image: /boot/initrd.img-5.15.0-52-generic
Found linux image: /boot/vmlinuz-5.15.0-25-generic
Found initrd image: /boot/initrd.img-5.15.0-25-generic
Found memtest86+ image: /boot/memtest86+.elf
Found memtest86+ image: /boot/memtest86+.bin
Warning: os-prober will not be executed to detect other bootable partitions.
Systems on them will not be added to the GRUB boot configuration.
Check GRUB_DISABLE_OS_PROBER documentation entry.
done
Processing: triggers for initramfs-tools (0.140ubuntu13)
update-initramfs: Generating /boot/initrd.img-5.15.0-52-generic

 ✓ Running dpkg … ━━━━━━━━━━━━━━━━━━━━━━━━ 100.0% • 0:00:00 • 771/771

Notices:
  Warning: Stopping snapd.service, but it can still be activated by:
    snapd.socket
  Warning: os-prober will not be executed to detect other bootable partitions.
Notice: The following packages require a reboot.
  gnome-shell
  libc6
  libssl3
  dbus
  dbus
  network-manager
  evolution-data-server
  linux-image-5.15.0-52-generic
  linux-base
Finished Successfully
dd@dd-vm-ubu2204d-beta:~$
```

Figure 4: After successfully completing an upgrade, Nala will inform you about the progress and prompt you if a restart is needed.



```
dd@dd-vm-ubu2204d-beta: ~
  zenity                    3.42.0-1                3.42.1-0ubuntu1              67 KB
  zenity-common             3.42.0-1                3.42.1-0ubuntu1             276 KB
  zlib1g                    1:1.2.11.dfsg-2ubuntu9  1:1.2.11.dfsg-2ubuntu        58 KB
                                                    9.2
  zstd                      1.4.8+dfsg-3            1.4.8+dfsg-3build1          603 KB

================================================================================
 Summary
================================================================================
 Auto-Remove    3 Packages
 Install        6 Packages
 Upgrade      376 Packages

 Total download size  834.2 MB
 Disk space required  596.9 MB

Do you want to continue? [Y/n]
  Downloading…
  Total Packages: 382/382
  Last Completed: libfreerdp-client2-2_2.6.1+dfsg1-3ubuntu2.2_amd64.deb
  Time Remaining: 0:00:00 ━━━━━━━━━━━━━━━ 100.0% • 834.2/834.2 MB • 5.2 MB/s

  Updating Packages
 Preconfiguring packages
 setting xserver-xorg-legacy/xwrapper/allowed_users from configuration file
 Unpacking:  libc6-dbg:amd64 (2.35-0ubuntu3.1) over (2.35-0ubuntu3)
 Unpacking:  libgomp1:amd64 (12.1.0-2ubuntu1~22.04) over (12-20220319-1ubuntu1)
 Unpacking:  gcc-12-base:amd64 (12.1.0-2ubuntu1~22.04) over (12-20220319-1ubuntu1)
 Setting up: gcc-12-base:amd64 (12.1.0-2ubuntu1~22.04)

  Running dpkg … ━━━━━━━━━━━━━━━━━━━━━━━━━━━ 0.4% • 0:03:07 • 3/771
```

Figure 5: If you do not want Nala to execute the `autoremove` command during an upgrade, you can prevent it in the configuration file.

And that's not all: You can run

```
sudo nala fetch
```

to tell Nala to test all available mirror servers for the performance at your location (Figure 6). In testing, Nala speed-checked 332 servers, resulting in a list of the 16 fastest German and European servers at our site. The first server in the list had a response time of 21ms, while the last clocked in at 28ms. You can enter the desired servers as a space-separated list of numbers and confirm by pressing the Enter key. Nala then writes the desired servers to a new source list named `nala-sources-list` and uses it from then on. It is a good idea to repeat this procedure every few months.

Historical

Have you ever tried to undo an upgrade that went wrong in Debian? For individual packages, this can easily be done by specifying the desired version for the downgrade next to the package name, as in

```
sudo apt install foo=0.98.2
```

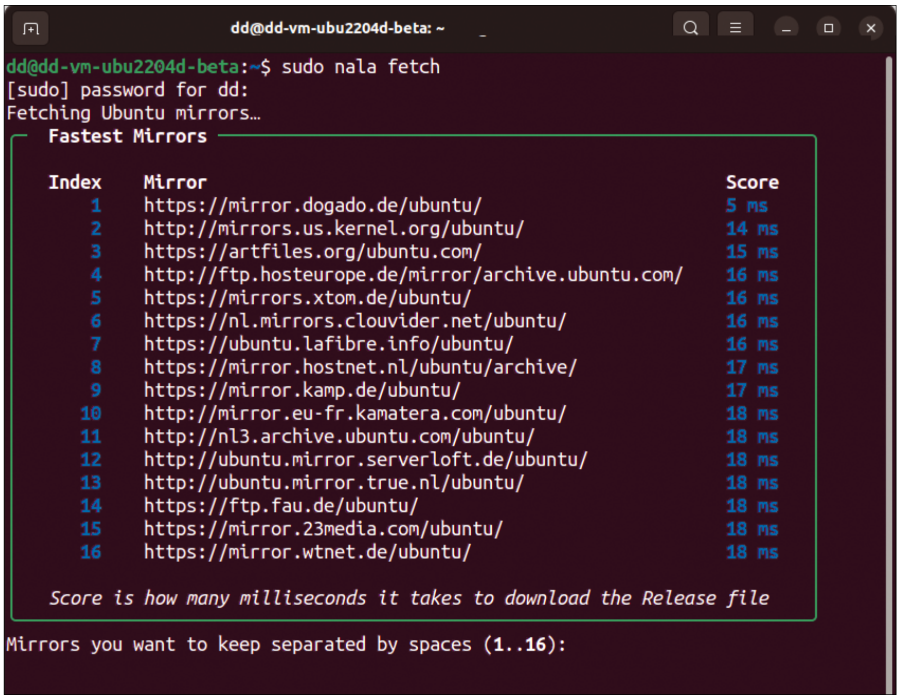
However, if you are updating a large number of packages, you then end up with constructs like the one shown in Listing 1.

Nala offers you a `history` function borrowed from Fedora's DNF package manager. It not only shows you what happened during the update, but it can also roll back the results completely. Currently, the `history` function is still under development and only works for individual packages and their dependencies. The output from

```
sudo nala history
```

shows the most recent actions, whether this be a single package installation, a removed package, or a system upgrade with 300 packages.

Especially for system upgrades where a bug has crept in, the `history` function will be immensely helpful once it is fully implemented. There are good reasons for more and more distributions to use snapshots and atomic upgrades in the form of images. However, because these modern techniques



```
dd@dd-vm-ubu2204d-beta: ~

dd@dd-vm-ubu2204d-beta:~$ sudo nala fetch
[sudo] password for dd:
Fetching Ubuntu mirrors…
 Fastest Mirrors

  Index    Mirror                                                  Score
      1    https://mirror.dogado.de/ubuntu/                        5 ms
      2    http://mirrors.us.kernel.org/ubuntu/                    14 ms
      3    https://artfiles.org/ubuntu.com/                        15 ms
      4    http://ftp.hosteurope.de/mirror/archive.ubuntu.com/     16 ms
      5    https://mirrors.xtom.de/ubuntu/                         16 ms
      6    https://nl.mirrors.clouvider.net/ubuntu/                16 ms
      7    https://ubuntu.lafibre.info/ubuntu/                     16 ms
      8    https://mirror.hostnet.nl/ubuntu/archive/               17 ms
      9    https://mirror.kamp.de/ubuntu/                          17 ms
     10    http://mirror.eu-fr.kamatera.com/ubuntu/                18 ms
     11    http://nl3.archive.ubuntu.com/ubuntu/                   18 ms
     12    http://ubuntu.mirror.serverloft.de/ubuntu/              18 ms
     13    http://ubuntu.mirror.true.nl/ubuntu/                    18 ms
     14    https://ftp.fau.de/ubuntu/                              18 ms
     15    https://mirror.23media.com/ubuntu/                      18 ms
     16    https://mirror.wtnet.de/ubuntu/                         18 ms

  Score is how many milliseconds it takes to download the Release file

Mirrors you want to keep separated by spaces (1..16):
```

Figure 6: **The `nala fetch` command checks all Debian mirrors and selects the fastest ones for your location.**

are still a long way off in Debian, Nala's `history` function offers a passable alternative option that requires hardly any effort.

The line-by-line output of `history` starts on the left with an ID, which plays a crucial role for rolling back (Figure 7). This is followed by the `apt` command, with Nala still partly confusing `update` with `upgrade` as parameters.

But because this is only about information, it does not have any real implications. Once you have found the action to undo, use the ID of the line in

```
sudo nala history undo <ID>
```

After doing so, everything reverts to the way it was before. If you do change your mind,

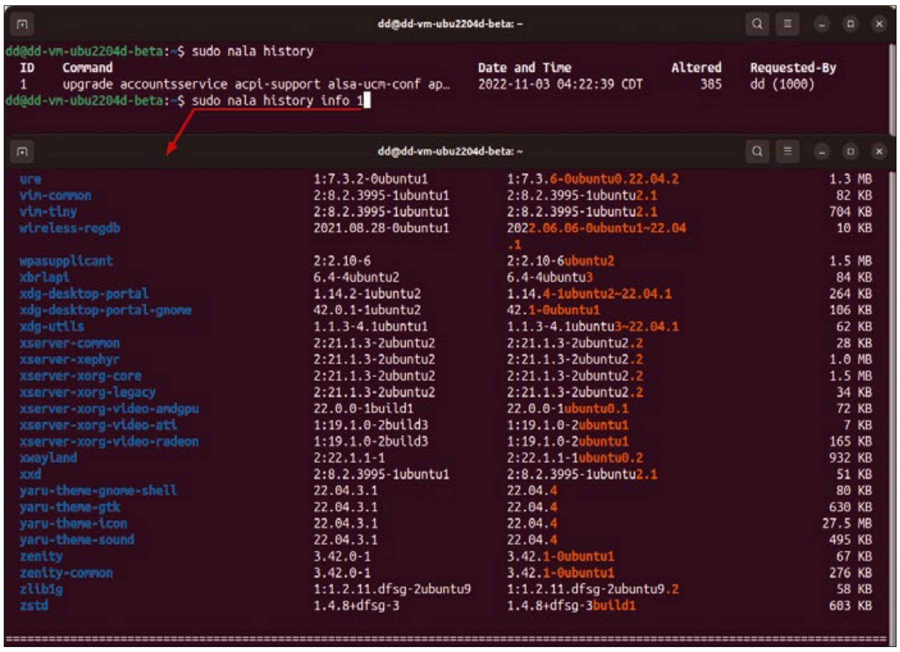
```
sudo nala history redo <ID>
```

will reverse the process and reinstall the upgrade.

Conclusions

Nala adds some important features to Debian's package management: The output is attractively colorful and informative. Clutter is avoided, with the tool highlighting everything that is important. Parallel downloads speed up updates and upgrades, while the `fetch` function additionally lets you choose the fastest mirror servers. But the real highlight is Nala's `history` function, which will beam Debian to a position close to snapshot-supported distributions in the future. The developer intends to port Nala from Python to Rust in the near future to increase download speed. He will then look to integrate the extended `history` function.

I have been using Nala daily for six months without any serious complications. In rare cases, the upgrade hangs for some unknown reason, but continues to run normally after pressing the Enter key. Nala extends `apt` in a way that makes it especially interesting for users of Debian Testing, Unstable, and their offshoots. However, thanks to Nala, updates are also more fun on Ubuntu and other Debian derivatives. ■■■



```
dd@dd-vm-ubu2204d-beta: ~

dd@dd-vm-ubu2204d-beta:~$ sudo nala history
 ID   Command                                       Date and Time              Altered   Requested-By
 1    upgrade accountsservice acpi-support alsa-ucm-conf ap…  2022-11-03 04:22:39 CDT      385   dd (1000)
dd@dd-vm-ubu2204d-beta:~$ sudo nala history info 1
```

```
dd@dd-vm-ubu2204d-beta: ~

 ure                          1:7.3.2-0ubuntu1            1:7.3.6-0ubuntu0.22.04.2        1.3 MB
 vim-common                   2:8.2.3995-1ubuntu1         2:8.2.3995-1ubuntu2.1            82 KB
 vim-tiny                     2:8.2.3995-1ubuntu1         2:8.2.3995-1ubuntu2.1           704 KB
 wireless-regdb               2021.08.28-0ubuntu1         2022.06.06-0ubuntu1~22.04        10 KB
                                                          .1

 wpasupplicant                2:2.10-6                    2:2.10-6ubuntu2                 1.5 MB
 xbrlapi                      6.4-4ubuntu2                6.4-4ubuntu3                     84 KB
 xdg-desktop-portal           1.14.2-1ubuntu2             1.14.4-1ubuntu2~22.04.1         264 KB
 xdg-desktop-portal-gnome     42.0.1-1ubuntu2             42.1-0ubuntu1                   106 KB
 xdg-utils                    1.1.3-4.1ubuntu1            1.1.3-4.1ubuntu3~22.04.1         62 KB
 xserver-common               2:21.1.3-2ubuntu2           2:21.1.3-2ubuntu2.2              28 KB
 xserver-xephyr               2:21.1.3-2ubuntu2           2:21.1.3-2ubuntu2.2             1.0 MB
 xserver-xorg-core            2:21.1.3-2ubuntu2           2:21.1.3-2ubuntu2.2             1.5 MB
 xserver-xorg-legacy          2:21.1.3-2ubuntu2           2:21.1.3-2ubuntu2.2              34 KB
 xserver-xorg-video-amdgpu    22.0.0-1build1              22.0.0-1ubuntu0.1                72 KB
 xserver-xorg-video-ati       1:19.1.0-2build3            1:19.1.0-2ubuntu1                 7 KB
 xserver-xorg-video-radeon    1:19.1.0-2build3            1:19.1.0-2ubuntu1               165 KB
 xwayland                     2:22.1.1-1                  2:22.1.1-1ubuntu0.2             932 KB
 xxd                          2:8.2.3995-1ubuntu1         2:8.2.3995-1ubuntu2.1            51 KB
 yaru-theme-gnome-shell       22.04.3.1                   22.04.4                          80 KB
 yaru-theme-gtk               22.04.3.1                   22.04.4                         630 KB
 yaru-theme-icon              22.04.3.1                   22.04.4                        27.5 MB
 yaru-theme-sound             22.04.3.1                   22.04.4                         495 KB
 zenity                       3.42.0-1                    3.42.1-0ubuntu1                  67 KB
 zenity-common                3.42.0-1                    3.42.1-0ubuntu1                 276 KB
 zlib1g                       1:1.2.11.dfsg-2ubuntu9      1:1.2.11.dfsg-2ubuntu9.2         58 KB
 zstd                         1.4.8+dfsg-3                1.4.8+dfsg-3build1              603 KB

====================================================================================================
```

Figure 7: **Nala's `history` function will support rolling back complete upgrades in the future, which is very helpful should something go wrong during an update.**

Info

[1] dpkg: *https://en.wikipedia.org/wiki/Dpkg*

[2] Apt: *https://en.wikipedia.org/wiki/APT_(software)*

[3] Michael Stapelberg's Debian critique: *https://michael.stapelberg.ch/posts/2019-03-10-debian-winding-down/*

[4] "Improving Linux Package Management" by Ferdinand Thommes, *Linux Magazine*, issue 247, July 2021, *https://www.linux-magazine.com/Issues/2021/247/Distri/(language)/eng-US*

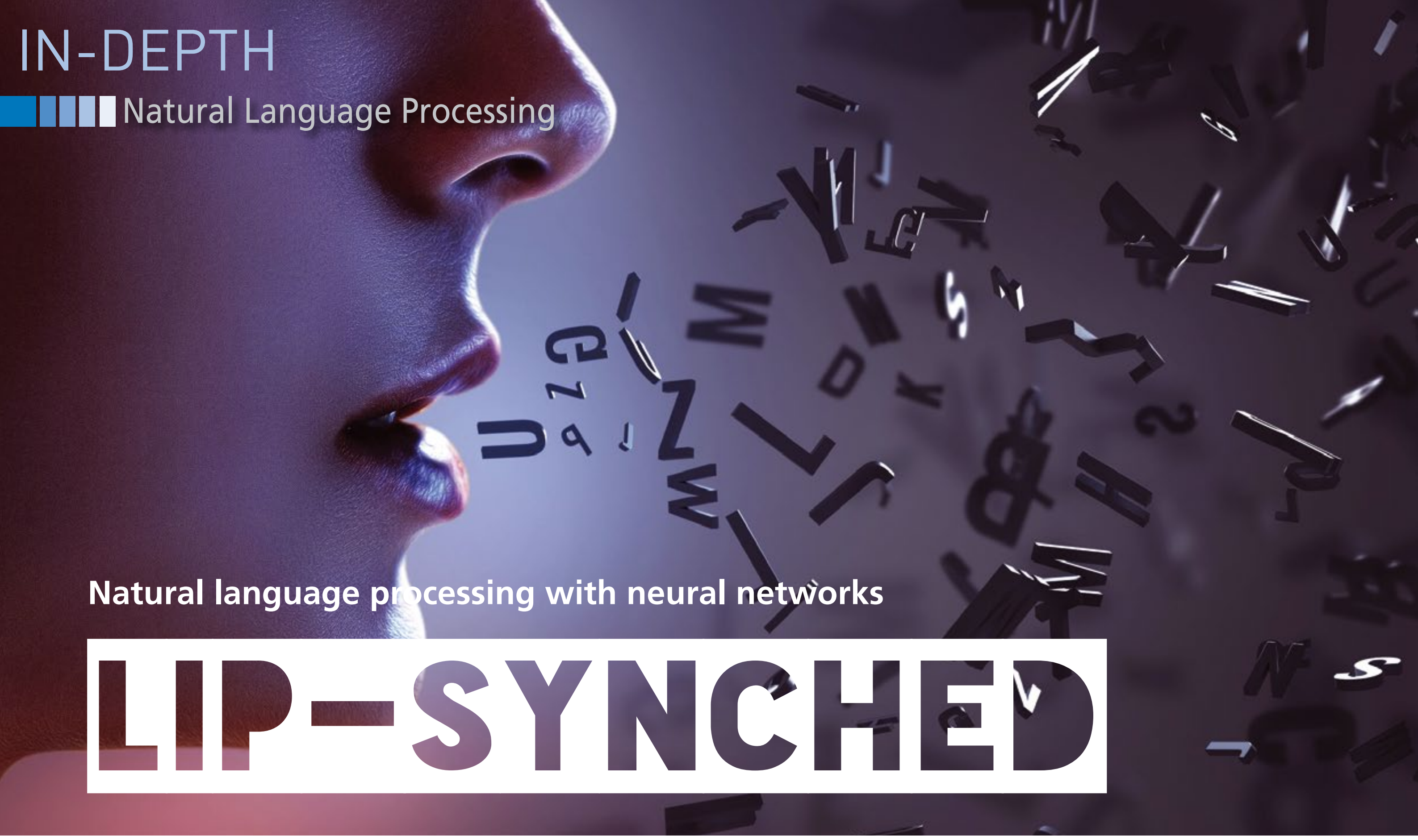
[5] python-apt: *https://apt-team.pages.debian.net/python-apt/library/index.html*

[6] Wiki: *https://gitlab.com/volian/nala/-/wikis/Installation*

[7] Hooks: *https://www.reddit.com/r/linux/comments/v2i0xb/nala_v090_now_in_debian_sid_and_testing/*

[8] Debian bug: *https://bugs.debian.org/cgi-bin/bugreport.cgi?bug=158486*

[9] apt-fast: *https://github.com/ilikenwf/apt-fast#debian-and-derivates*



Natural language processing with neural networks

LIP-SYNCHED

If an actor's lip movements don't match the spoken text in a dubbed movie, it not only stresses people who are hard of hearing, but it can also make things difficult for everyone. AI can help solve this problem with lip-sync translations of movie scripts. *By Timo Baumann and Christian Schuler*

Automated natural language processing is an important field of artificial intelligence (AI) application. Natural language processing tasks range from text searches (such as web searches) to interaction with spoken language (such as with Siri, Alexa, or similar voice-controlled agents). Methods for intelligent language processing that go beyond the simple memorization and patterning of texts have been under development for more than 70 years. The famous Turing test [1] even considers the understanding and production of language to be a central criterion for AI.

Automated speech processing, along with speech recognition, is also one of the earliest applications for big data analytics. As early as 1952, Bell Laboratories' Audrey speech recognition system, for example, was able to set the parameters for each user such that the system recognized sequences of digits with a high degree of accuracy. Initially, this fine-tuning of the parameters had to be done manually. Automating this step takes us to machine learning, the process of adjusting model parameters based on training data.

The results of this machine learning are the models and their parameters. Accordingly, experts do not refer to AI, but instead to AI models. The widespread adoption of neural networks was a breakthrough for machine learning. Although their theoretical foundations were laid early on, more widespread use of neural networks initially required advances in computing power.

The individual neurons of the network each perform a small and relatively simple task. A neuron has no direct information about its neighboring, upstream, or downstream neurons. It acts like a small cog in a pocket watch; it has to be there to keep the watch ticking, but it does not measure time of its own accord.

The task of each neuron is as follows: It first totals the weighted inputs of the neurons upstream of it. The neuron's activation depends on whether this total exceeds a certain threshold. Instead of being proportional, this addition is non-linear, meaning that values far below the threshold result in a *0* and values above result in a *1*. The behavior can be compared to that of classical transistors; it basically allows (sufficiently large) neural networks to compute any possible function.

Unlike transistors, however, activation is not abrupt, but continuous (there are also values between *0* and *1*) and can therefore be differentiated. This allows the error costs that a neural network makes, for example, when classifying an image as "cat image" or "no cat image," to be converted to the individual parameters of the model (the weights of the inputs and thresholds for activating each neuron). To do this, you need to form the (partial) derivatives after each of the parameters, which tells you how changing the respective parameter influences the error.

The parameters are then adjusted slightly toward smaller errors and from there the error costs are recalculated, and so on. Error feedback (backpropagation) and adjusting the parameters toward smaller errors (gradient descent) ensure that the set of neurons can learn to solve a task in a combined way. Neural networks are trained by presenting them with pairs of inputs and expected outputs. You then adjust the model parameters bit by bit until the results have the lowest possible error costs.

Neurons in the neural network are organized in layers, much like visual processing in the brain. Although you can prove that theoretically a single inner layer is sufficient to approximate arbitrary functions, it has been shown that networks with more layers are often easier to train (Figure 1). This is where deep learning comes in. The boundary from shallow to deep is defined by corporate marketing departments and actually has no practical relevance.

Either way, machine learning has thus far not tended toward creating artificial general intelligence (AGI) that behaves intelligently in all areas of the world (despite the claims of marketing departments). Instead, the capabilities in each case depend on the data with which the models were trained. The models are specialists, which is unsurprising: You don't expect a musician to be able to fix a car or a mathematician to make leaded glass windows.

But some models (such as the ones described in this article) are now trained with such a large and varied amount of data, as well as sometimes on different targets, that they may well give a superficial impression of intelligence.

Sequence Learning Problems

Natural language is characterized, among other things, by sentences consisting of different numbers of words, with the order of the words being critically important. This is even more complicated for spoken language. In the sentence "I didn't say we should kill him!," you can put the main stress on different words, and each time it will result in a different meaning. Check it out!

Language develops over time, in reading and writing, and even more so in listening and speaking. It's only natural that people process language word by word, unlocking the meaning bit by bit until they reach the end of a text. Some machine learning techniques have difficulty with sequence learning problems, for example, when faced with sequences of words (i.e., sentences) or letters (i.e., words).

Recurrent neural networks (RNNs) and their variants and derivatives are particularly well suited for sequence learning problems (see Figure 2). RNNs do not have a fixed number of inner neuron layers; instead, this number depends on the length of the input. Each layer processes one element of the input sequence, while also considering the preceding inner layer.

Unlike normal neural networks, the inner layers of an RNN share their parameters. An RNN can learn to grasp the shared meaning of the two words "scoop" and "ice" regardless of exactly

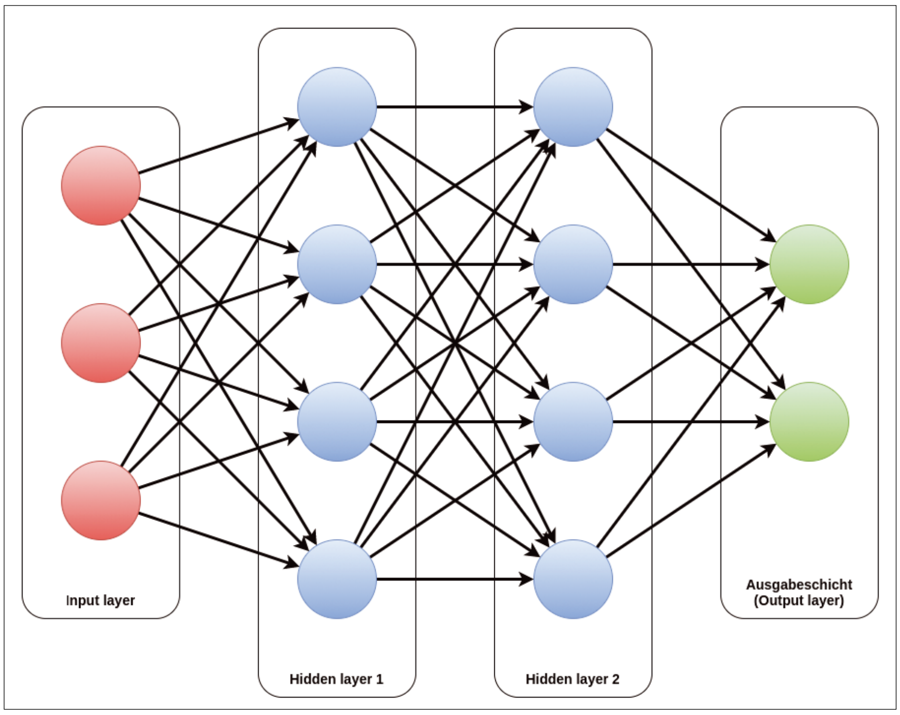


Figure 1: An example of a neural network with two inner layers. The (red) input neurons are activated to reflect the input. All subsequent layers each calculate their consequent activation based on the learned parameters. The result can be seen in the activation of the (green) output neurons.

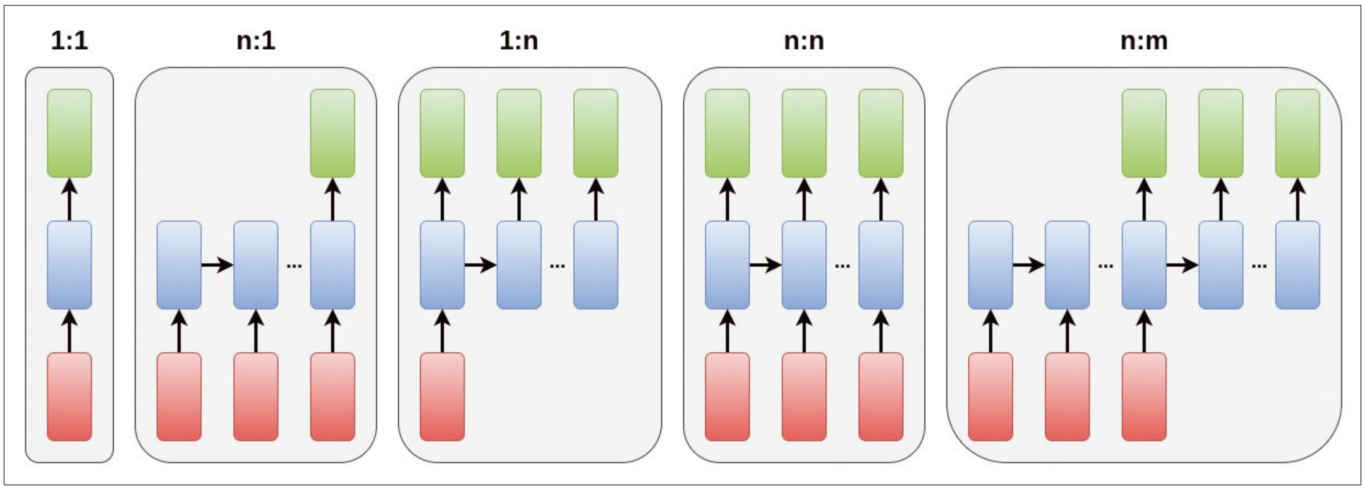


Figure 2: Possible sequence learning tasks that occur during speech processing. (Image modified from Karpathy, 2015 [2].)

where they are juxtaposed in the sentence. In fact, the contexts that the RNN can capture are not limited to adjacent words. It can – at least theoretically – learn to recognize arbitrarily complex and remote relationships.

RNNs can handle multiple variations of sequence learning problems. You can see the simplest case on the far left in Figure 2: a 1:1 relationship between input and output. It does not contain a sequence at all, but it is equivalent to the example shown in Figure 1.

The many-to-one architecture (n:1) is very common. It is used for classification and for instances where only the input is a sequence, among other things. In this way, for example, an input text can be assigned to a category in a given category system. The input here consists of a sequence of words and the output consists of the category to be determined. The process runs iteratively (from left to right in Figure 2) through the inner states (blue). The result (green) is based on the entire sequence of input words (red).

The opposite of the sequence learning problem is producing variable length output from fixed length input. This is desirable, for example, when generating texts, such as when you want to generate a weather forecast from a fixed number of measured values. This is where the one-to-many (1:n) architecture comes into play. After training, you use the model to determine the next word in the output sequence in each case (or to complete the output).

The first many-to-many architecture (n:n) shown in Figure 2 produces an output sequence that is as long as the input sequence. Among other things, this can be used to handle tagging tasks (e.g., tagging company names in texts). Another example would be tagging the departure and arrival train stations, dates and times, and other information that a user could provide when purchasing a train ticket.

Finally, the many-to-many architecture (n:m) on the far right allows inputs and outputs of different lengths, opening up a wide range of new possibilities that the n:n architecture does not offer. The question and the corresponding answer just happen to be the same length in Figure 2.

Texts often cannot be translated into another language with the same content if you are not allowed to change the number of words in the process. This model is known as an encoder/decoder model because it first encodes the input sequence into a representation of fixed length in line with the n:1 principle and then decodes it to form the output sequence. For long sequences, this results in a bottleneck because the model has to squeeze all the information through the fixed representation.

Extensions of the encoder-decoder model use attention control to recompute the attention over the intermediate steps of the input sequence after each step of decoding. This is very helpful, for example, for reliable translation from one language to another, because during encoding the main thing that needs to be captured is the overall meaning of the input sentence. What follows from this is whether, say, the German word *bank* should be translated as "bench" or "bank" in English.

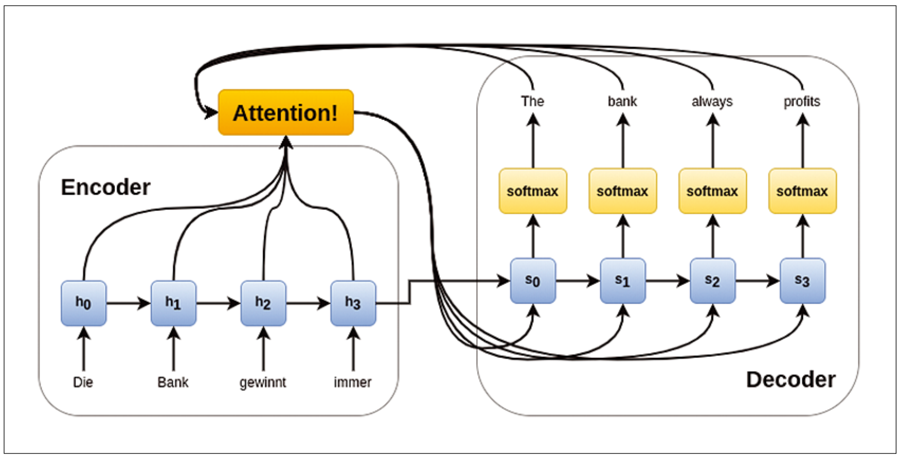


Figure 3: An n:m encoder-decoder model with attention control for end-to-end natural language to natural speech conversion.

In addition, attentional control compensates for differences in sentence structure during decoding.

Listen and See

A special problem arises if you need to adapt a machine translation system with an encoder-decoder architecture to incorporate additional external knowledge into the translation process (Figure 3). This is necessary, for example, to create translations that can be optimally lip-synced. But first, let's explain why this is important.

When people speak, the lips, jaw, and parts of the face move in a way that reflects the sounds that are produced. The upper and lower lips need to touch to produce M and B sounds, while A or O are spoken with the lips open. People who are hard of hearing use this in lipreading to understand the content of what is being said without any (intelligible) sound. Even people with normal hearing always look at the speaker's lips and facial expressions and use that to understand what is being said – to a greater or lesser extent depending on the background noise.

The reverse phenomenon is known as the McGurk effect. If the visually perceived lip movements differ from what you hear, the visual impression will tend to override the auditory impression. "Gaga" is understood to be "Baba" in a movie where the speaker's lips close. Inappropriate lip movements not only lead to false perceptions, the cognitive dissonance between auditory and visual impression stresses the brain.

Movie translations therefore need to take into consideration that viewers will not just hear the speaker, but also see them. The translation needs to include lip movements so the dubbed film is as pleasant as possible to watch in the target language and to avoid artifacts caused by dubbing, such as phantom movements of the lips that spoil the viewing. Audiovisual translation of media is often intended to give the viewer the impression that the speaker in the image did not even have to be translated and dubbed, but that they actually spoke the text you can hear.

This means an additional constraint in the translation process because the visual channel also has to be considered. While translation is usually about

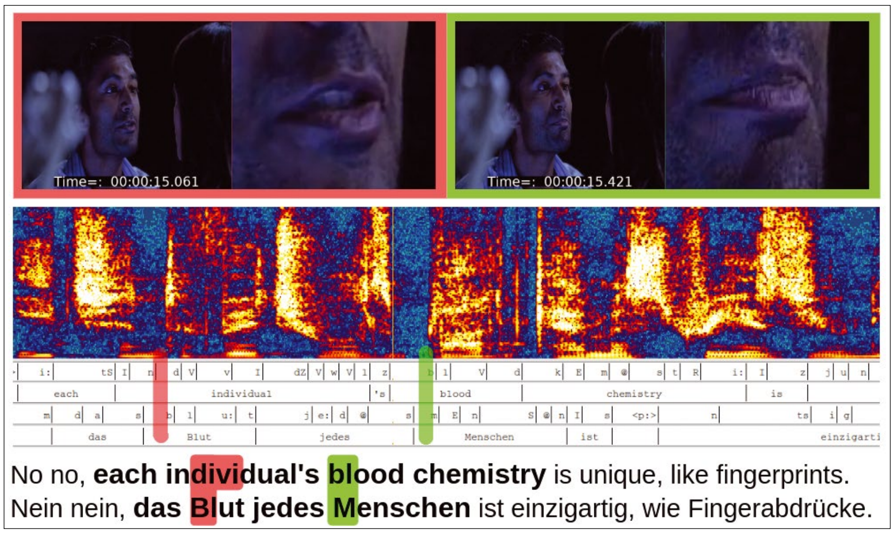

Time=: 00:00:15.061
Time=: 00:00:15.421
each
individual
's
blood
chemistry
is
das
Blut
jedes
Menschen
ist
einzigarti
No no, each individual's blood chemistry is unique, like fingerprints.
Nein nein, das Blut jedes Menschen ist einzigartig, wie Fingerabdrücke.

Figure 4: Maintaining lip synchronization in dubbing using the example of a short scene from *Heroes*, the US TV series.

finding a rendering that converts the content of a source text into the target language as faithfully as possible, audiovisual translation involves balancing content and the visual fit. First and foremost, audiovisual translations need to be of similar length. This can be difficult when words with the same meaning have different lengths in two languages (such as *einzigartig* in German, which translates to "unique" in English). In addition, you have to pay attention to the lip and jaw positions, at least as long as the speaker is in picture.

Figure 4 demonstrates this with an example from the television series *Heroes*. In the original English language version, the actor says "No, no, each individual's blood chemistry is unique, like fingerprints." It goes without saying that this word sequence precisely matches the visible movements. While a translation into German that is as faithful to the source as possible might now be a choice between *individuellen Blutchemie* (individual blood chemistry) or *Chemie des Bluts jedes Menschen* (chemistry of the blood of every human being), the chemistry bit was actually completely dropped in the lip-synced version. What German viewers hear at this point is *"Nein nein, das Blut jedes Menschen ist einzigartig, wie Fingerabdrücke"* ("No, no, each person's blood is unique, like fingerprints"), which matches the timing very well.

However, it's not perfect. At the point marked in green, the "M" is replaced by "Bl" in lip-sync. This is desirable because the two sounds can only be produced by the lips meeting. The situation is completely different with the area marked in red. While "divi" in the original version requires a perceptible opening of the lips, the "bl" which German viewers now hear can only be produced by closing the lips.

Minor deviations that occur rarely are not noticed by most viewers. In addition, prolonged and repeated viewing of

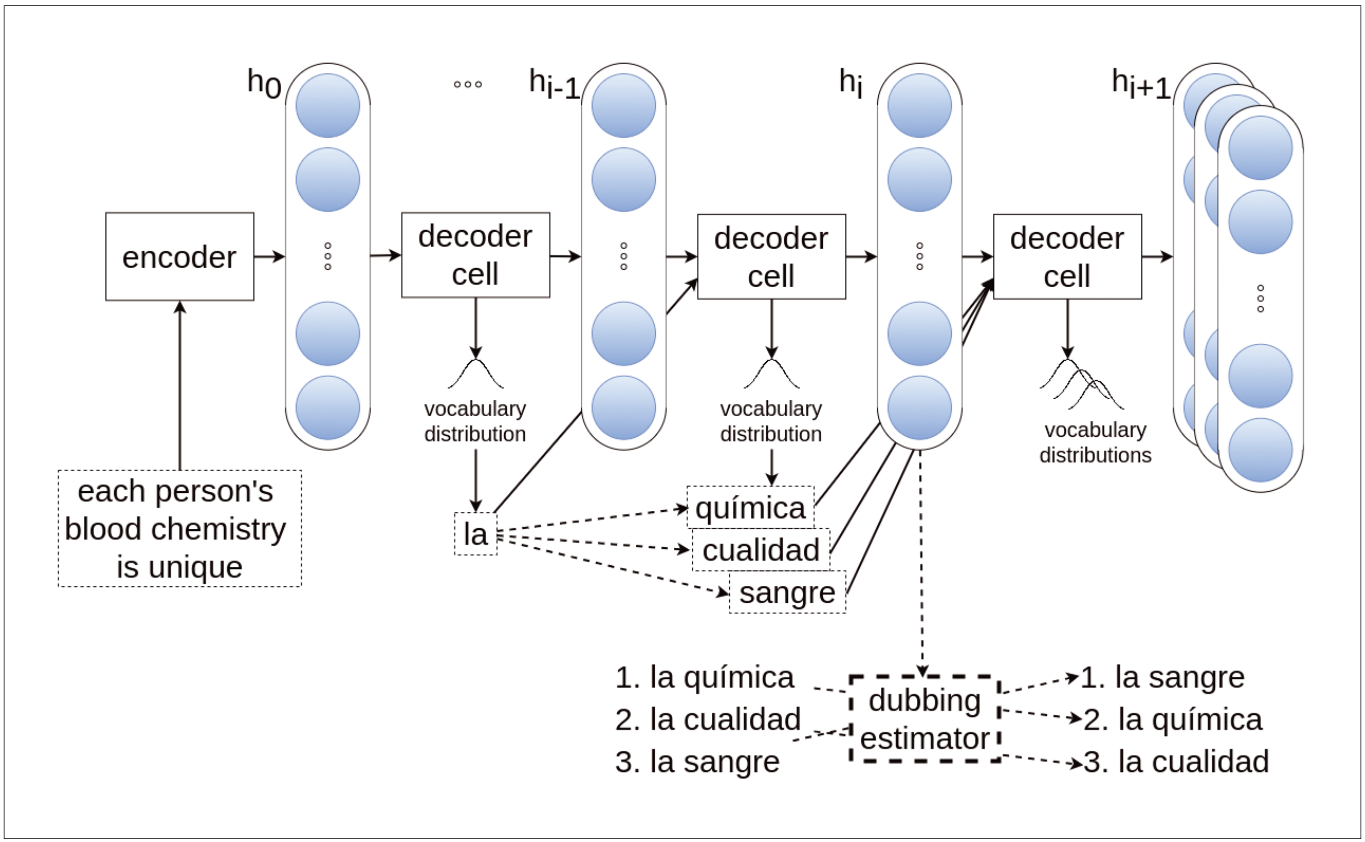


Figure 5: The encoder-decoder architecture extended by a dubbing estimator improves the synchronous speech of the translation.

dubbed material can cause viewers to become so familiar with it that they (often unconsciously) overlook this minor annoyance. However, if major lip-sync glitches reoccur in a movie or become too extreme (for example, if a character's voice is heard for seconds while their lips remain motionless), this drastically reduces the viewer's perception of the translation quality.

Lip-Synced Computer Translation

Extending the encoder-decoder model discussed earlier at a crucial point helps the model to generate translations that can subsequently be lip-synced. The lower part of Figure 5 shows the "dubbing estimator" that we developed. During automatic translation, the dubbing estimator reaches a compromise between a translation that is as faithful to the source as possible and a lip-synced translation that fits the movements in the best possible way.

To do this, the dubbing estimator does two key things. It looks at the speech duration and syllable count in the source language and matches this with the estimated duration and syllable count in the target language. The dubbing estimator also balances the results of a video analysis of facial movements (and whether or not the speaker is actually visible) with expected movements when the translation is spoken. It is useful that the facial patterns (called visemes, which are speech sounds that look alike) are less strict than the sounds: P, B, and M look practically the same in the video.

The encoder in this system remains unchanged and is not shown separately in Figure 5 for this reason. However, during decoding, it is not only the word most likely to occur next in the translation that is created, but several words, and based on these, once again, a number of the most probable next words. The model sorts the resulting word sequences by their estimated translation quality: the best first and the worst last. The dubbing estimator now calculates the lip-sync quality for each candidate and weighs these two values together. Translations that are a poor match end up at the back of the list in order to achieve the best possible compromise between translation and dubbing quality.

We also ran our experiments with English-Spanish data, which is why Figure 5 shows Spanish words. You can see that *la quìmica* and *la cualidad* are initially the better translations, but in combination with the lip-sync quality they can't compete with *la sangre*. This results in the translation *La sangre da cada indivìduo es ùnico, como una huella*, which is a pretty good lip-sync candidate. In Spanish, it makes sense to shorten "blood chemistry" to create enough time for *de cada indivìduo es ùnico*.

Outlook

The example shown here is the subject of current research in the field of language technology. Scientists are currently still investigating questions such as: Which weighing of the individual factors, such as translation quality and synchronization quality, optimizes the overall quality? At which point exactly do people even notice that the lip-sync is no longer perfect?

It is also important to remember that world languages vary considerably. Consequently, this problem is so complex that a single model cannot do justice to it. As a result, it is essential to train separate models for the different pairs of languages to be translated. ■■■

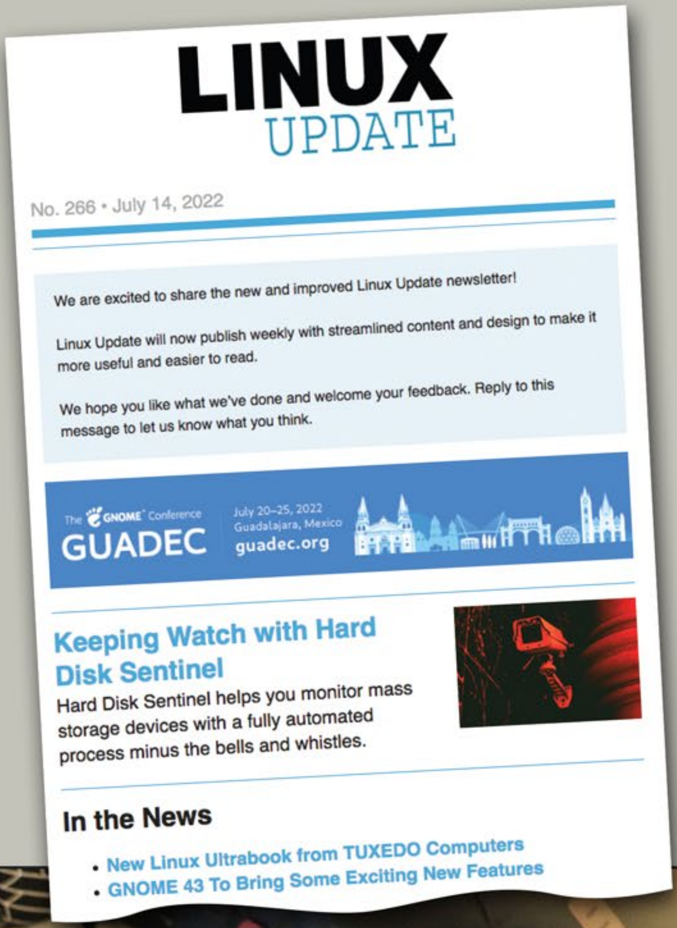
Info

[1] "Computing Machinery and Intelligence" by A.M. Turing, *Mind*, volume LIX, October 1950, pp. 443-460, *https://academic.oup.com/mind/article/LIX/236/433/986238*

[2] Karpathy (2015); *http://karpathy.github.io/2015/05/21/rnn-effectiveness/*

Author

Timo Baumann is Professor of Artificial Intelligence at the Faculty of Computer Science and Mathematics at OTH Regensburg where he focuses on natural language processing. His research focuses on socially and interactively adequate conversational agents, prosody processing, and automatic lip-synchronous dubbing. **Christian Schuler** is currently studying computer science in the Master of Science program at the University of Hamburg. His bachelor thesis was about the acquisition of perceived quality of lip synchronicity in audiovisually translated material.



LINUX UPDATE

No. 266 • July 14, 2022

We are excited to share the new and improved Linux Update newsletter!

Linux Update will now publish weekly with streamlined content and design to make it more useful and easier to read.

We hope you like what we've done and welcome your feedback. Reply to this message to let us know what you think.

The GNOME Conference GUADEC July 20–25, 2022 Guadalajara, Mexico guadec.org

Keeping Watch with Hard Disk Sentinel

Hard Disk Sentinel helps you monitor mass storage devices with a fully automated process minus the bells and whistles.

In the News

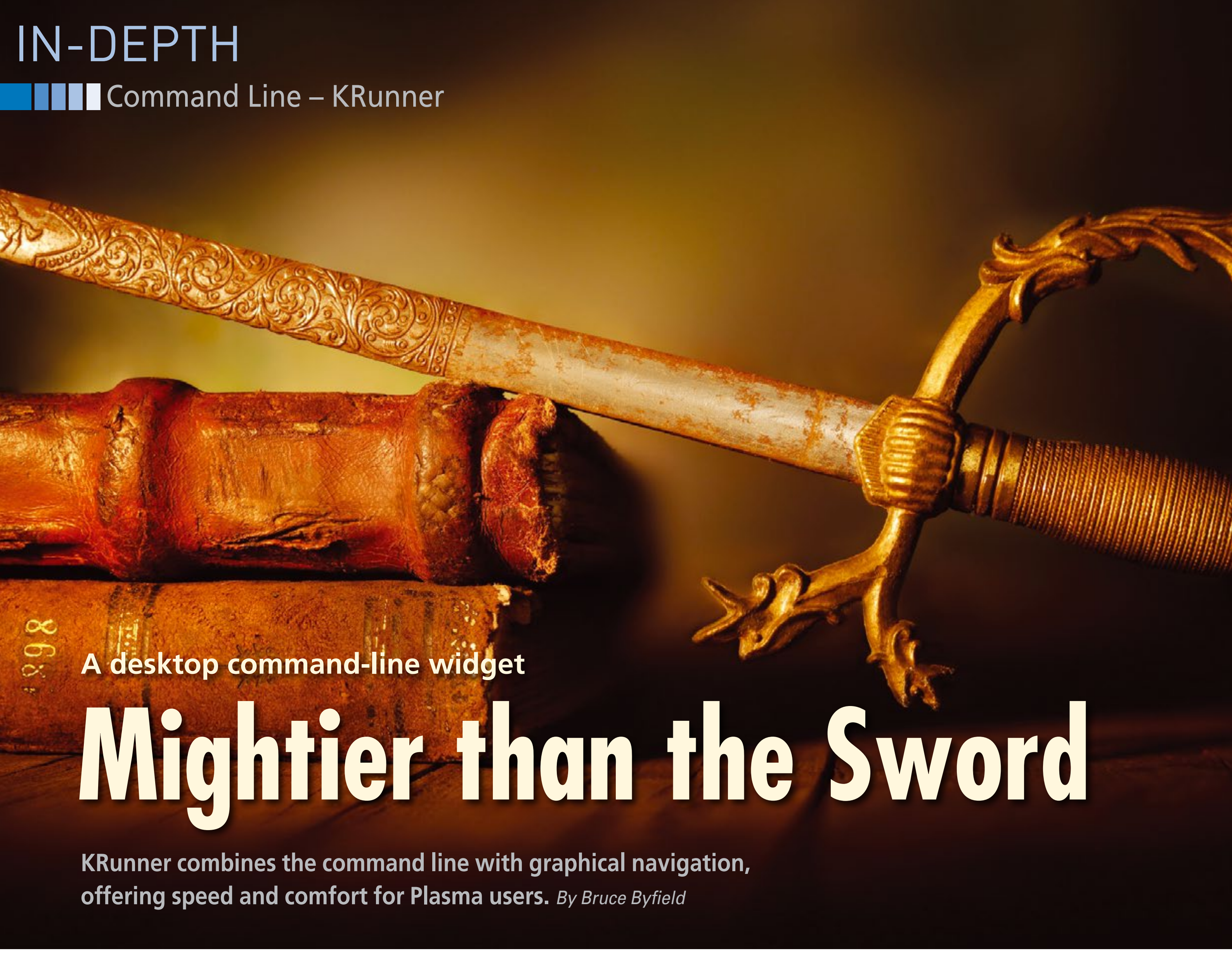
- New Linux Ultrabook from TUXEDO Computers
- GNOME 43 To Bring Some Exciting New Features

Need more Linux?

Our free Linux Update newsletter delivers insightful articles and tech tips to your mailbox every week. Subscribe free, and you'll discover:

- **Original articles on real-world Linux**
- **Linux and open source news**
- **Tips on Bash scripting and other advanced techniques**
- **Open source jobs and events**
- **and more!**

bit.ly/Linux-Update



A desktop command-line widget

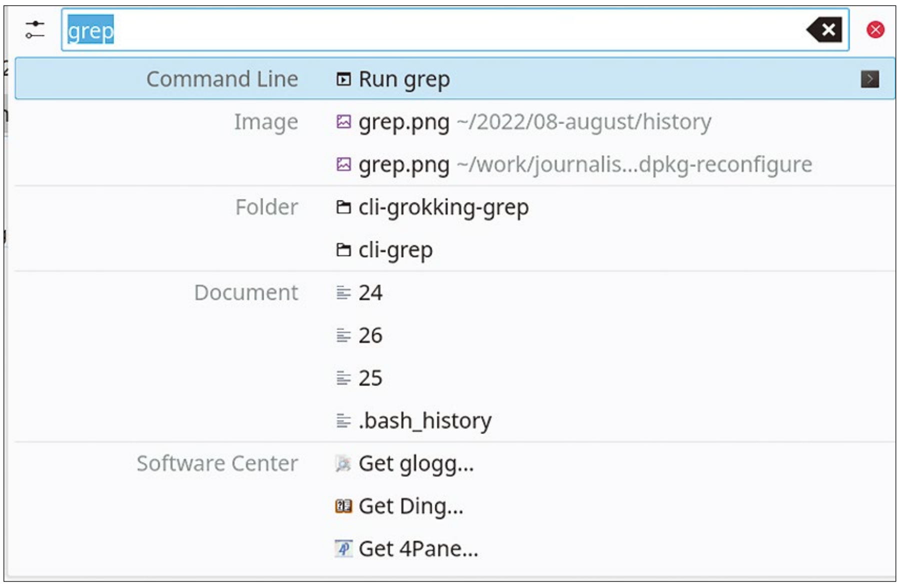
Mightier than the Sword

KRunner combines the command line with graphical navigation, offering speed and comfort for Plasma users. *By Bruce Byfield*

For most users, the command line and the desktop environment are distinct. One notable exception is KRunner [1], which is installed by default with KDE's Plasma. Interacting with other default Plasma applications, as well as the system hardware, KRunner is basically a convenient widget for entering a single command, but it is also an application launcher and general navigation tool, as well as a calculator, a measurement and currency converter, music player controller, and even a spell checker – all controllable from the keyboard. In fact, should you choose, you can control the desktop entirely from KRunner, making it one of the most versatile applications in all of Plasma, despite suffering from few limitations.

The KRunner Interface

KRunner uses graphical navigation in the service of the command line (Figure 1). You can run it from the menu, but starting it with one of the KRunner



```
grep
Command Line      Run grep
Image             grep.png ~/2022/08-august/history
                  grep.png ~/work/journalis...dpkg-reconfigure
Folder            cli-grokking-grep
                  cli-grep
Document          24
                  26
                  25
                  .bash_history
Software Center   Get glogg...
                  Get Ding...
                  Get 4Pane...
```

Figure 1: KRunner is a versatile command-line widget for interacting with the desktop environment.

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/*.

keyboard shortcuts (Alt + F2 or Alt + Space) is more efficient. The KRunner window has an entry field at the top, with a list of possible text completions beneath it. You select these items with the arrow keys or mouse. To the left of the entry field is a button that opens the KRunner System Settings (Figure 1). From the settings, you can choose whether KRunner opens at the top center or top middle of the screen, and whether its text completions are determined partly from previous choices. Below is a list of 26 plugins, some of which can be configured to change the order in which results are displayed, or similar features. If you are short on memory or have no use for a plugin, you can disable it (Figure 2).

Entering Commands

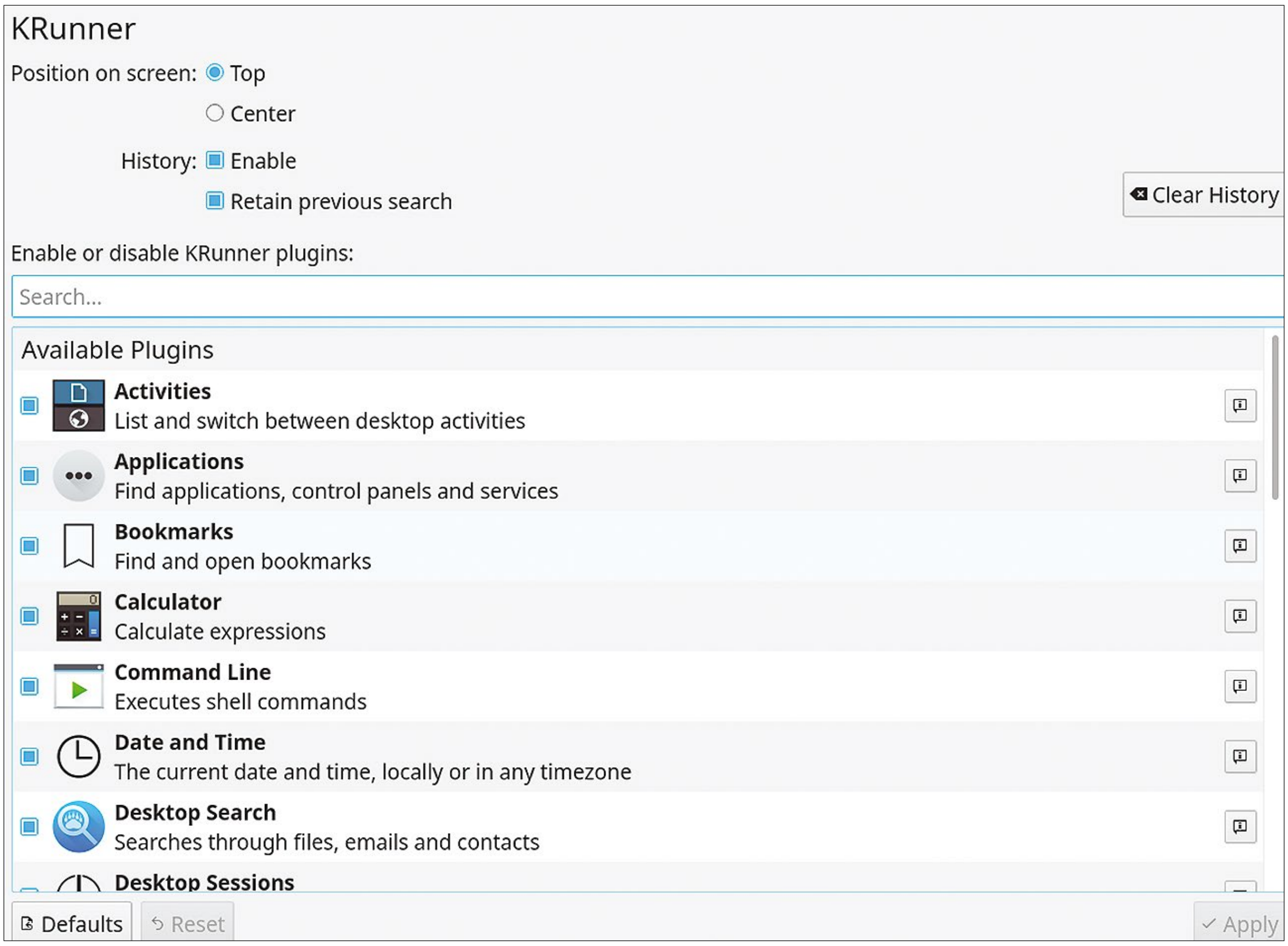
Most commands that can be entered in a terminal can be entered in KRunner, using the same syntax as in the Bash shell. At the top of the drop-down list, you can select the option to run the command. In the same way, you can install software from KRunner. Type in your best guess at an application's name, and KRunner will offer suggestions. Click a suggestion, and the KDE Plasma Software Center opens.

KRunner also has its own handful of commands to call different functions. Should you want to repeat a command, select it from the drop-down list at the end of the entry field. One of the most useful KRunner commands is `kill`, which shuts down any application, including a misbehaving one. Note, though, that KRunner's `kill` command is not the same as Bash's. In fact, it is an arbitrary choice of characters that can be changed in *System Settings | KRunner | Terminate Applications*. You can also choose whether the list of applications that can be killed displays with the least or most CPU useage first or according to typing completion. Similarly, to view a man page, the syntax is `Man:/COMMAND`. The man page opens in the KDE man page viewer in Konqueror. Typing `#COMMAND` no longer works, although it did in past versions of KRunner.

Opening Resources

One of KRunner's main uses is as an alternative to scanning menus and file managers. Files and directories can be opened by entering their paths. For instance, `file:/home/HOME` opens the home directory in Dolphin; a file path opens the file in the appropriate application. Conveniently, if you don't know the path, you can enter just the file name and choose it from the list of possible results.

Web page addresses and bookmarks in your default browser can be opened by typing them in KRunner's entry field. To save time, you can use the Web Search Keywords in System Settings. For example, typing `backports` opens Debian Backports, and typing `arch` opens the ArchWiki for Arch Linux. If a web page is not among the installed defaults, you can add a keyword.


KRunner

Position on screen: [✓ Top] Center

History: [✓ Enable] [✓ Retain previous search]

Enable or disable KRunner plugins:

Available Plugins

- [✓] **Activities** List and switch between desktop activities
- [✓] **Applications** Find applications, control panels and services
- [✓] **Bookmarks** Find and open bookmarks
- [✓] **Calculator** Calculate expressions
- [✓] **Command Line** Executes shell commands
- [✓] **Date and Time** The current date and time, locally or in any timezone
- [✓] **Desktop Search** Searches through files, emails and contacts
- **Desktop Sessions**

Figure 2: KRunner is modular, so you can disable plugins that you don't need for faster performance.

Similarly, you can open a virtual workspace with `Desktop NUMBER`, but if you customize the name, the best KRunner can do is open the Virtual Workspace window in System Settings. Bookmarks in Dolphin can also be opened, including external drives or a contact in KAddressbook in your email reader. The KDE UserBase Wiki suggests that Activities can be started by typing their names, but in practice that feature does not work in recent Plasma releases.

Calculations and Conversions

KRunner includes a calculator that not only does basic arithmetic, but also advanced mathematical functions [2], including algebra, calculus, and geometry. In addition, typing a figure followed by a standard currency abbreviation (i.e., $ or €) or a three letter abbreviation (i.e., CAD for Canadian Dollars or GBP for Great British Pound) will show its current exchange rate with other currencies based on figures from the European Central Bank. In the same way, you can enter measurements with standard abbreviations such as lb for pound and k for kilometer to convert them into other units – a function especially useful for those who switch regularly between metric and imperial systems.

Finding Date and Time

Entering `date` or `time` displays current information. By default, the results are for your system's time zone, but another time zone can be added at the end of the command. For some reason, the results are in a formal sentence, and there is no way to change the format, which somewhat lessens the usefulness.

Spelling and Special Characters

The command `spell STRING` suggests possible alternative words, while `#HEXADECIMAL UNICODE` shows a special character. In both cases, selecting a result copies it to the clipboard so you can paste it into a document.

Controlling Music Players

When a music player designed for KDE is open, like Amarok or Clementine, you can use KRunner to control it. Entering the name of an artist, album, or song will add the result to the end of the current search, while `play SONG search` will play the result immediately. When a song is selected, you can also use the commands `play`, `pause`, `skip`, or `next`.

Computer Management

From KRunner, you can go directly to a section of System Settings. As well, KRunner can be used to enter the usual shutdown controls: `sleep`, `lock`, `restart`, and `shutdown`. In addition, `screen brightness PERCENTAGE` will adjust the monitor screen.

Limitations

KRunner is not a complete replacement for all aspects of Plasma. It is meant to perform a single operation and then close. It is not suitable, for instance, for compiling binary code, because you would want to see all the messages while building. The only solution I have found is to create a new Activity and leave it unpopulated so that typing with the Activity open automatically opens KRunner. Just as annoying, the documentation appears incomplete and is occasionally outdated, perhaps because of a lack of communication between the KRunner maintainers and other applications. Still another weakness is that KRunner is designed to work with standard Plasma applications and will not work as intended if you prefer other applications.

Yet despite these limitations, if you take the time to learn the parts of KRunner that fit with your workflow and work with your preferred apps, KRunner is often faster than the desktop. It is also more ergonomic and can save you hundreds of mouse clicks per day – proof once again that the command line can sometimes be mightier than a graphical interface. ∎

Info

[1] KRunner: *https://docs.kde.org/trunk5/en/plasma-desktop/plasma-desktop/krunner.html*

[2] Supported calculator functions: *http://qalculate.github.io/manual/qalculate-definitions-functions.html*

A Go password manager for the terminal

Hide and Seek

A Go application for the terminal helps Mike Schilli remember his passwords. *By Mike Schilli*

Whether it's on a Post-it note under the screen or in a commercial application like OnePass, users have to write down their passwords somewhere. The Go application for the terminal presented in this article encrypts the sensitive data for storage on the hard drive and displays selected entries after entering the master password. The secret data leaves traces only in the computer's memory, with the traces vanishing when the program is closed.

Resourceful users could simply put all account names and passwords in a text file and encrypt it. However, to add new entries, the file would have to be decrypted and then re-encrypted after editing. To ensure that no unencrypted data remains on the drive, you would then need to run a scrub command to overwrite the deleted file after every edit. After decryption, all of the passwords would come up at the same time, emblazoned on the screen, where a passing colleague with sharp eyesight might catch a glimpse of one or more of them.

A number of password apps manage passwords in an exemplary manner, but do you really want to trust a random company with your confidential data and rely on them not to make mistakes during encryption or data management? Let's not forget that apps like OnePass hit you with significant monthly fees, and a guy like me has to count my pennies. The passview program (`pv`) presented in this article manages an encrypted collection of passwords and displays a selected entry in a terminal user interface (UI) after entering the master password (Figure 1). You can scroll through the entries and pick out the one you want before its sensitive data actually appears at the push of a button.

When you press Enter, `pv` removes the asterisks for a selected entry, revealing the secret account and password details. If you move up or down the list with the cursor keys *K* and *J* (like in Vim of Vi), `pv` again masks the released entry with asterisks. Pressing *Q* quits the program and clears the terminal window. No sensitive data remains; on the hard disk you are just left with the encrypted password file.

Portable

As a Go binary, the program already contains everything it needs for running on similar architectures. All you have to do is copy the binary and the encrypted password file to systems where you want access to the passwords. To add new entries, just invoke the program with the `--add` option. `pv` then asks you for the master password. If you enter it correctly, you are allowed to append a new line to the encrypted file at the `New entry:` prompt (Listing 1).

The first word of the newly added entry is the service associated with the password (`gmail` in this example). Its name also appears in the masked version of the line in the UI (like in the first line of Figure 1). The name acts as a navigation aid to help you find, select, and display the desired entry. The rest of the newly inserted line contains the username and password. You can freely choose the format and, for example, save only mnemonics instead of the complete data.

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.

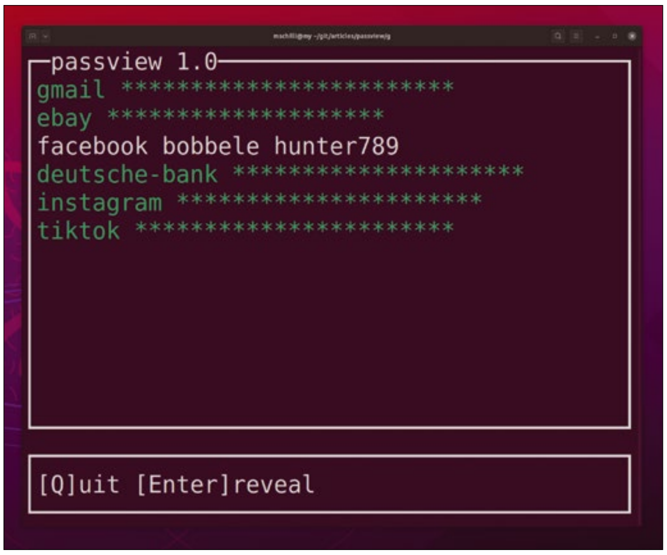
After entering the new data (or if you call `pv` without options) the terminal UI appears with the scrollable listbox revealing selected entries if desired.

Crypto Genius

The password safe uses symmetric encryption. That means that it uses the same master password to encrypt and decrypt the file. The Age [1] project on GitHub provides a ready-made Go library written by a Google engineer for encrypting and decrypting data. It mainly uses public key methods, but symmetric encryption is also on the list. By the way, according to the project page, Age is pronounced like the Italian word "aghe" (needles).

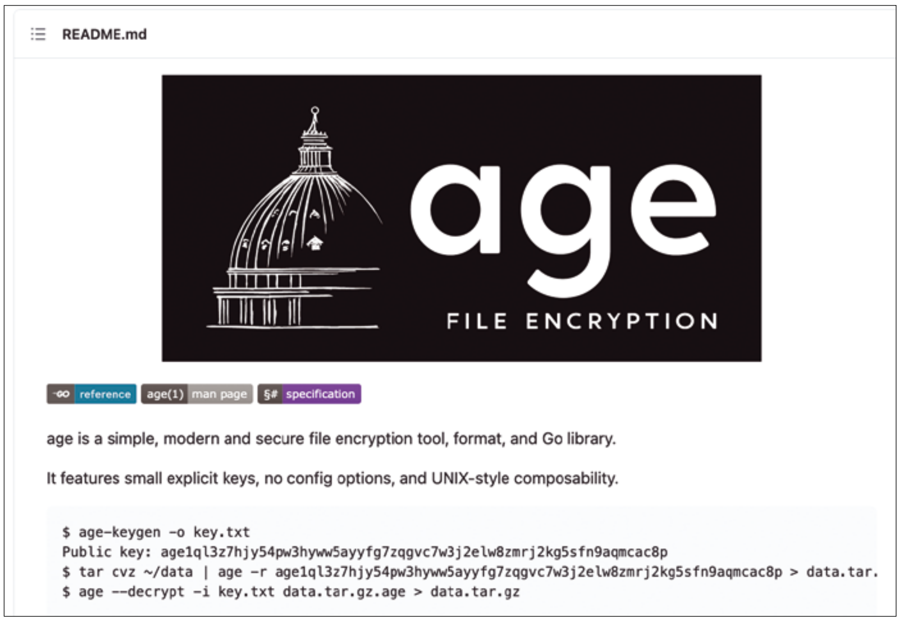
Listing 1: New Password

```
$ pv --add
Password: ***
New entry: gmail bodo@gmail.com hunter123
```



```
passview 1.0
gmail ************************
ebay ********************
facebook bobbele hunter789
deutsche-bank *********************
instagram **********************
tiktok ***********************

[Q]uit [Enter]reveal
```

Figure 1: The `pv` program in action.



age is a simple, modern and secure file encryption tool, format, and Go library.

It features small explicit keys, no config options, and UNIX-style composability.

```
$ age-keygen -o key.txt
Public key: age1ql3z7hjy54pw3hyww5ayyfg7zqgvc7w3j2elw8zmrj2kg5sfn9aqmcac8p
$ tar cvz ~/data | age -r age1ql3z7hjy54pw3hyww5ayyfg7zqgvc7w3j2elw8zmrj2kg5sfn9aqmcac8p > data.tar.
$ age --decrypt -i key.txt data.tar.gz.age > data.tar.gz
```

Figure 2: The Age encryption project.

Symmetrically Encrypted

Symmetric encryption is the most practical solution for a file that is only accessed by one user at any given time. If multiple people share access, public-private key pairs (also using methods from the Age library) could be used to implement a solution that gives different users the ability to access a shared file with their own passwords.

Listing 2 shows the `writeEnc()` and `readEnc()` functions used by the main program to later encrypt and decrypt the plaintext data of the password file. Line 9 uses `test.age` to define the name of the encrypted password file on disk. The `.age` extension indicates that it was encrypted by the Age library.

The Age library uses an object of the `Recipient` type for writing (i.e., encrypting). In other words, the recipient is the entity to which encrypted data is sent. The call to the `NewScryptRecipient()` function in line 11 takes the password as the only parameter, and `Scrypt` points to the symmetric `crypt` function that Age implements. Line 15 opens the encrypted password file for writing, using `O_CREATE` to create the file if it does not already exist.

By the way, the same file opening option exists in the C programming language on Unix, where it is missing the last "E" because it's called `O_CREAT` there. Ken Thompson, one of the Unix founding fathers, was once asked what he would do better if he had to design Unix again, and he promptly said: "I'd spell 'creat' with an 'e'" [2]. Go now obviously granted him this wish.

Armoring for Transfer

The `O_TRUNC` option rewinds the write location in an existing password file to the beginning so that subsequent print commands simply overwrite it. Now, the binary data in the encrypted file might confuse editors, or a network transfer program might be tempted to restructure binary sequences while transferring the content over to a remote location. This is why line 20 sets up an `armor` type writer to re-encode the binary data, which will then still appear scrambled, but at least as lines of manageable length and without non-printable characters (Figure 3).

The functions used in Listing 2 are a good illustration of the writer mechanism so often used in Go. A writer always takes data from somewhere and then writes it somewhere else. For example, `OpenFile()` in line 15 opens a file and returns a writer object named `out`. The armor mechanism from the *armor* package takes this writer object and returns its own writer object, `armorWriter`. This object, in turn, is taken by the `Encrypt()` function in line 22, which then returns another writer `w`, writing encrypted data, to which line 27 then starts writing to using `io.WriteString()`.

In other words, the code implements a contraption of nested functions reminiscent of a Unix pipe. The first stage takes the clear-text data, and the armored and encrypted data drops out at the end. Thanks to the writer interface supported by Go, the functions in the chain don't have to worry about the type of data they are transporting: As long as every link in the chain supports the writer interface, everything runs like clockwork.

In our case, the Age functions even use the `WriteCloser` interface, which supports both the `Write()` and `Close()` calls. The `Close()` calls are extremely important for buffered output. If the `Close()` call is left out, any memory caches used in the pipe may not be cleared at the end, and a truncated and therefore unreadable mess of data will quickly accumulate at the end of the pipe.

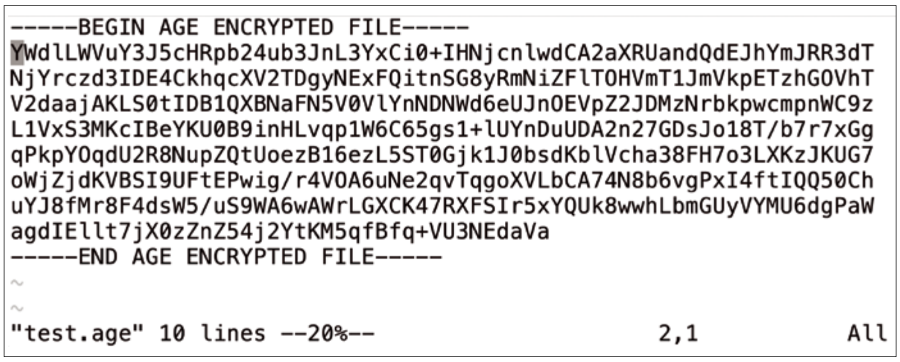
Encrypted Read

Conversely, `readEnc()` reads data from the encrypted password file starting in line 32 and returns the clear-text data as a string. To do this, the function takes the master password for symmetric encryption as a string and upgrades it to an `Identity` object, for the call to the `Decrypt()` function later on in line 44.

But again, the reader must first get through the armor of the encrypted data. This is done by the `armor` type `Reader` in line 43 which in turn receives another reader object for the open password file as a parameter. To read the data from the armor breaker's reader, decrypt it, and store it in a string for return, line 48 uses `io.Copy()` to vacuum up all the reader data and drop the data into the waiting `out bytes` buffer. The buffer's `String()` method turns the byte array into a string, while line 51 returns the clear-text data to the caller.

I don't want the entries in the password viewer's listbox to come up immediately in full glory when the UI appears later. Rather, I only want to be able to read the first word of each line, while asterisks obfuscate everything else. To do this, the `mask()` utility function in Listing 3 takes a string, iterates over its characters, and replaces them with an asterisk if the `tomask` flag is set. Initially, this is not the case until line 11 detects a space in the string. The algorithm then thinks it has reached the position to the right of the first word. When it gets there, it sets `tomask` to `true` and hides the rest of the string under asterisks.

The `main()` function in Listing 4 checks for the optional `--add` command-line argument using the *flag* package. If the flag is set, the `if` block in line 25


```
-----BEGIN AGE ENCRYPTED FILE-----
YWdlLWVuY3J5cHRpb24ub3JnL3YxCi0+IHNjcnlwdCA2aXRUandQdEJhYmJRR3dT
NjYrczd3IDE4CkhqcXV2TDgyNExFQitnSG8yRmNiZFlTOHVmT1JmVkpETzhGOVhT
V2daajAKLS0tIDB1QXBNaFN5V0VlYnNDNWd6eUJnOEVpZ2JDMzNrbkpwcmpnWC9z
L1VxS3MKcIBeYKU0B9inHLvqp1W6C65gs1+lUYnDuUDA2n27GDsJo18T/b7r7xGg
qPkpYOqdU2R8NupZQtUoezB16ezL5ST0Gjk1J0bsdKblVcha38FH7o3LXKzJKUG7
oWjZjdKVBSI9UFtEPwig/r4VOA6uNe2qvTqgoXVLbCA74N8b6vgPxI4ftIQQ50Ch
uYJ8fMr8F4dsW5/uS9WA6wAWrLGXCK47RXFSIr5xYQUk8wwhLbmGUyVYMU6dgPaW
agdIEllt7jX0zZnZ54j2YtKM5qfBfq+VU3NEdaVa
-----END AGE ENCRYPTED FILE-----
~
~
"test.age" 10 lines --20%--                         2,1            All
```
Figure 3: **The encrypted password file on the hard disk.**

Listing 2: **crypto.go**

```
01 package main
02 import (
03   "bytes"
04   "filippo.io/age"
05   "filippo.io/age"
06   "io"
07   "os"
08 )
09 const secFile string = "test.age"
10 func writeEnc(txt string, pass string) error {
11   recipient, err := age.NewScryptRecipient(pass)
12   if err != nil {
13     return 0, err
14   }
15   out, err := os.OpenFile(secFile, os.O_RDWR|os.O_
   CREATE|os.O_TRUNC, 0600)
16   if err != nil {
17     return 0, err
18   }
19   defer out.Close()
20   armorWriter := armor.NewWriter(out)
21   defer armorWriter.Close()
22   w, err := age.Encrypt(armorWriter, recipient)
23   if err != nil {
24     return 0, err
25   }
26   defer w.Close()
27   if _, err := io.WriteString(w, txt); err != nil {
28     return 0, err
29   }
30   return nil
31 }
32 func readEnc(pass string) (string, error) {
33   identity, err := age.NewScryptIdentity(pass)
34   if err != nil {
35     return "", err
36   }
37   out := &bytes.Buffer{}
38   in, err := os.Open(secFile)
39   if err != nil {
40     return "", err
41   }
42   defer in.Close()
43   armorReader := armor.NewReader(in)
44   r, err := age.Decrypt(armorReader, identity)
45   if err != nil {
46     return "", err
47   }
48   if _, err := io.Copy(out, r); err != nil {
49     return "", err
50   }
51   return out.String(), nil
52 }
```

Listing 3: util.go

```
package main
func mask(s string) string {
  masked := []byte(s)
  tomask := false
  for i := 0; i < len(s); i++ {
    if tomask {
      masked[i] = '*'
    } else {
      masked[i] = s[i]
    }
    if s[i] == ' ' {
      tomask = true
    }
  }
  return string(masked)
}
```

jumps to the code starting in line 26, which collects a new user password entry from standard input and appends it to the text of the previously decrypted password file.

No File, No Problem

To do this, line 14 displays the `Password:` prompt for collecting the master password. Line 15 reads it using the standard *terminal* package via its `ReadPassword()` function. The `ReadPassword()` function turns off the terminal's echo, so the user can type the password without it being displayed. If the password does not match the one originally set for the password file, `readEnc()` in line 19 fails and `panic()` in line 22 aborts the program. But if `readEnc()` fails because the password file does not yet exist, line 21 traps this and tells the program to continue until either a new entry is appended later or the empty file is displayed in the UI.

On the text of the decrypted file, line 33 uses `TrimSuffix()` to remove the last newline character and then employs `Split()` to split the whole blob into an array of line strings; both functions are from the standard *strings* package. Line 34 then passes the array to the `runUI()` function, telling it to launch the UI. The UI keeps running until the user hits the quit button, ending the main program, and shutting down the UI.

Listing 4: pv.go

```
01 package main
02 import (
03   "bufio"
04   "errors"
05   "flag"
06   "fmt"
07   "golang.org/x/crypto/ssh/terminal"
08   "os"
09   "strings"
10 )
11 func main() {
12   add := flag.Bool("add", false, "Add new password entry")
13   flag.Parse()
14   fmt.Printf("Password: ")
15   password, err := terminal.ReadPassword(int(os.Stdin.Fd()))
16   if err != nil {
17     panic(err)
18   }
19   txt, err := readEnc(string(password))
20   if err != nil {
21     if !errors.Is(err, os.ErrNotExist) {
22       panic(err)
23     }
24   }
25   if *add {
26     fmt.Printf("\rNew entry: ")
27     reader := bufio.NewReader(os.Stdin)
28     entry, _ := reader.ReadString('\n')
29     txt = txt + entry
30     writeEnc(txt, string(password))
31     return
32   }
33   lines := strings.Split(strings.TrimSuffix(txt, "\n"), "\n")
34   runUI(lines)
35 }
```

Two Widgets

The terminal UI, like in the previous Snapshot columns, uses the *termui* package from GitHub. Listing 5 calls `ui.Init()` in line 12 to initialize its functions and calls `ui.Close()` to acknowledge a user abort in the `defer` statement in line 15. This neatly folds up the UI to leave a usable terminal for the shell.

The terminal UI shown in Figure 1 consists of two stacked widgets: On top, there is a listbox with the password entries, which the user can scroll through. It also supports leafing through multiple pages if the list of entries is longer than the maximum number of lines displayed. Below the listbox, at the bottom of the terminal window, a paragraph widget indicates which keys the user can press next: Enter reveals the selected password, while *Q* exits the program.

To allow the UI to take advantage of the entire geometry of the terminal window, line 25 queries the window dimensions, using the `TerminalsDimensions()` helper function from the *termui* package. From the width and height of the window, lines 26 and 27 then determine the position and dimensions of the two stacked widgets. In this case, the paragraph widget is assigned the bottom three lines, while the listbox on top gets everything else. Horizontally, both widgets extend to the edges of the terminal window.

The listbox entries reside in the `Rows` attribute of the listbox as an array slice of strings. Line 17 populates this array with the `rows` array slice. Before this happened, the `for` loop starting in line 9 stuffed all masked entries into `rows` but kept the original lines in `lines`. The two array slices for masked and unmasked entries make it easy to later reveal masked entries: The code only needs to look at the same index number in the original slice to reveal the unmasked content.

After line 28 has drawn the widgets on screen, line 29 fires off the `PollEvents()` Go routine, which will intercept all of the user's keystrokes concurrently from now on and send them to the `uiEvents` channel. From there, the program fetches events via the `select` statement in the infinite `for` loop starting in line 30 and immediately responds to all incoming keystrokes. If the user presses *K* to scroll up, line 35 uses `hideCur()` (starting in line 51) and the `mask()` function to hide a password that may have been previously revealed in the current listbox entry. Then, `ScrollUp()` (line 36) tells the listbox to scroll up one item, and the subsequent `Render` command smoothly displays the change in the UI. The same applies to pressing *J*, which lets the user scroll down through the list of entries.

Line 44 intercepts presses of the Enter key and calls the `showCur()` function defined in line 55. The function fetches the original unmasked password entry from the `lines` list and replaces the currently selected line of the listbox with it. And, hey presto, account and password are displayed on the screen in clear text. `hideCur()` starting in line 51 does the opposite and hides the current entry using the `mask()` function when the user moves away.

Listing 6: Compiling the Program

```
$ go mod init pv
$ go mod tidy
$ go build pv.go crypto.go util.go ui.go
```

Installation

As always, the binary can be generated from the Go code using the typical three-step process (Listing 6). This process fetches all the dependent libraries from GitHub, compiles them, and binds everything together to create the finished `pv` binary. You can then copy this to any target computer with a similar architecture. It'll run there without complaints, and it also conveniently even conjures up the UI into the terminal on remote machines. You will want to copy the `test.age` password file to a file in your home directory for production operation; the password reminder is then ready for use. ■■■

Info

[1] Age: *https://github.com/FiloSottile/age*

[2] "What did Ken Thompson mean when he said, 'I'd spell creat with an "e".'?": *https://unix.stackexchange.com/questions/10893/what-did-ken-thompson-mean-when-he-vsaid-id-spell-creat-with-an-e*

Listing 5: ui.go

```
01 package main
02 import (
03   "fmt"
04   ui "github.com/gizak/termui/v3"
05   "github.com/gizak/termui/v3/widgets"
06 )
07 func runUI(lines []string) {
08   rows := []string{}
09   for _, line := range lines {
10     rows = append(rows, mask(line))
11   }
12   if err := ui.Init(); err != nil {
13     panic(err)
14   }
15   defer ui.Close()
16   lb := widgets.NewList()
17   lb.Rows = rows
18   lb.SelectedRow = 0
19   lb.SelectedRowStyle = ui.NewStyle(ui.ColorBlack)
20   lb.TextStyle.Fg = ui.ColorGreen
21   lb.Title = fmt.Sprintf("passview 1.0")
22   pa := widgets.NewParagraph()
23   pa.Text = "[Q]uit [Enter]reveal"
24   pa.TextStyle.Fg = ui.ColorBlack
25   w, h := ui.TerminalDimensions()
26   lb.SetRect(0, 0, w, h-3)
27   pa.SetRect(0, h-3, w, h)
28   ui.Render(lb, pa)
29   uiEvents := ui.PollEvents()
30   for {
31     select {
32     case e := <-uiEvents:
33       switch e.ID {
34         case "k":
35           hideCur(lb)
36           lb.ScrollUp()
37           ui.Render(lb)
38         case "j":
39           hideCur(lb)
40           lb.ScrollDown()
41           ui.Render(lb)
42         case "q", "<C-c>":
43           return
44         case "<Enter>":
45           showCur(lb, lines)
46           ui.Render(lb)
47       }
48     }
49   }
50 }
51 func hideCur(lb *widgets.List) {
52   idx := lb.SelectedRow
53   lb.Rows[idx] = mask(lb.Rows[idx])
54 }
55 func showCur(lb *widgets.List, lines []string) {
56   idx := lb.SelectedRow
57   lb.Rows[idx] = lines[idx]
58 }
```

MakerSpace

Shell Programming in Python

Snake Shell

Create lightweight Raspberry Pi scripts with Xonsh, a Python shell that lets you write scripts in Python with Bash commands mixed in. *By Brooke Metcalfe and Pete Metcalfe*

For Raspberry Pi users, Xonsh [1] offers many opportunities to write some extremely lightweight scripts, with Python connecting to physical devices and Bash utilities accessing system and file resources.

In this article, we look at two lean Xonsh projects for the Raspberry Pi. The first program connects to a DHT11 temperature and humidity sensor and shows the results in a Bash dialog in just five lines. The second project calls the Bash top utility in a lean eight lines to show the Raspberry Pi idle time and user time on a 16x2 LCD screen.

Getting Started

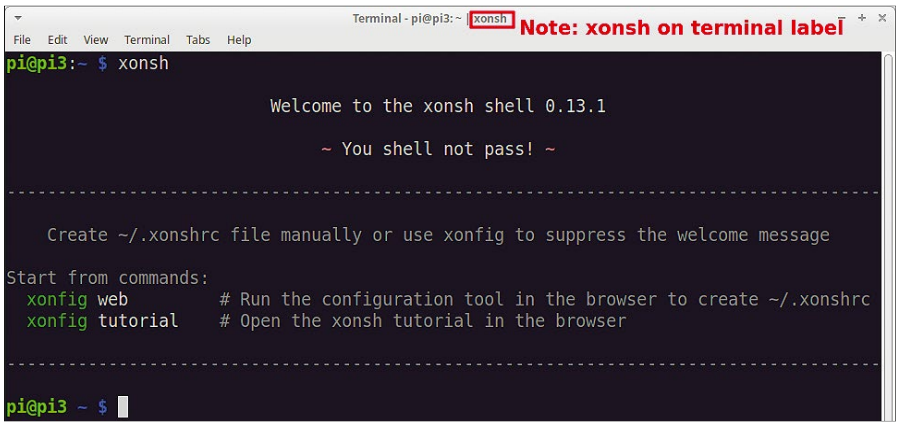
Xonsh has two requirements: Python 3.8 or greater and a Bash shell. To install and run Xonsh on a Raspberry Pi, Ubuntu, or Debian system, enter:

```
sudo apt install xonsh
xonsh
```

When the Xonsh shell opens, the terminal label changes so you can see that you are working in Xonsh rather than a standard Bash shell (Figure 1). When the shell first starts up, two options are presented: *xonfig tutorial*, which opens a browser help window, and *xonfig web*, which allows users to tweek the xonsh shell configuration.

Within the Xonsh shell, you can enter both Bash and Python statements, such as:

```
$ # Mix Bash and Python Lines
$ echo "Time is:" $(date +%T) ; # Bash
Time is: 12:59:51

$ print("%d + %d = %d" % ⮐
        (2,3,2+3)) ; # Python
2 + 3 = 5
```

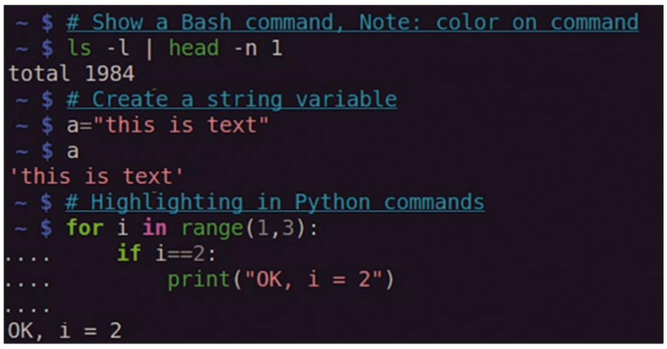

Terminal - pi@pi3: ~
xonsh
Note: xonsh on terminal label
File Edit View Terminal Tabs Help

```
pi@pi3:~ $ xonsh

                 Welcome to the xonsh shell 0.13.1

                      ~ You shell not pass! ~

------------------------------------------------------------------------------

    Create ~/.xonshrc file manually or use xonfig to suppress the welcome message

Start from commands:
  xonfig web         # Run the configuration tool in the browser to create ~/.xonshrc
  xonfig tutorial    # Open the xonsh tutorial in the browser

------------------------------------------------------------------------------

pi@pi3 ~ $
```

Figure 1: The Xonsh terminal.

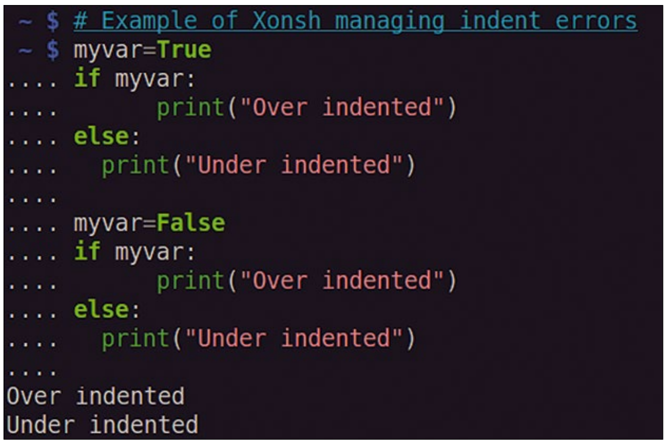
Xonsh in the Terminal

The Xonsh shell is designed for Python users, so some handy features are available from the command prompt. The first of these features is code highlighting. Xonsh highlights comments, strings, functions, and command statements (Figure 2).



```
~ $ # Show a Bash command, Note: color on command
~ $ ls -l | head -n 1
total 1984
~ $ # Create a string variable
~ $ a="this is text"
~ $ a
'this is text'
~ $ # Highlighting in Python commands
~ $ for i in range(1,3):
....     if i==2:
....         print("OK, i = 2")
....
OK, i = 2
```

Figure 2: Xonsh highlights Bash and Python statements.



```
~ $ # Example of Xonsh managing indent errors
~ $ myvar=True
.... if myvar:
....       print("Over indented")
.... else:
....   print("Under indented")
....
.... myvar=False
.... if myvar:
....       print("Over indented")
.... else:
....   print("Under indented")
....
Over indented
Under indented
```

Figure 3: Xonsh manages Python indentation errors.

Xonsh also handles Python indentation errors for single-line statements and non-control statements. Figure 3 shows an example of Python code that successfully runs with both overindented and underindented statements.

The next useful feature is Python help (Figure 4). The Tab key offers a drop-down dialog of Python objects or methods from the typed string. The arrow keys let you select the required help options within the dialog.

Bash Zenity Dialogs with Python

Python has some great graphic libraries such as Tkinter, Qt, and PySimpleGUI. These libraries are excellent for complex GUI applications, but they can be overkill if you only need a simple dialog.

The Zenity utility [2] is a quick way to present a variety of dialogs in just one line of Bash code. Zenity is preloaded on Raspberry Pi OS and most Linux installations, so no added installation is usually required.

The basic syntax for a Zenity information dialog is:

```
$ zenity --info --title=myTitle ⮐
         --text=myMessage
```

In Xonsh, Python statements and variables are used directly in a Bash statement as:

```
@(<Python statement>)
```

For example, the Bash `echo` command can print a Python statement:

```
$ import sys
$ echo "OS: " @(sys.platform)
OS:  linux
```

These two steps can be combined to show the OS in a Zenity dialog:

```
$ import sys
$ zenity --info --title=System_OS ⮐
         --text=@(sys.platform)
```

Once you have these two concepts working (Zenity and Python embedded in Bash), you can start creating some quick and easy Raspberry Pi applications.

Raspberry Pi DHT11 Sensor Project

The DHT11 and DHT22 sensors are low-cost components (~$1 to $10) that measure ambient temperature and humidity. Figure 5 shows a typical DHT11 setup with the data pin (blue wire) set to GPIO17 (note that sensor pin arrangements may vary).

The DHT11/DHT22 Python library is installed by:

```
pip install Adafruit-DHT
```

Listing 1 uses the Python *Adafruit_DHT* library to get the temperature and humidity (line 9) and then passes the data variables to a Zenity info dialog (line 11).

To run this script, enter:

```
$ xonsh dht11_dlg.sh
```

```
$ # or make the script executable ⮐
    and then run it:
$ chmod +x dht11_dlg.sh
$ ./dht11_dlg.sh
```

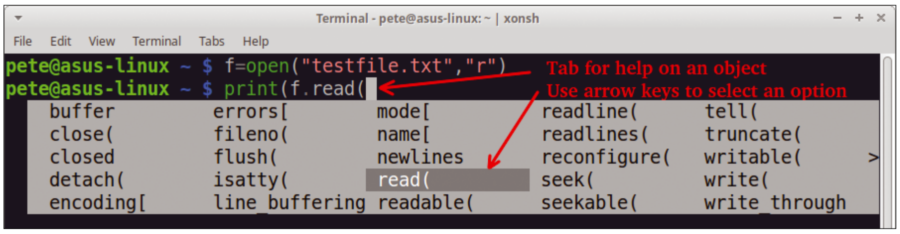
Figure 6 shows the project hardware with the Zenity temperature and humidity dialog. Once you have the hang of how to use Zenity dialogs, you can modify this project to work with a variety of other Raspberry Pi sensors, such as BMP180 barometers, moisture sensors, or passive infrared (PIR) motion sensors.

LCD Project

A number of different LCD screens can be connected to a Rasperry Pi. For this project, we used a 16x2 (two rows of 16 characters) I2C LCD screen ($3-$15), which has an easy four-wire setup (SDA, data line; SCL, clock line; VCC, power input; and GND, ground).

The first step on this project is to check and enable I2C (inter-integrated circuit, a serial communication bus) on the Raspberry Pi, which you can do by running:

```
# Check I2C status: 0=enabled, ⮐
                    1 = disabled
sudo raspi-config nonint get_i2c
# Enable I2C: 0=enabled, 1 = disabled
sudo raspi-config nonint do_i2c 0
```


Terminal - pete@asus-linux: ~ | xonsh
File Edit View Terminal Tabs Help
Tab for help on an object
Use arrow keys to select an option

```
pete@asus-linux ~ $ f=open("testfile.txt","r")
pete@asus-linux ~ $ print(f.read(
    buffer          errors[          mode[           readline(       tell(
    close(          fileno(          name[           readlines(      truncate(
    closed          flush(           newlines        reconfigure(    writable(   >
    detach(         isatty(          read(           seek(           write(
    encoding[       line_buffering   readable(       seekable(       write_through
```

Figure 4: Xonsh offers Python help.

The next step is to connect the LCD to the Raspberry Pi. The LCD lights up after it is wired to the Pi. The `i2cdetect` utility shows active I2C hardware addresses (Figure 7).

A Python library for this hardware is installed with:

```
sudo pip3 install rpi_lcd
```

The *rpi_lcd* library assumes an I2C address of 27. Unfortunately the LCD device used was at address 3F, so you have to take an extra step to change the address:

```
# change the LCD default address ↩
  from 27 to 3f
cd /usr/local/lib/python3.9/↩
  dist-packages/rpi_lcd
sudo sed ↩
  -i 's/address=0x27/↩
      address=0x3f/' __init__.py
```

To test that the LCD screen is working, enter a few lines at the Xonsh prompt:

```
from rpi_lcd import LCD
lcd = LCD()
lcd.text("Hi from Xonsh",1)
```

The goal for this project was to show the Raspberry Pi's user time and idle time on the LCD screen. The `top` utility displays Linux running processes and CPU stats. By piping the output from `top` to a `grep` statement, you can isolate a line of just the CPU stats. A third pipe with `awk` creates a string with either the user time or the CPU idle times (Figure 8).

The output from these long Bash statements can be stored in variables to be used directly in Xonsh Python. The script in Listing 2 gets the user time (line 10) and the idle time (line 12) and shows these strings on the LCD display (lines 14, 15). The display is updated every five seconds by looping with a Python *while* statement (line 8) and delaying with a Bash *sleep* statement (line 16). By mixing Python and Bash in Xonsh, this script only requires eight lines of code. Figure 9 shows the LCD project with live data.

The 16x2 LCD display is a great device for a variety of projects. We had a lot of fun experimenting with other Bash utilities (e.g., `df`, `iostat`, `vmstat`, and `ifstat`) and displaying the result on the LCD screen.

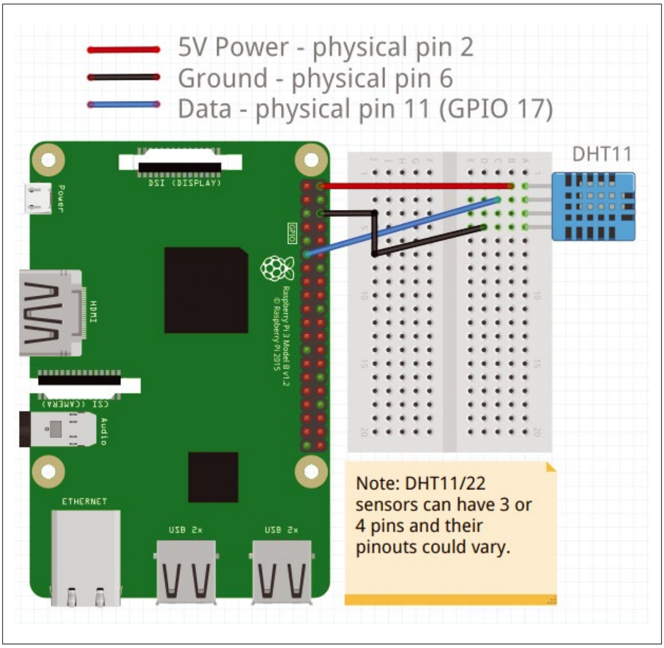
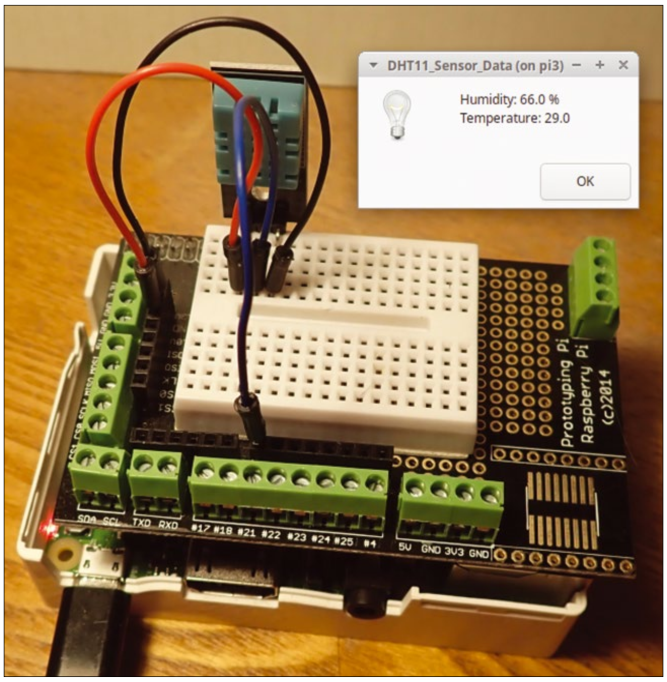

5V Power - physical pin 2
Ground - physical pin 6
Data - physical pin 11 (GPIO 17)
DHT11
DSI (DISPLAY)
Power
HDMI
CSI (CAMERA)
Audio
ETHERNET
USB 2x
USB 2x
GPIO
Raspberry Pi 3 Model B v1.2
Note: DHT11/22 sensors can have 3 or 4 pins and their pinouts could vary.

Figure 5: DHT11 sensor wired to a Raspberry Pi.

Listing 1: dht11_dlg.sh

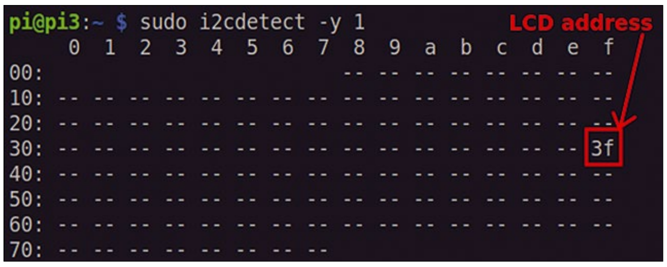
```
#!/home/pi/.local/bin/xonsh
#
# dht11_dlg.sh
    - using xonsh to show DHT11 sensor data in a dialog
#
import Adafruit_DHT
sensor=Adafruit_DHT.DHT11
gpio=17
# Get the humidity and temperature
humidity, temp = Adafruit_DHT.read_retry(sensor, gpio)
# show the data in a Zenity info dialog
zenity --info --title=DHT11_Sensor_Data  --width=150
          --text=@( "Humidity: {} %\nTemperature: {}".
              format(humidity,temp) )
```


DHT11_Sensor_Data (on pi3)

Humidity: 66.0 %
Temperature: 29.0

OK

Figure 6: DHT11 temperature and humidity project set up on a Raspberry Pi prototyping HAT (hardware attached on top) and breadboard.



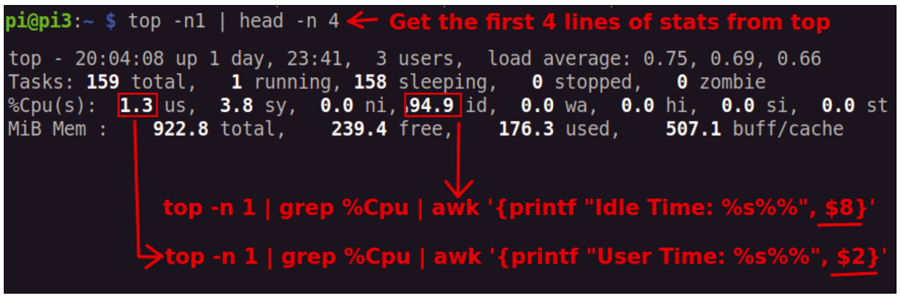
```
pi@pi3:~ $ sudo i2cdetect -y 1
     0  1  2  3  4  5  6  7  8  9  a  b  c  d  e  f
00:                         -- -- -- -- -- -- -- --
10: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
20: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
30: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- 3f
40: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
50: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
60: -- -- -- -- -- -- -- -- -- -- -- -- -- -- -- --
70: -- -- -- -- -- -- -- --
```

Figure 7: The i2cdetect utility shows I2C connected.

Summary

Raspberry Pi users could really reap some benefits from lightweight Xonsh scripts. We found that the utility allowed us to do some manual testing at the command prompt and create some very lean code examples.

Because Xonsh is a Python shell, all control statements (e.g., `while` and `for` loops, `if` statements) should be executed in Python and not in Bash. ∎



```
pi@pi3:~ $ top -n1 | head -n 4 <- Get the first 4 lines of stats from top
top - 20:04:08 up 1 day, 23:41,  3 users,  load average: 0.75, 0.69, 0.66
Tasks: 159 total,   1 running, 158 sleeping,   0 stopped,   0 zombie
%Cpu(s):  1.3 us,  3.8 sy,  0.0 ni, 94.9 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   922.8 total,   239.4 free,   176.3 used,   507.1 buff/cache

          top -n 1 | grep %Cpu | awk '{printf "Idle Time: %s%%", $8}'

          top -n 1 | grep %Cpu | awk '{printf "User Time: %s%%", $2}'
```

Figure 8: Use Bash to get user and idle time.

Listing 2: top_2_lcd.sh

```
#!/home/pi/.local/bin/xonsh
#
# top_2_lcd - show Bash Top data on a LCD screen
#
from rpi_lcd import LCD
lcd = LCD()

while True:
  # Get the user CPU time
  user=$(top -n 1 | grep %Cpu | awk '{printf "User Time: %s%%", $2}')
  # Get the total idle time
  idle=$(top -n 1 | grep %Cpu | awk '{printf "Idle Time: %s%%", $8}')
  # Show user time on line 1, Idle time on line 2
  lcd.text( user , 1)
  lcd.text( idle , 2)
  sleep 5
```

Info

[1] Xonsh: *https://xon.sh/*

[2] Zenity documentation: *https://help.gnome.org/users/zenity/stable/*

Author

Brooke Metcalfe is in her third year of Environmental Engineering at Carleton University in Ottawa, Canada.

Pete Metcalfe is a software engineer and Brooke's father. They have worked together on Raspberry Pi and software projects since Brooke was 12 years old.

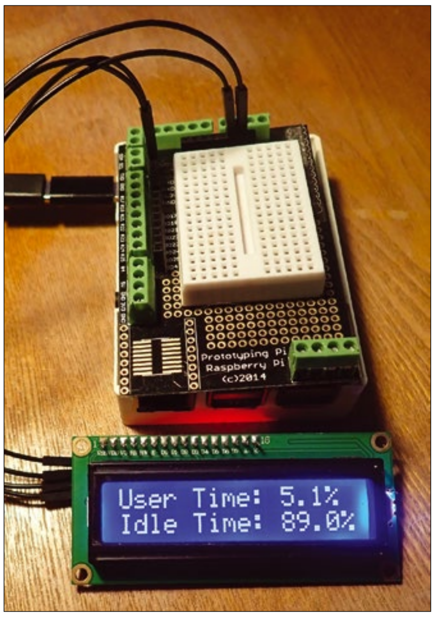
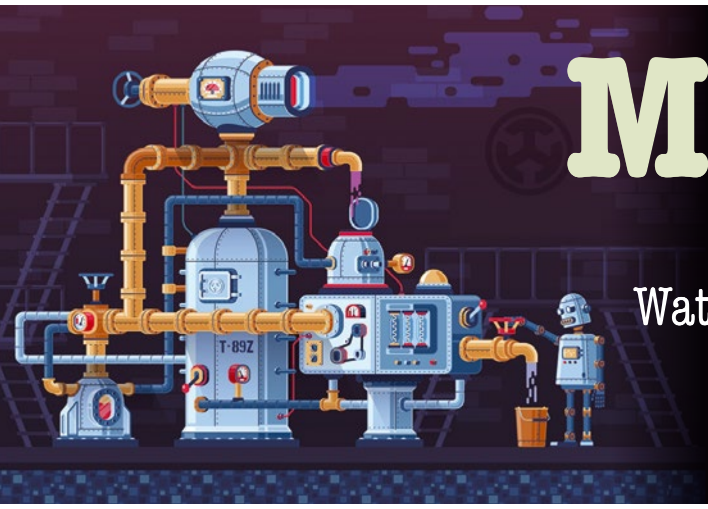


Figure 9: Show Bash strings on an LCD screen.



MakerSpace

Water your plants with a Raspberry Pi

Watering Pi

With a Pi Zero and a few components, you can build an inexpensive and reliable automatic watering system for your plants in next to no time. By Swen Hopfe

Whether in an apartment, on a balcony, in a greenhouse, or in a garden, if you are not at home and want to water your plants remotely, an automated system is your only option. In this article, I show you how to harness the power of a Raspberry Pi Zero as a reliable helper to manage the watering system.

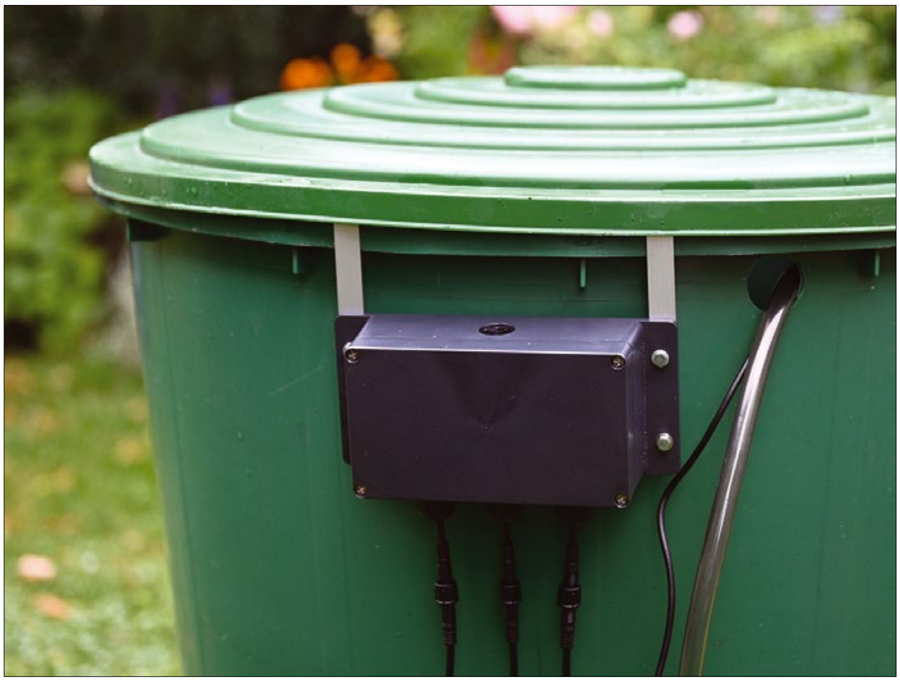


Figure 1: The control unit and supply line assembled on the 200-liter storage tank.

Design

Automatic irrigation is nothing new. These systems not only exist on a large scale in agriculture and horticulture but have also been available for many years for domestic use. Some providers attach their systems directly to the water supply. In this project, I'll instead draw water from a 200-liter tank reservoir with a supply that will last for a couple of dry days (Figure 1). The advantage of a tank is that you do not have to deal with a pressure line and you do not lose an uncontrolled amount of water in the event of an accident. The aim is to create a robust solution that you can tailor entirely to your own needs, thanks to the flexibility of the Raspberry Pi and your own hardware and software.

The storage tank for just a room can be much smaller; in fact, a water bucket is all you need, with a submersible pump and a riser to deliver the irrigation water. Two pipes supply water to plants through a branch distributor. The whole thing can be controlled on demand by solenoid valves. The project design also uses two moisture sensors to check that the water reaches the plants. (See the "Parts List" box.)

Getting Started

To get the Raspberry Pi up and running, it's a good idea to download a new Pi

Parts List

- Raspberry Pi Zero W (model 1 or 2)
- Centrifugal pump with 1.2m head
- Solenoid valves (x2)
- Relay modules (x3)
- ADC (ADS1115, or similar)
- Moisture sensors (x2)
- Plugin power supply (5V)
- Housing, wiring
- Various hoses and clamps
- Adapters, control valves, ground spikes

OS image in the usual way and transfer it to a microSD card. With the screen and keyboard plugged in, the boot options and network settings can be configured. A desktop environment is not needed, which is why I went for automated login from the CLI with SSH enabled. All further settings can then be configured in a terminal window from a computer on the same network. The Raspberry Pi needs a hostname that reminds you of the task in hand (e.g., *watering* in this case). The next step is to create a separate folder for the project files in your home directory.

Structure

The main electronics of the control system will live in a ready-made housing with a rubber seal in the lid and a terminal strip, on which all the external wires will be patched (Figure 2). Two small support plates are bolted on inside to fasten all the modules securely.

The housing for the control unit holds the Raspberry Pi, an analog-to-digital converter (ADC) module, and three relays. The external power supply, connections for two soil moisture sensors, two solenoid valves, and the pump are routed in from below with three four-core lines. At the top, where I sealed the unused screw hole with transparent material, an LED indicates operational readiness.

The Raspberry Pi needs a total of four GPIO pins for the control PINs on the relay boards and the signal LED; the ADC is connected over the I2C bus. The internal terminal strip routes the converter's analog inputs with the humidity sensors and the switching outputs of the relays for the pump and solenoid valves to the outside. The schematic (Figure 3) and associated program files are on the GitHub page for the project [1] [2].

Now it's time to connect the external hardware, pump, and solenoid valves to the hose material (Figure 4). Because no suitable distributor was available downstream of the pump, I soldered one myself from brass. The two control valves allow for a two-way system that can supply two groups of plants with different water needs.

Thanks to the valves, the two main lines can also be reliably blocked against the reservoir being evacuated in idle

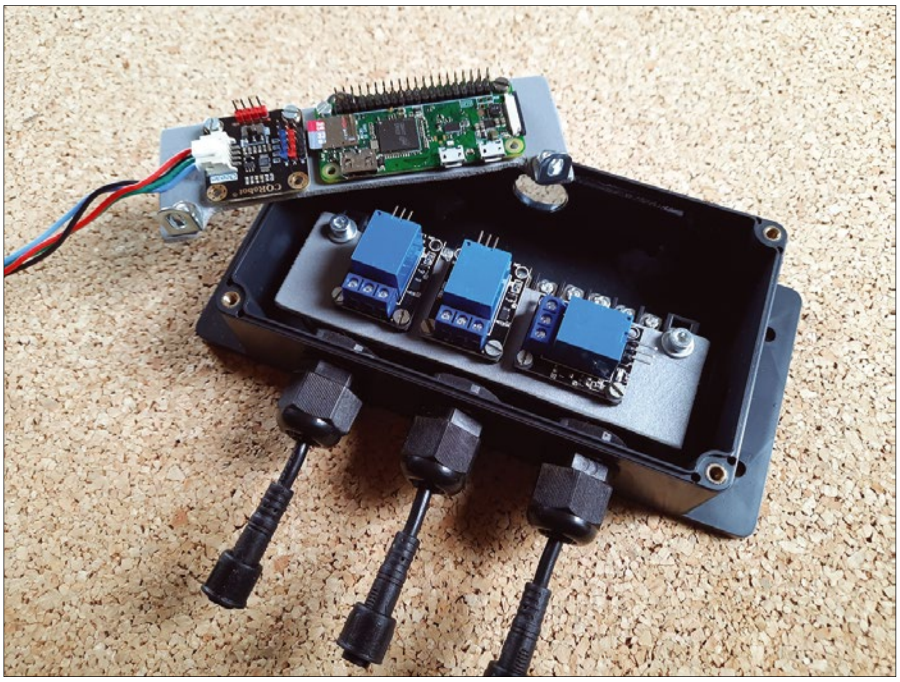


Figure 2: Installing the hardware in the waterproof housing of the control unit.

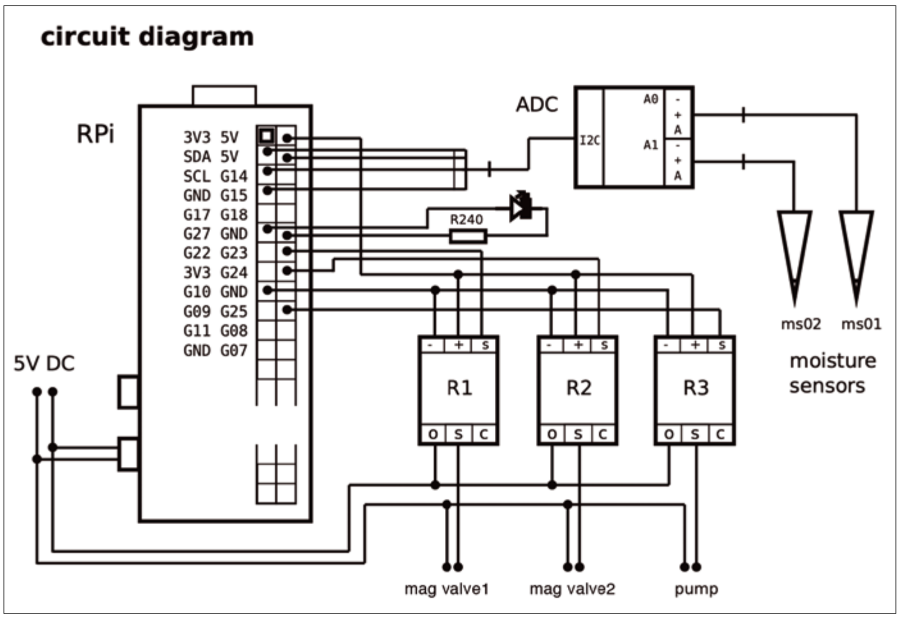


Figure 3: Project schematic.

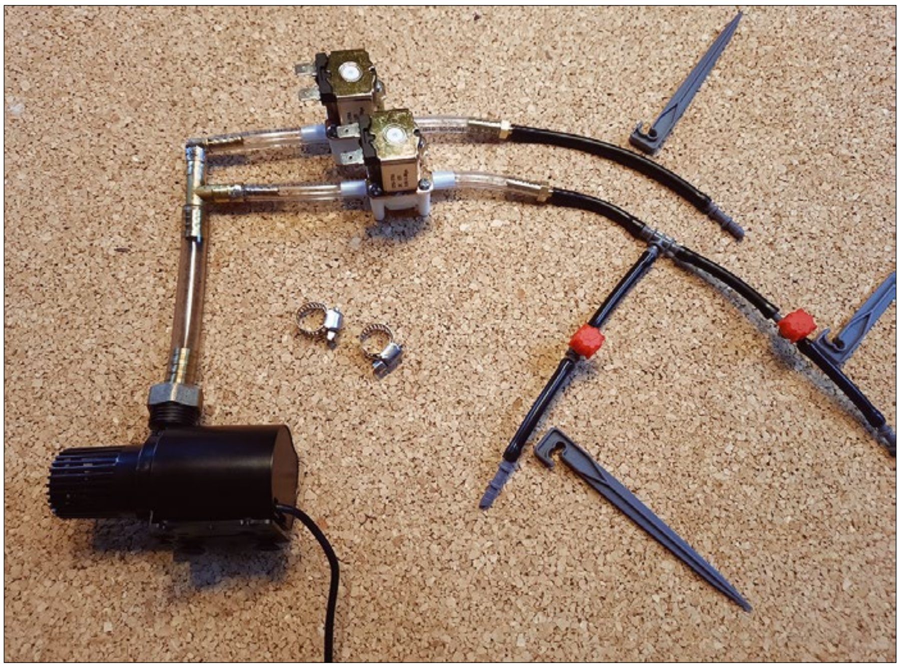


Figure 4: Shorter paths: the simple connection layout for the water supply.

condition. Distribution downstream can be done with a smaller hose diameter; T-pieces are used for more branches. Small regulators ensure that water reaches all the plant locations evenly; setting them up requires some trial and error. For an initial test, I first extended the connection on the centrifugal pump to allow immersion.

If everything works satisfactorily, you can then proceed to trim hoses and cables to the required lengths, route them correctly, and fasten them with ground spikes. Hose clamps around the thicker hoses reliably prevent them slipping off the connections while you are away.

Control

If you look at commercial irrigation systems, they often seem relatively complicated to operate. In contrast, I want this system to be as simple as possible, so the DIY system remains manageable and works safely during periods of absence from home. That said, the automatic watering system does have one special trick up its sleeve: It can extend the watering time to achieve a certain level of moisture in the soil with the use of sensors that provide appropriate feedback from two different plant locations.

Capacitive sensors were chosen because they are less susceptible to corrosion than resistive sensors. Although I am talking about models with analog output, they can be connected quite easily to the Raspberry Pi in the control center through the ADC. To determine the various limits up front, I immersed the sensors in water in a test (Figure 5) and then put them in moist soil. The ADC in this setup is an ADS1115, which is addressed in Python 3 with a library by Adafruit. The easiest way to install it is to use the command:

```
pip3 install Adafruit_ADS1x15
```

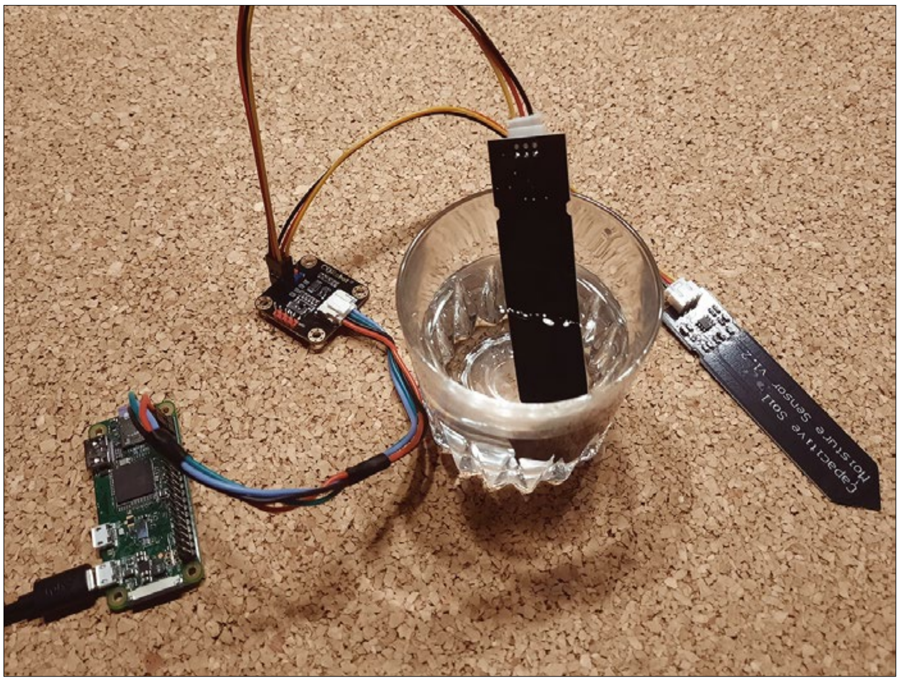


Figure 5: To determine limits, test the humidity sensors attached to the ADC.

This solution relies on a central control script named `watering.py`, which resides in the Raspberry Pi's memory and takes care of flow control and polling the humidity sensors. The timing settings are stored there, as well. In an initial saver mode, the system now waters the plants for two minutes at 5:00pm every day. Controlled by the second solenoid valve, it then does the same thing for a little longer – five minutes. The values can be adjusted as required.

To water the plants, the Raspberry Pi always opens both solenoid valves first and then starts pumping from the tank. Once the timer for the first water delivery has expired, the system checks the first moisture sensor to discover whether the soil moisture at the plant location has reached the "wet" state; if not, follow-up is provided. The first valve then shuts off. The second line remains open for a longer time and is controlled along the same principle until the script first stops the pump and then finally shuts off the second valve.

The Python script resides in a separate folder below the home directory. Because it is called in `rc.local`, it runs automatically at boot time. If you like, you can also convert the script into a genuine daemon. Everything has been kept quite simple here, but it's for daily use and can be extended at any time if you feel the urge to do so. The script and the remote control PHP website described below can be found in my GitHub project [1], where the wiring and routing diagram for the water supply also reside.

Because the Raspberry Pi is connected to the local network, you can also access and maintain it there over SSH. To do so, just log in to a terminal with:

```
ssh pi@watering
```

In contrast to similar projects, however, the automatic watering device does not have a separate terminal-based menu. At the end of the day, it is a simple workhorse that always follows the same routine. The idea is to do everything else that needs to be done in the web browser.

Web-Based Remote Control

The watering system works reliably even without Internet access, but it should at least have remote access to avoid the need to read everything at the device or to install buttons and a touchscreen. You have to remember that everything needs to be pretty much waterproof.

If the irrigation system in your garden is outside the range of your router, a repeater set up halfway can help. Alternatively, during the test phase, simply disconnect the control unit from the connectors and reprogram the Raspberry Pi on your home network. A network connection is always required for remote access, so you can intervene, even when on vacation.

Some of today's commercial watering systems come with a smartphone app. I

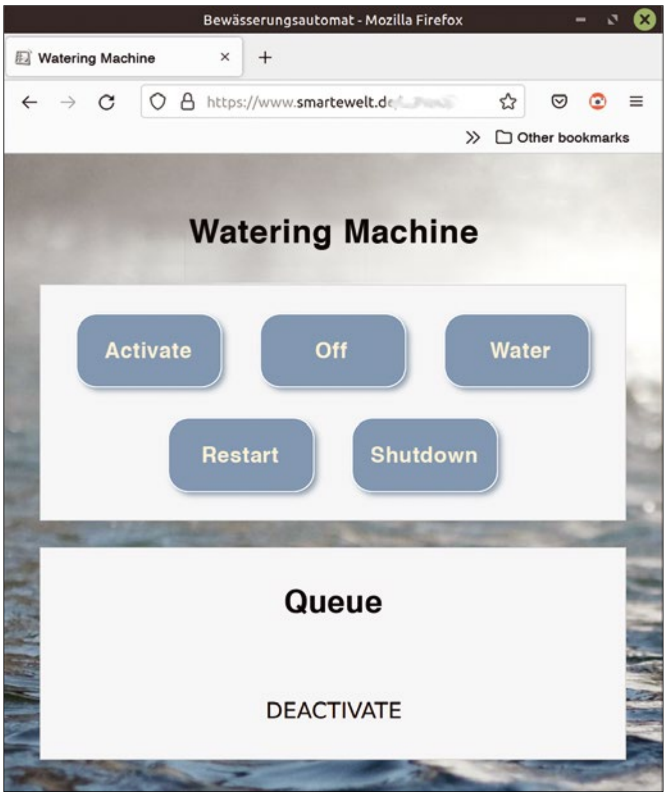


Figure 6: Web-based control.

Table 1: Web Interface Functions

Name	Function
`Activate`	Enable watering timer (default: on).
`Off`	Disable timer. System remains switched on.
`Water`	Immediate water delivery for 10 seconds through both lines.
`Restart`	Reboots the system. The web interface is active again after restarting the Raspberry Pi.
`Shutdown`	Shut down the Raspberry Pi. A manual off-on cycle is then required.

designed a web interface that also works on cell phones. Without the benefit of a vendor-operated cloud, the Raspberry Pi, including the HTTP server, needs to be accessible over a static IP address on the Internet.

However, I went for a simpler variant and located the user interface (Figure 6) in a PHP file in my own web space. After calling the URL, the interface sends predefined commands to the watering system at the push of a button. Currently, the five actions in Table 1 are implemented. The Raspberry Pi queries the corresponding commands in the control script at short intervals and triggers the matching actions if needed.

The basis for the implementation is a development by Christian Grieger [3]. The advantage of his method is that it does not require elaborate arrangements such as dynamic DNS. The disadvantage is that feedback arrives with a bit of a delay; however, this delay is not likely to be of much importance, although you will definitely want to secure the remote control website with a login.

Conclusions

The plant watering system does exactly what it's supposed to do, in a positive way. I ran it at home in sunny weather during the last few days of summer, and checking it for functionality was no problem (Figure 7).

The system will star as my vacation stand-in next year. Trying to supply the whole garden would probably be overshooting the mark, but if you are looking to rescue your most important plants in a limited area (e.g., a greenhouse, conservatory, or living room), this controller can be a real helper. ■■■

Info

[1] Project on GitHub: *https://github.com/swenae/watering*

[2] English version of code and image files: *https://linuxnewmedia.thegood.cloud/s/5Rzx9tQW2FJ6N3Z*

[3] Electronics projects by Christian Grieger: *https://elektro.turanis.de* (in German)

Author

Swen Hopfe works for a medium-sized company with a focus on smart cards and near-field communication (NFC). When he is not taking photos, in the great outdoors, or in his garden, he focuses on topics such as the Raspberry Pi, Internet of Things, and home automation.

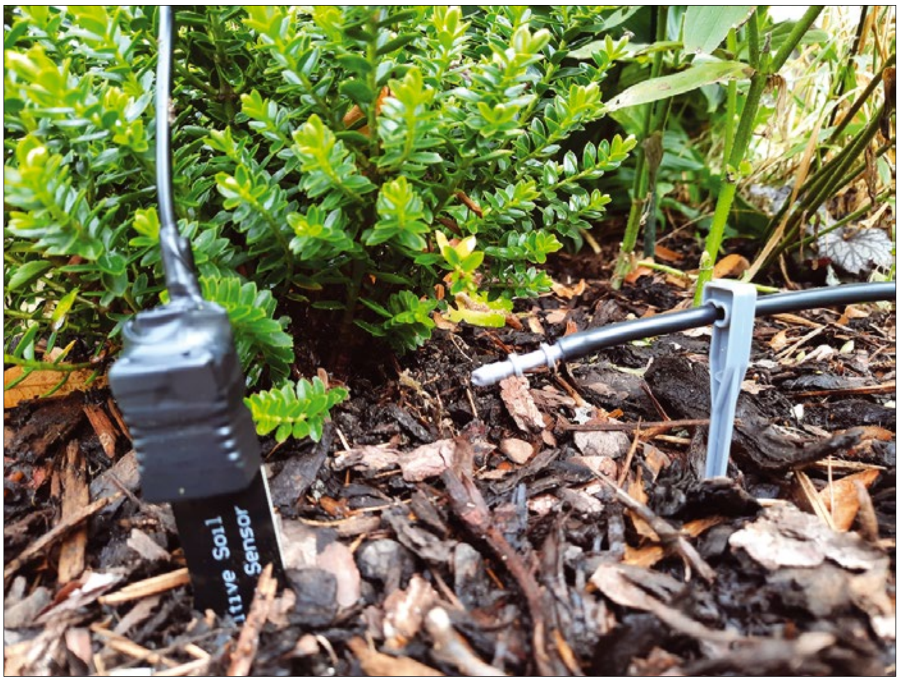
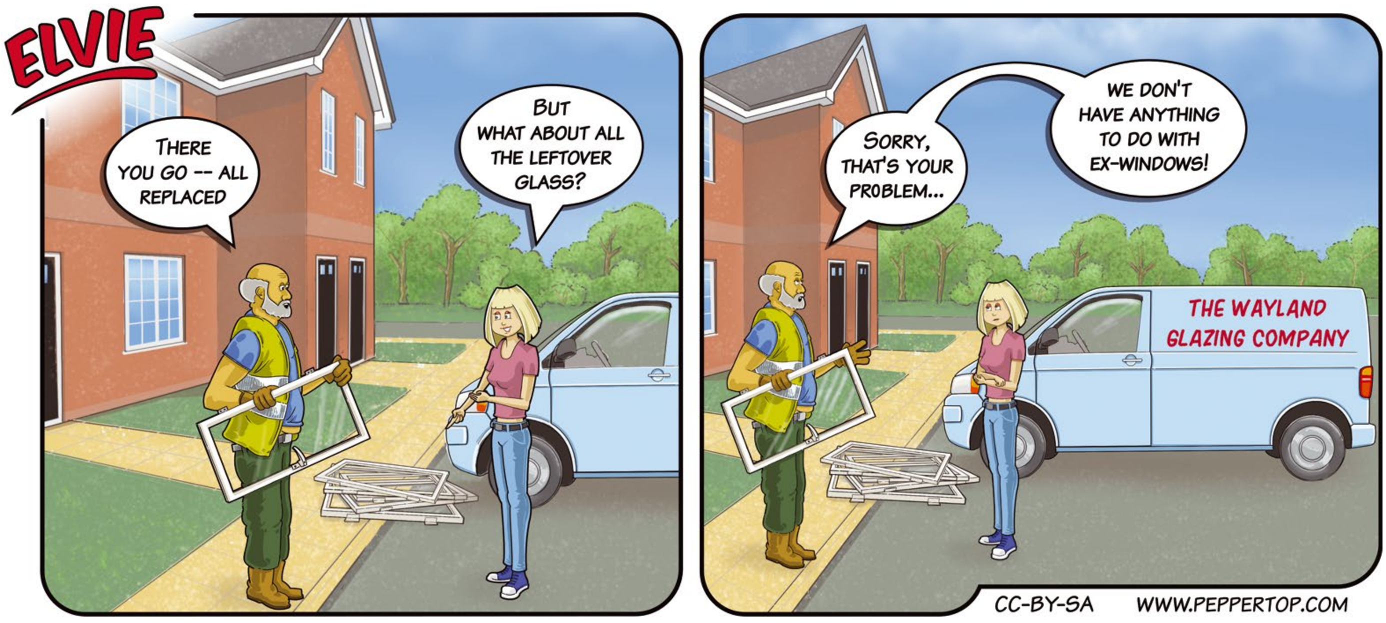


Figure 7: The water outlet and the sensor spike in the plant bed.

Linux is running right now in millions of homes, for users who just need a reliable desktop system for office work, gaming, email, and other tasks. But everyone knows that Linux is also a fabulous server system. If you need a server, but you don't have the time or inclination to become a Linux server administrator, YunoHost might be your best bet. The goal of YunoHost is to make it very easy to set up and use a Linux server. In many cases, YunoHost lets you install a service with a single mouse click. This month we introduce you to Linux server management with YunoHost. Also in this month's Linux Voice, manage your finances with Skrooge and manage your processes with btop++.

Image © Olexandr Moroz, 123RF.com

LINUXVOICE


ELVIE
THERE YOU GO -- ALL REPLACED
BUT WHAT ABOUT ALL THE LEFTOVER GLASS?
SORRY, THAT'S YOUR PROBLEM...
WE DON'T HAVE ANYTHING TO DO WITH EX-WINDOWS!
THE WAYLAND GLAZING COMPANY
CC-BY-SA
WWW.PEPPERTOP.COM

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

MADDOG'S DOGHOUSE

Free and open source software can be intimidating at first; a little guidance can help you on your way. BY JON "MADDOG" HALL

Getting started with commands

Recently, I was at a conference and I met a young person who wanted to start using free and open source software, but they did not know where to start.

They told me that they were a programmer and even could do simple programs in C, but they wanted an open source IDE (there are a couple) and they were "even willing to learn `vim` or `emacs`, but they are really hard."

Now I have been a `vim` person for a long time, mostly through a series of editors that first started using hardcopy terminals. While these editors were not written by the same people, each was typically compatible enough that I moved from one to the other as the new ones occurred. I was using `vi`/`ex` (very usable on a hardcopy terminal) when I migrated to `vi`/`ex` on character cell terminals that used termcap to translate strings of cursor-control output for positioning characters on the screen.

I do realize, however, that beginning to use a powerful editor like `vim` or `emacs` seems daunting. There are whole books on how to use these editors for what seems to be a relatively simple task: moving and editing text characters.

This is actually true for most of open source (including free) software and is multiplied by the fact that you can learn "below" what the developers actually expose to the end user. It is easy to become overwhelmed.

My advice to my young friend was to concentrate on a small subset of commands for `vim` (I could have easily substituted `emacs`, but I chose `vim`) to open a file, save a file, go into input mode, escape from input mode, move around in the full-screen mode, etc.

Over time you can learn about how to search for strings or move to the end of the line or the beginning with one or two keystrokes, but you can learn those as you go.

Should you try to remember all the commands? No, but you think "I bet that `vim` has this command, let me find it in the help facility."

Likewise this is true of GNU/Linux itself. While the power of Linux really lies beneath the graphical interface at what most people think of as "the command line" or "shell level," many people can do whatever they need to do without ever learning these levels of usage.

People can create files using the graphical text editor (on my system this is `xed`,) launched through the menu system, which is very mouse based. You can move through the directory structure using a graphical file manager. You can print on a file by clicking on it and choosing the function of "print" with your mouse. It's totally graphical, and many applications are available that work just that way, no command line needed.

However, to really experience the power of GNU/Linux, you have to be willing to search, experiment, and learn.

In 1977 I became a Unix systems administrator at Bell Laboratories. While I had worked on a variety of operating systems and hardware, I had never worked on a Unix system before. Bell Labs (the creators of Unix) hired me because I had demonstrated the ability to learn on my own and apply it.

Bell Labs sent me to a one-week course in Unix and set me up to have two mentors help me with the administration while I came on board.

One night I was sitting at the (paper) console of the Unix system, and I had hundreds of files that had to be updated by moving one element on each line of each file to another place on the same line of that file. I was patiently (and for people that know me, patience is not my strongest strength) editing one file after the other, doing this on each line. I estimated that it would take eight hours or more.

After about two hours of this, I stopped. I said to myself "I do not know that Unix has a command to do this, but I am willing to bet that it does." So I stopped my editing and started looking through the Unix manual.

After a short time I came across the command `cut`, which seemed to do what I wanted, but I needed more. At the bottom of the page I saw the words "See also paste." Looking at that page, I formulated a plan to use `cut` and `paste` to do what I needed. Twenty minutes later I walked out of the room.

I did not memorize everything that `cut` could do, or even the rest of the commands, but after that, once a year, I spent an hour or two just refreshing my mind about the different Unix commands.

I knew the rest of the commands would do what I needed when I needed it. ■■■

Set up self-hosted server services at the push of a button

Quick Service

Setting up a server manually on Linux can sometimes test your patience, but with YunoHost you can install and configure your servers with just a few mouse clicks.

BY ERIK BÄRWALDT

Linux is considered an excellent server operating system mainly because of its stability. But installing and configuring a server requires in-depth knowledge, even on the free operating system, and is something usually done at the command line. Enter YunoHost [1], which lets you set up a fully configured server on your intranet with just a few mouse clicks.

The new version 11 of YunoHost is available both as a standalone distribution and for use on virtual machines. The system supports both 32- and 64-bit hardware, meaning that you can recycle your older computer systems. On top of this, YunoHost is available for many ARM-based single-board computers, and even older Raspberry Pi or Orange Pi systems are suitable as the basis for a server on your network.

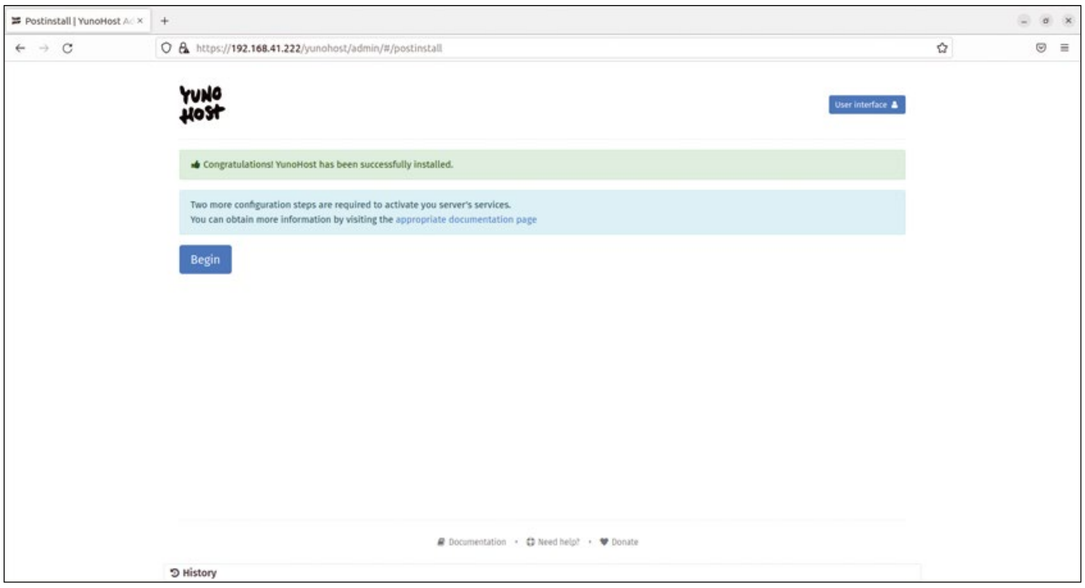
The software supports installation on a remote computer via SSH. However, this requires Debian 11 "Bullseye" as the preinstalled underpinnings on the remote machine. After installing the basic system, you can easily integrate the desired services via a web-based interface. The system is not limited to a couple of well-known services, but supports no fewer than 350 preconfigured programs.

Installation

The project's website lists the minimum hardware requirements for each platform to run YunoHost [2]. Because these requirements are very low both in terms of RAM (512MB) and in terms of storage (just 16GB free disk space), even older 32-bit computers with conventional hard disks are suitable for YunoHost operation. Detailed instructions for the different approaches to installing the system as a function of the target platform are available on the project site. But, basically, the Debian setup wizard is used in both text and graphics mode.

The basic installation sets up a database and a web server. On completing the setup, just reboot the system and you will see a login prompt. YunoHost is now ready for use. You then talk to the system via the graphical web interface from any device that has a web browser.

Figure 1: Use the browser-based wizard to complete the basic configuration.


Postinstall | YunoHost A
https://192.168.41.222/yunohost/admin/#/postinstall
YUNO HOST
User interface
Begin
Documentation • Need help? • Donate
History

Congratulations! YunoHost has been successfully installed.

Two more configuration steps are required to activate you server's services.
You can obtain more information by visiting the appropriate documentation page

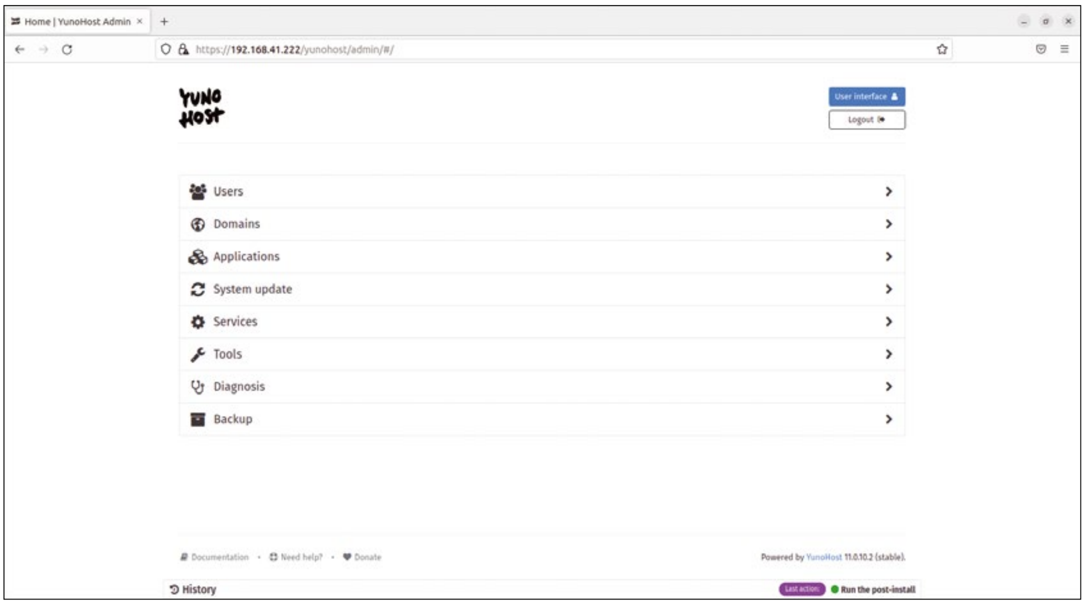


Figure 2: The web interface clearly groups all the settings on a single page.

To do this, simply type *https://YunoHost.local* or, alternatively, the IP address of the system in your browser's address bar. A setup wizard then launches with a start page reporting the successful installation of the server. Pressing the *Begin* button takes you to the configuration dialog (Figure 1).

In the first dialog you need to create a domain for the YunoHost server. If you don't have your own domain, the software offers you the option of generating a dynamic domain. The DynDNS service keeps your domain accessible on the Internet even if your IP address changes. After this, you are prompted for a new admin password with a length of at least eight characters. Finally, you need to log in as admin in the following window. This takes you to the admin start page, which lists different categories for configuring options (Figure 2).

The recommended approach is to click *System update* first after completing the initial installation. The software then determines the packages to be refreshed and lists them. You can start the update by clicking *Update all packages* at the bottom of the page. The routine updates both the operating system itself and the list of available applications. After completing the update, click on the small house icon in the top left corner to return to the main window.

Account Management

After the update, create a user account with extended authorizations which will act as an administrator in the future and can be reached by email. Then configure your new server to automatically send notifications to the administrator in case of problems or anomalies. You now also need to create the users who will have access to the server applications later.

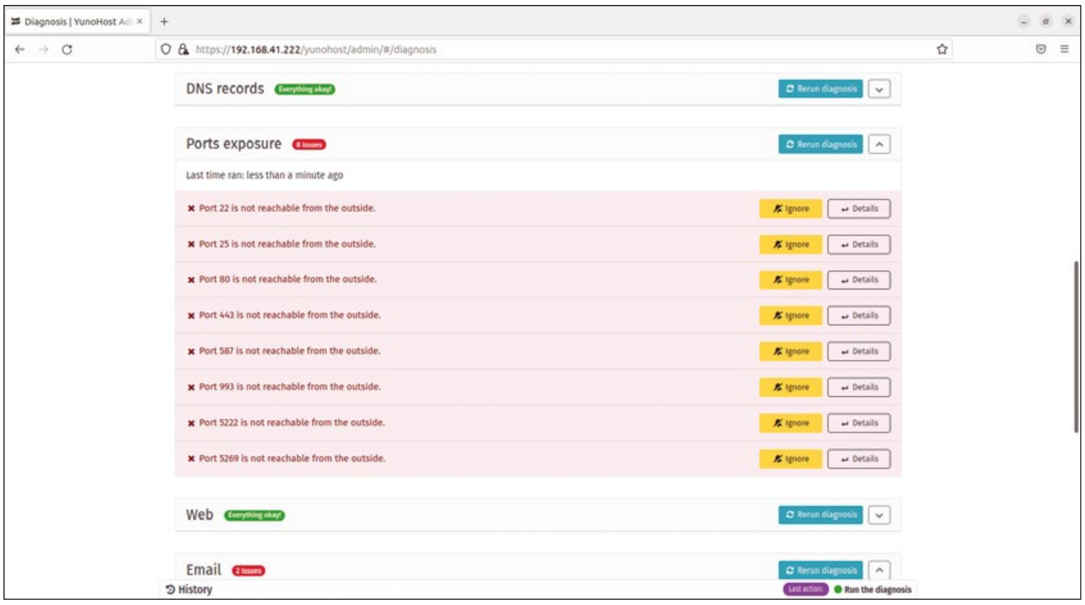
To do this, select *Users* in the main menu and *New User* in the dialog box that then appears. This will take you to the input mask for creating new users. Based on the domain, YunoHost also creates multiple email accounts. After completing the dialog box, press *Save*. Then open the settings for this user in the next window by clicking the small open arrow to the right of the user name.

In the *Mail forward* field of the user display, now specify an email address to which YunoHost will forward any notifications. To do this, press the blue button on the right to edit the account. Then select *Add a mail forward* from the dialog underneath and press *Save* to save the entry.

Diagnostics

When done, it is a good idea to run system diagnostics to find and fix potential problems. YunoHost runs system diagnostics twice a day by default. If problems are detected, the administrator is notified by email. You can start manual diagnostics by selecting *Diagnosis* in the main menu and then selecting *Start initial diagnosis*. The server now performs a comprehensive check of the system and after a few minutes displays the results in tabular form (Figure 3).

Diagnostic categories without abnormalities are marked by a green button following the respective group. Anomalies that do not affect the function of the system are tagged with a yellow warning. Red buttons indicate that there is a problem in this group that negatively affects the function. Each category with a warning or a detected problem has two additional buttons on the right side of the window: Clicking on *Details* will display details about the detected problem, while clicking on *Ignore* tells YunoHost to



DNS records Everything okay!

Ports exposure 8 issues

Last time ran: less than a minute ago

- Port 22 is not reachable from the outside.
- Port 25 is not reachable from the outside.
- Port 80 is not reachable from the outside.
- Port 443 is not reachable from the outside.
- Port 587 is not reachable from the outside.
- Port 993 is not reachable from the outside.
- Port 5222 is not reachable from the outside.
- Port 5269 is not reachable from the outside.

Web Everything okay!

Email 2 issues

Figure 3: Diagnostics help YunoHost detect errors in the configuration, which it then categorizes.

disregard the respective entry in subsequent diagnostics.

Numerous problems with red highlighting will tend to appear in the *Ports exposure* category in particular; this is because many routers block the ports required for access from the Internet. You can ignore ports 25, 80, and 443, which are especially important here, as long as you do not need access to the server from the outside. If required, you can change the status for this check by selecting *Ignore*.

Firewall and Logs

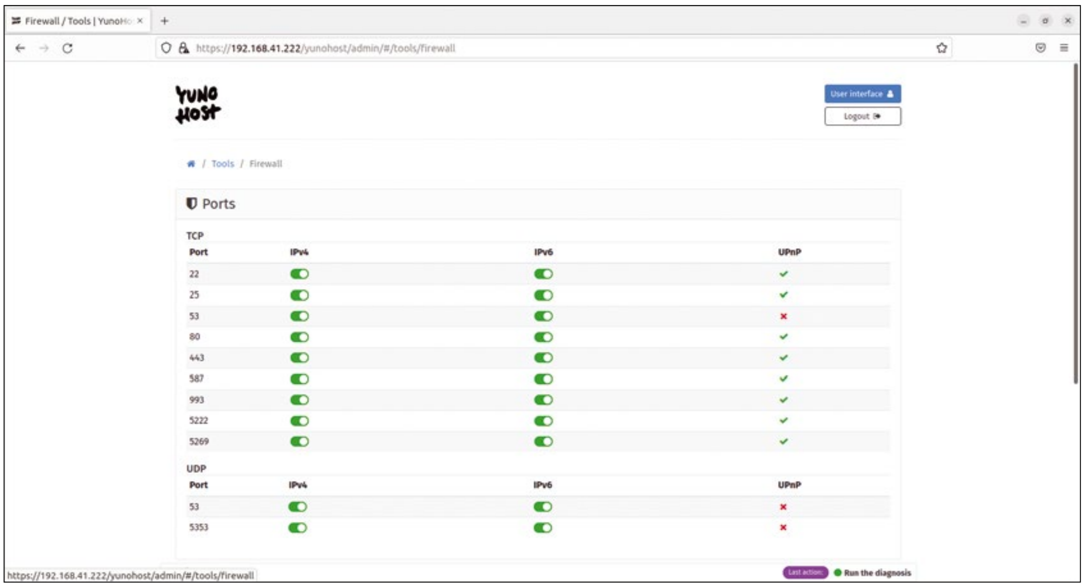
YunoHost enables an internal firewall during installation, which you can conveniently control using sliders. You can access the configuration in *Tools | Firewall*. The ports listed for IPv4 and IPv6 can be opened or blocked using a green slider (Figure 4). The *Operations* table further down the page also lets you define your own firewall rules. You can save the changes by pressing the button of the same name.

If problems occur, or applications do not work correctly, it is best to take a look at the system logs, which you can access via *Tools | Logs*. A click on one of the entries opens a sub-window that lists the detailed log entries, with color highlighting in part. This makes it far easier to keep track of extensive logs.

Services

To install the desired services, click *Applications* in the web GUI. In the next window, you will first see an empty interface with a note that no apps have

Figure 4: The integrated firewall lets you define the ports you want to allow or block.

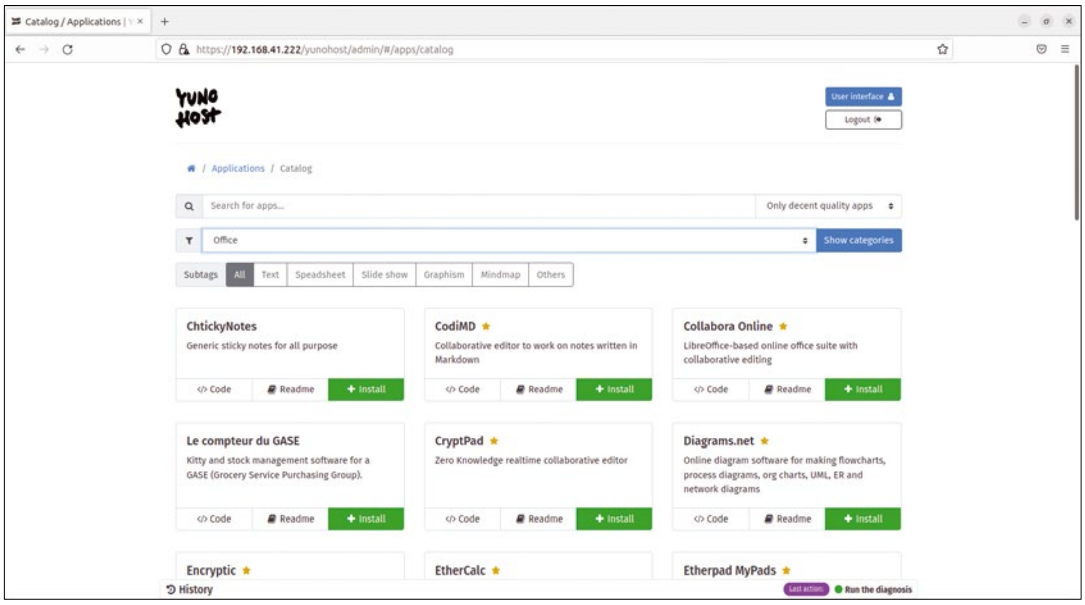

Firewall / Tools | YunoHo
https://192.168.41.222/yunohost/admin/#/tools/firewall
YUNO HOST
User interface
Logout
Tools / Firewall
Ports
Last action:
Run the diagnosis
https://192.168.41.222/yunohost/admin/#/tools/firewall

TCP

Port	IPv4	IPv6	UPnP
22	🟩	🟩	✓
25	🟩	🟩	✓
53	🟩	🟩	✗
80	🟩	🟩	✓
443	🟩	🟩	✓
587	🟩	🟩	✓
993	🟩	🟩	✓
5222	🟩	🟩	✓
5269	🟩	🟩	✓

UDP

Port	IPv4	IPv6	UPnP
53	🟩	🟩	✗
5353	🟩	🟩	✗



ChtickyNotes
Generic sticky notes for all purpose

CodiMD
Collaborative editor to work on notes written in Markdown

Collabora Online
LibreOffice-based online office suite with collaborative editing

Le compteur du GASE
Kitty and stock management software for a GASE (Grocery Service Purchasing Group).

CryptPad
Zero Knowledge realtime collaborative editor

Diagrams.net
Online diagram software for making flowcharts, process diagrams, org charts, UML, ER and network diagrams

Encryptic

EtherCalc

Etherpad MyPads

Figure 5: The web GUI displays installable applications in a tile view.

been set up as yet. After selecting *Install*, YunoHost shows you the catalog of installable applications sorted by groups. In it you will find all installable server services arranged in tiles. Above the catalog there are two search lines. The upper line is for free text input; you can enter the name of the application you are looking for here. A click on the magnifying glass symbol to the left shows you the matches.

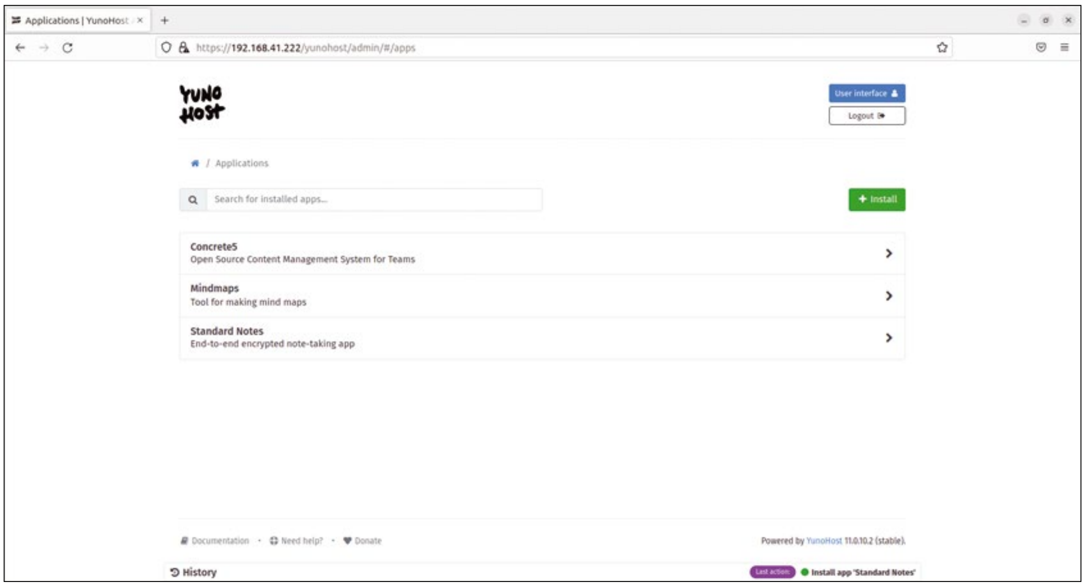
But if you want to select an application from a particular category of services, then click on one of the two triangles on the right side of the second search line. A list of categories appears, and you can select a group. After doing so, the tile view changes, showing only tiles from the selected application category (Figure 5). If additional subcategories exist, then the server displays them in a horizontal buttonbar below the search lines.

Once you have found the app you want, click *Install* in its tile bottom right. After doing so, YunoHost will display another page with information about the application and installation parameters. You can use the page to make changes, if necessary, name a path, and then start the installation by pressing *Install*.

During the setup, the routine also takes dependent packages into account; a progress bar shows the progress. When the process is complete, you will find the newly set up application in *Applications* (Figure 6).

Data Backup

If you have many services installed and numerous users, it only makes sense to back up the entire



Concrete5
Open Source Content Management System for Teams

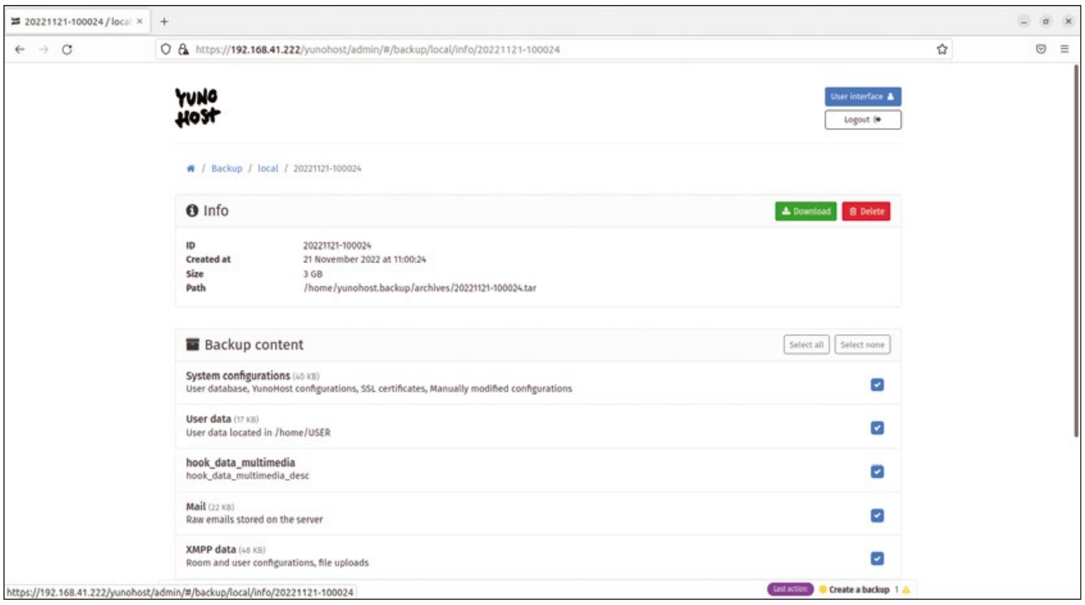
Mindmaps
Tool for making mind maps

Standard Notes
End-to-end encrypted note-taking app

Powered by YunoHost 11.0.10.2 (stable).

Install app 'Standard Notes'

Figure 6: The page lists new applications one below the other.



20221121-100024 / local
https://192.168.41.222/yunohost/admin/#/backup/local/info/20221121-100024

User interface
Logout

/ Backup / local / 20221121-100024

Info

Download Delete

ID	20221121-100024
Created at	21 November 2022 at 11:00:24
Size	3 GB
Path	/home/yunohost.backup/archives/20221121-100024.tar

Backup content

Select all Select none

- **System configurations** (40 KB) User database, YunoHost configurations, SSL certificates, Manually modified configurations ✓
- **User data** (17 KB) User data located in /home/USER ✓
- **hook_data_multimedia** hook_data_multimedia_desc ✓
- **Mail** (22 KB) Raw emails stored on the server ✓
- **XMPP data** (48 KB) Room and user configurations, file uploads ✓

https://192.168.41.222/yunohost/admin/#/backup/local/info/20221121-100024
Last action: Create a backup

Figure 7: Backups can also be conveniently created in YunoHost with just a few mouse clicks.

system on a regular basis. Clicking *Backup* will take you to a view that lists all the backup locations one below the other. Clicking the arrow to the right of it opens the actual backup window, which lists all backups.

Start a backup via *New backup* top right in the window. The following view lets you define what you want to save. YunoHost distinguishes between the system and the installed applications. You can check or uncheck the boxes to register or deregister the individual components for backup. For each group, all components can also be selected using *Select all* or excluded from the backup using *Select none*.

When done, press *Create a backup* bottom right. The software will now start the backup; a progress bar lets you keep track of its progress. After completing the backup, the module creates a table entry in the *Backup* window, where the name of the associated tar archive consists of the backup date. Clicking on the open arrow to the right of the entry displays information on the respective backup point. You can find information on the path, the backed up components, and the scope of the backup (Figure 7). Pressing *Download* saves the backup on the local computer.

To restore an archive, click on *Restore* bottom right. By checking or unchecking the boxes to the right of the individual components, you can include or exclude each of them from the data restore. After that, the actual recovery takes place.

User Portal

Once you have completed all the settings and finished installing the desired server services, you can log out from the YunoHost management interface. The users created by the administrator are then given access to the services via the user portal (Figure 8). Depending on the rights the administrator has granted for access to the different applications when creating each user in YunoHost, the portal will contain different tiles, each one representing a service.

Because the server works with LDAP/SSO-based authentication, you will not typically need to log in separately when calling the individual applications after logging in to YunoHost. You can switch between the individual tools, just like with conventional desktop applications.

On the other hand, the user's own configuration options in the user portal are limited. After clicking on the respective avatar in the portal, the user is taken to a dialog that only lets them edit their name, password, email address, and email forwarding address.

Conclusions

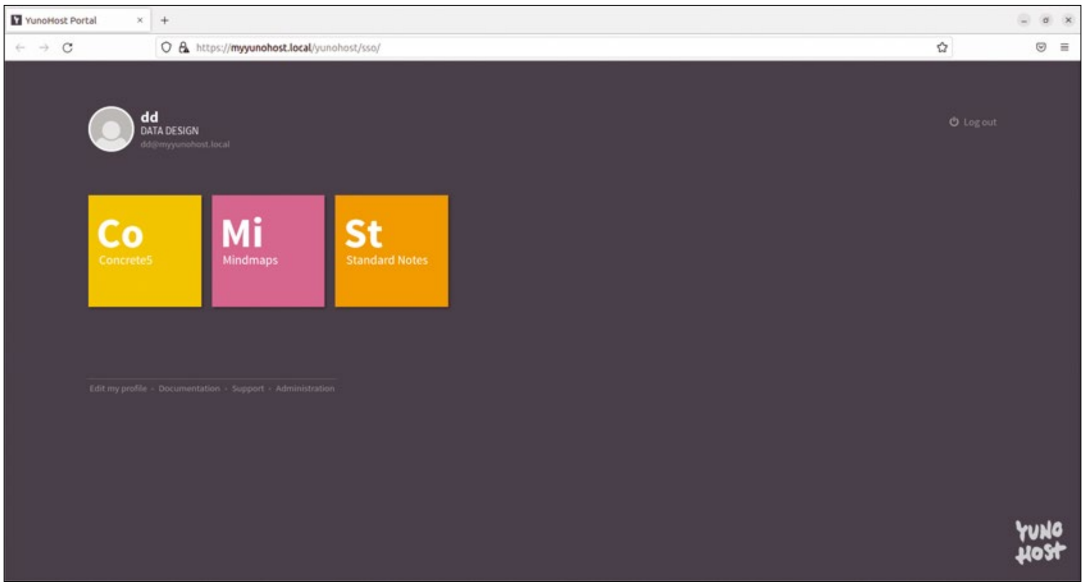
YunoHost helps even less experienced users build a working server environment without wasting time. The system makes creating and managing users an easy and intuitive process, while the large number of available applications leaves little to be desired. Thanks to single sign-on via the user portal, users can switch between applications without having to log on again. This makes YunoHost an excellent choice for smaller workgroups that want to use collaborative services. ■■■

Info

[1] YunoHost: *https://YunoHost.org*

[2] Minimum requirements: *https://YunoHost.org/de/install/hardware:regular*

Figure 8: Select the desired service by clicking on the corresponding tile.



Process management in a gamer outfit

System Under Control

Btop++ combines a high level of convenience with blistering speed in system monitoring and process management. BY FERDINAND THOMMES

System monitoring and process management are not necessarily popular, but they are indispensable tasks when you work with computers, whether on the desktop or on servers. Tools that help us do this need to show us as much information as possible in the space available. Terminal-based tools are particularly practical because they can be used both on desktops with a graphical interface and on remote computers.

At the command line, top ("table of processes") is the classic application in this category. First released in 1984 for BSD 4.1, it has also been available on Linux since 1991. Top displays a continuously updated list of all processes and also provides information about the processor load, memory consumption, the number of tasks, and other data (Figure 1). What the tool doesn't offer is clear-cut visual representation of the values.

In 2004, top was combined with the character-oriented ncurses user interface and has since been in residence on many a Linux machine in the form of htop. The display is clearer than with top alone, not least thanks to a configurable colored interface (Figure 2).

bashtop, bpytop, btop++

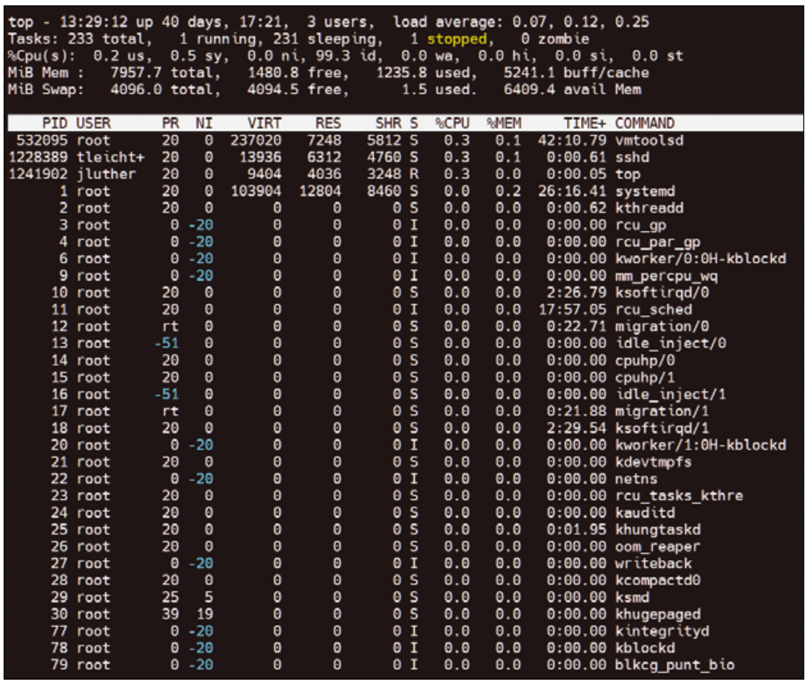
Developer Jakob P. Liljenberg, aka aristocratos, had a different graphical implementation in mind. He named it bashtop [1], and it is a Bash script (Figure 3), as you might guess from the name. To update the displayed values faster, Liljenberg then rewrote the tool in Python and published it as bpytop [2].

But even this incarnation was not yet sufficient for his needs, which prompted a rewrite in C++, dubbed btop++ [3]. Fans of the flexible process manager joked on the Reddit platform that the only thing missing now was a port to Rust, which could then culminate in a conversion to assembler. The name of the fictitious assembler tool is left to your imagination.

Btop++ takes us to the topic of this article. Btop++ is visually little different from its predecessors in Bash and Python, but it noticeably achieves the objective of accelerating value updates. On top of this, the C++ program has lower CPU and RAM requirements than its predecessors. Btop++ is not only available for Linux, but also for FreeBSD and macOS. A Windows port named btop4win is already done and will be released in the near future. Btop already works on the Windows Subsystem for Linux (WSL).

Version 1.2.8, released in June 2022, is already included in the package sources of many distributions, while others have btop++ 1.2.7. If the distribution you are using only offers an older version, you can easily install btop using the package manager or compile it yourself [4]. Version 1.2.9, which was the latest version when this issue went to press, offers a number of bug fixes, improvements in detail, and new themes, compared to its predecessors (Figure 4).

The requirements for the best possible display of btop are a terminal with 24-bit true color [5],



```
top - 13:29:12 up 40 days, 17:21,  3 users,  load average: 0.07, 0.12, 0.25
Tasks: 233 total,   1 running, 231 sleeping,   1 stopped,   0 zombie
%Cpu(s):  0.2 us,  0.5 sy,  0.0 ni, 99.3 id,  0.0 wa,  0.0 hi,  0.0 si,  0.0 st
MiB Mem :   7957.7 total,   1480.8 free,   1235.8 used,   5241.1 buff/cache
MiB Swap:   4096.0 total,   4094.5 free,      1.5 used.   6409.4 avail Mem

    PID USER      PR  NI    VIRT    RES    SHR S  %CPU  %MEM     TIME+ COMMAND
 532095 root      20   0  237020   7248   5812 S   0.3   0.1  42:10.79 vmtoolsd
1228389 tleicht+  20   0   13936   6312   4760 S   0.3   0.1   0:00.61 sshd
1241902 jluther   20   0    9404   4036   3248 R   0.3   0.0   0:00.05 top
      1 root      20   0  103904  12804   8460 S   0.0   0.2  26:16.41 systemd
      2 root      20   0       0      0      0 S   0.0   0.0   0:00.62 kthreadd
      3 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_gp
      4 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 rcu_par_gp
      6 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/0:0H-kblockd
      9 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 mm_percpu_wq
     10 root      20   0       0      0      0 S   0.0   0.0   2:26.79 ksoftirqd/0
     11 root      20   0       0      0      0 I   0.0   0.0  17:57.05 rcu_sched
     12 root      rt   0       0      0      0 S   0.0   0.0   0:22.71 migration/0
     13 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 idle_inject/0
     14 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/0
     15 root      20   0       0      0      0 S   0.0   0.0   0:00.00 cpuhp/1
     16 root     -51   0       0      0      0 S   0.0   0.0   0:00.00 idle_inject/1
     17 root      rt   0       0      0      0 S   0.0   0.0   0:21.88 migration/1
     18 root      20   0       0      0      0 S   0.0   0.0   2:29.54 ksoftirqd/1
     20 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kworker/1:0H-kblockd
     21 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kdevtmpfs
     22 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 netns
     23 root      20   0       0      0      0 S   0.0   0.0   0:00.00 rcu_tasks_kthre
     24 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kauditd
     25 root      20   0       0      0      0 S   0.0   0.0   0:01.95 khungtaskd
     26 root      20   0       0      0      0 S   0.0   0.0   0:00.00 oom_reaper
     27 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 writeback
     28 root      20   0       0      0      0 S   0.0   0.0   0:00.00 kcompactd0
     29 root      25   5       0      0      0 S   0.0   0.0   0:00.00 ksmd
     30 root      39  19       0      0      0 S   0.0   0.0   0:00.00 khugepaged
     77 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kintegrityd
     78 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 kblockd
     79 root       0 -20       0      0      0 I   0.0   0.0   0:00.00 blkcg_punt_bio
```

Figure 1: Top is the great grandmother of process management on Linux. Its resource usage is minimal, but the values are difficult to understand at a glance.

UTF-8 encoding, and a font that supports Braille. You also need a terminal with at least 80x24 lines. If the characters are displayed differently from the screenshots [6] on GitHub, you can find out why on the btop++ GitHub page under "Notice (Text rendering issues)" [7].

What btop Can Do

Btop displays real-time statistical values for the CPU, memory, storage media, network, and processes, updating them every two seconds by default. The values can be manipulated using the keyboard and mouse. All buttons with a character highlighted in red, in the default view, respond to mouse clicks, and you can scroll through the process table using the mouse wheel. Using the keyboard tends to be more practical all told and is definitely faster. Another editing mode that can be enabled in the settings is Vim mode. If you are a Vim user, you can use your muscle memory to navigate the process list with the *H*, *J*, *K*, and *L* keys.

The interface separates the different sections visually; each uses a different set of colors on a standard black background. The upper area is occupied by the CPU. On the right, btop++ shows the individual cores with the respective loads as percentages and the temperature for each core. Above this you can see the battery charging status, if the system has a battery. On the left, a real-time graph visualizes the workload for the last few minutes.

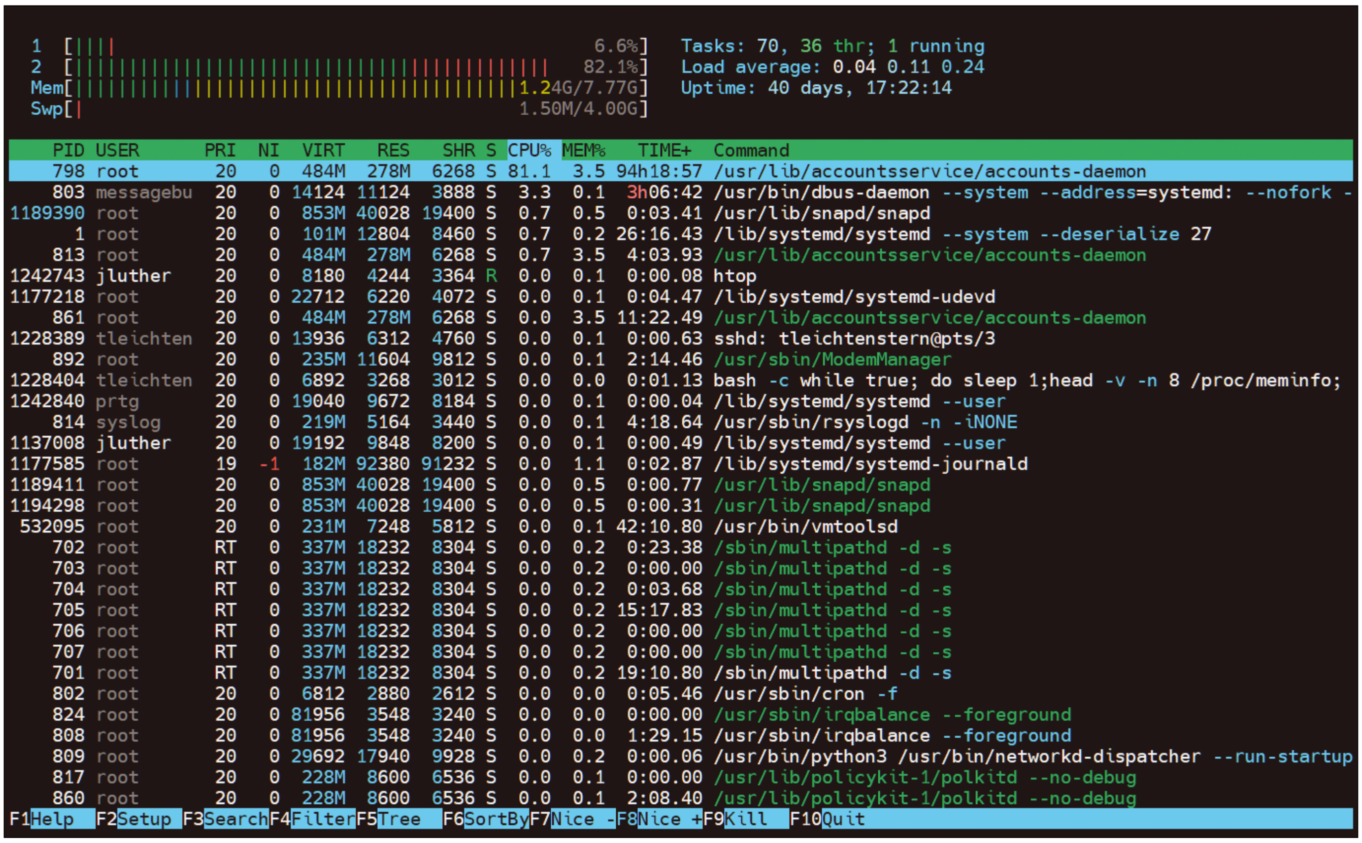
Below this, btop++ splits the view into two panes. Center left you can see the data for the RAM and storage media. The display for the RAM shows you the total memory, along with the values for in-use, cached, and free memory. To the right, small graphs indicate the storage media occupancy. Internal and external HDDs or SSDs appear here, along with the EFI boot partition if available. The area below this is for the network. Like for the CPU, btop++ shows the numeric values on the right, while a graph on the left visualizes the throughput for outgoing and incoming network traffic.

The right half of the display next to these two areas is taken up by the process list with a field at the top showing the details of the currently selected process. Below this there is a list of running processes and other information.

Well Prepared

After completing the install, the first launch, and finding your way around, you may want to press Esc or *M* to access the configuration. You will find the *Options*, *Help*, and *Quit* items in the menu that now opens. The settings in *Options* are distributed across the *general*, *cpu*, *mem*, *net*, and *proc* tabs.

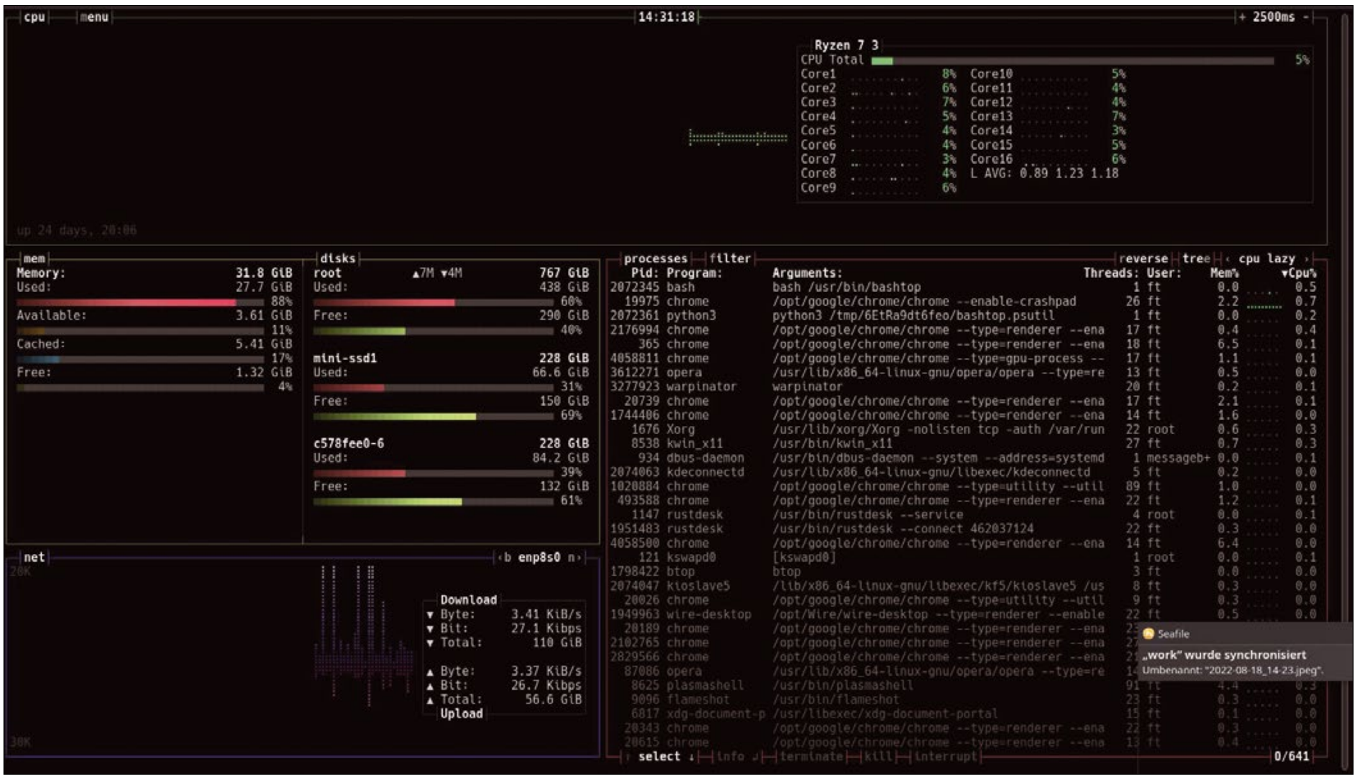
In the *general* tab, you can select your favorite theme from a list of about two dozen themes at the top. You can also define the background and how the display boxes are positioned on it. Three



```
  1  [||||                                                 6.6%]   Tasks: 70, 36 thr; 1 running
  2  [||||||||||||||||||||||||||||||||||||||||||||||||||  82.1%]   Load average: 0.04 0.11 0.24
  Mem[|||||||||||||||||||||||||||||||||||||||||||||||1.24G/7.77G]   Uptime: 40 days, 17:22:14
  Swp[|                                              1.50M/4.00G]

    PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
    798 root       20   0  484M  278M  6268 S 81.1  3.5 94h18:57 /usr/lib/accountsservice/accounts-daemon
    803 messagebu  20   0 14124 11124  3888 S  3.3  0.1  3h06:42 /usr/bin/dbus-daemon --system --address=systemd: --nofork -
1189390 root       20   0  853M 40028 19400 S  0.7  0.5  0:03.41 /usr/lib/snapd/snapd
      1 root       20   0  101M 12804  8460 S  0.7  0.2 26:16.43 /lib/systemd/systemd --system --deserialize 27
    813 root       20   0  484M  278M  6268 S  0.7  3.5  4:03.93 /usr/lib/accountsservice/accounts-daemon
1242743 jluther    20   0  8180  4244  3364 R  0.0  0.1  0:00.08 htop
1177218 root       20   0 22712  6220  4072 S  0.0  0.1  0:04.47 /lib/systemd/systemd-udevd
    861 root       20   0  484M  278M  6268 S  0.0  3.5 11:22.49 /usr/lib/accountsservice/accounts-daemon
1228389 tleichten  20   0 13936  6312  4760 S  0.0  0.1  0:00.63 sshd: tleichtenstern@pts/3
    892 root       20   0  235M 11604  9812 S  0.0  0.1  2:14.46 /usr/sbin/ModemManager
1228404 tleichten  20   0  6892  3268  3012 S  0.0  0.0  0:01.13 bash -c while true; do sleep 1;head -v -n 8 /proc/meminfo;
1242840 prtg       20   0 19040  9672  8184 S  0.0  0.1  0:00.04 /lib/systemd/systemd --user
    814 syslog     20   0  219M  5164  3440 S  0.0  0.1  4:18.64 /usr/sbin/rsyslogd -n -iNONE
1137008 jluther    20   0 19192  9848  8200 S  0.0  0.1  0:00.49 /lib/systemd/systemd --user
1177585 root       19  -1  182M 92380 91232 S  0.0  1.1  0:02.87 /lib/systemd/systemd-journald
1189411 root       20   0  853M 40028 19400 S  0.0  0.5  0:00.77 /usr/lib/snapd/snapd
1194298 root       20   0  853M 40028 19400 S  0.0  0.5  0:00.31 /usr/lib/snapd/snapd
 532095 root       20   0  231M  7248  5812 S  0.0  0.1 42:10.80 /usr/bin/vmtoolsd
    702 root       RT   0  337M 18232  8304 S  0.0  0.2  0:23.38 /sbin/multipathd -d -s
    703 root       RT   0  337M 18232  8304 S  0.0  0.2  0:00.00 /sbin/multipathd -d -s
    704 root       RT   0  337M 18232  8304 S  0.0  0.2  0:03.68 /sbin/multipathd -d -s
    705 root       RT   0  337M 18232  8304 S  0.0  0.2 15:17.83 /sbin/multipathd -d -s
    706 root       RT   0  337M 18232  8304 S  0.0  0.2  0:00.00 /sbin/multipathd -d -s
    707 root       RT   0  337M 18232  8304 S  0.0  0.2  0:00.00 /sbin/multipathd -d -s
    701 root       RT   0  337M 18232  8304 S  0.0  0.2 19:10.80 /sbin/multipathd -d -s
    802 root       20   0  6812  2880  2612 S  0.0  0.0  0:05.46 /usr/sbin/cron -f
    824 root       20   0 81956  3548  3240 S  0.0  0.0  0:00.00 /usr/sbin/irqbalance --foreground
    808 root       20   0 81956  3548  3240 S  0.0  0.0  1:29.15 /usr/sbin/irqbalance --foreground
    809 root       20   0 29692 17940  9928 S  0.0  0.2  0:00.06 /usr/bin/python3 /usr/bin/networkd-dispatcher --run-startup
    817 root       20   0  228M  8600  6536 S  0.0  0.1  0:00.00 /usr/lib/policykit-1/polkitd --no-debug
    860 root       20   0  228M  8600  6536 S  0.0  0.1  2:08.40 /usr/lib/policykit-1/polkitd --no-debug
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

Figure 2: Htop made the way information is visualized clearer (not least due to the colored display), but controlling the app is still less than intuitive.

options let you change the appearance of the graphs. If so desired, you can select a different display format for each box individually. You can also turn off the indicators for the battery here if you are working on a desktop. The other tabs let you manipulate how the components are positioned and the values displayed (Figure 5). If you prefer to handle the configuration in a text file, you

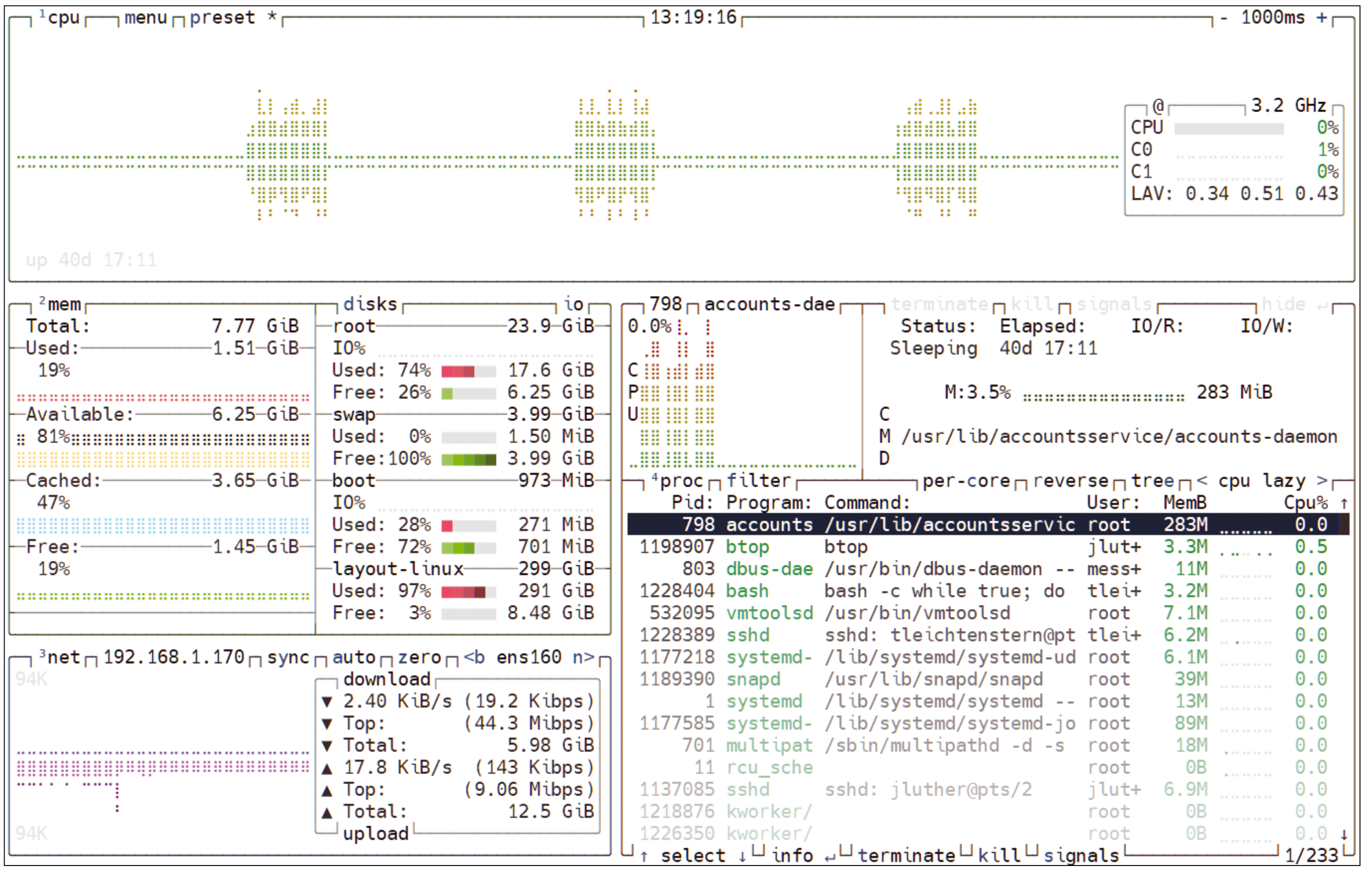


```
cpu  menu                              14:31:18                                   + 2500ms -
                                                 Ryzen 7 3
                                                 CPU Total                                5%
                                                 Core1        8%  Core10     5%
                                                 Core2        6%  Core11     4%
                                                 Core3        7%  Core12     4%
                                                 Core4        5%  Core13     7%
                                                 Core5        4%  Core14     3%
                                                 Core6        4%  Core15     5%
                                                 Core7        3%  Core16     6%
                                                 Core8        4%  L AVG: 0.89 1.23 1.18
                                                 Core9        6%
up 24 days, 20:06
mem                          disks
Memory:           31.8 GiB   root        ▲7M ▼4M    767 GiB
Used:             27.7 GiB   Used:                  438 GiB
                       88%                              60%
Available:        3.61 GiB   Free:                  290 GiB
                       11%                              40%
Cached:           5.41 GiB
                       17%   mini-ssd1              228 GiB
Free:             1.32 GiB   Used:                 66.6 GiB
                        4%                              31%
                             Free:                  150 GiB
                                                        69%

                             c578fee0-6             228 GiB
                             Used:                 84.2 GiB
                                                        39%
                             Free:                  132 GiB
                                                        61%
net                                         ‹b enp8s0 n›
20K
                                   Download
                                 ▼ Byte:       3.41 KiB/s
                                 ▼ Bit:        27.1 Kibps
                                 ▼ Total:         110 GiB

                                 ▲ Byte:       3.37 KiB/s
                                 ▲ Bit:        26.7 Kibps
                                 ▲ Total:        56.6 GiB
                                   Upload
30K
```

```
processes  filter                                              reverse  tree  ‹ cpu lazy ›
    Pid: Program:        Arguments:                              Threads: User:      Mem%   ▼Cpu%
2072345 bash            bash /usr/bin/bashtop                          1 ft         0.0     0.5
  19975 chrome          /opt/google/chrome/chrome --enable-crashpad   26 ft         2.2     0.7
2072361 python3         python3 /tmp/6EtRa9dt6feo/bashtop.psutil       1 ft         0.0     0.2
2176994 chrome          /opt/google/chrome/chrome --type=renderer --ena  17 ft      0.4     0.4
    365 chrome          /opt/google/chrome/chrome --type=renderer --ena  18 ft      6.5     0.1
4058811 chrome          /opt/google/chrome/chrome --type=gpu-process --  17 ft      1.1     0.1
3612271 opera           /usr/lib/x86_64-linux-gnu/opera/opera --type=re  13 ft      0.5     0.0
3277923 warpinator      warpinator                                     20 ft         0.2     0.1
  20739 chrome          /opt/google/chrome/chrome --type=renderer --ena  17 ft      2.1     0.1
1744406 chrome          /opt/google/chrome/chrome --type=renderer --ena  14 ft      1.6     0.0
   1676 Xorg            /usr/lib/xorg/Xorg -nolisten tcp -auth /var/run  22 root    0.6     0.3
   8538 kwin_x11        /usr/bin/kwin_x11                              27 ft         0.7     0.3
    934 dbus-daemon     /usr/bin/dbus-daemon --system --address=systemd   1 messageb+ 0.0   0.1
2074063 kdeconnectd     /usr/lib/x86_64-linux-gnu/libexec/kdeconnectd   5 ft         0.2     0.0
1020884 chrome          /opt/google/chrome/chrome --type=utility --util  89 ft      1.0     0.0
 493588 chrome          /opt/google/chrome/chrome --type=renderer --ena  22 ft      1.2     0.1
   1147 rustdesk        /usr/bin/rustdesk --service                     4 root       0.0     0.1
1951483 rustdesk        /usr/bin/rustdesk --connect 462037124          22 ft         0.3     0.0
4058500 chrome          /opt/google/chrome/chrome --type=renderer --ena  14 ft      6.4     0.0
    121 kswapd0         [kswapd0]                                       1 root       0.0     0.1
1798422 btop            btop                                            3 ft         0.0     0.0
2074047 kioslave5       /lib/x86_64-linux-gnu/libexec/kf5/kioslave5 /us   8 ft      0.3     0.0
  20026 chrome          /opt/google/chrome/chrome --type=utility --util   9 ft      0.3     0.0
1949963 wire-desktop    /opt/Wire/wire-desktop --type=renderer --enable  22 ft      0.5     0.0
  20189 chrome          /opt/google/chrome/chrome --type=renderer --ena  23
2102765 chrome          /opt/google/chrome/chrome --type=renderer --ena  27
2829566 chrome          /opt/google/chrome/chrome --type=renderer --ena  21
  87086 opera           /usr/lib/x86_64-linux-gnu/opera/opera --type=re  14
   8625 plasmashell     /usr/bin/plasmashell                           91 ft         4.4     0.3
   9096 flameshot       /usr/bin/flameshot                             23 ft         0.3     0.0
   6817 xdg-document-p  /usr/libexec/xdg-document-portal               15 ft         0.1     0.0
  20343 chrome          /opt/google/chrome/chrome --type=renderer --ena  22 ft      0.3     0.0
  20615 chrome          /opt/google/chrome/chrome --type=renderer --ena  13 ft      0.4     0.0
↑ select ↓  info ↵  terminate  kill  interrupt                                          0/641
```

Seafile
„work" wurde synchronisiert
Umbenannt: "2022-08-18_14-23.jpeg".

Figure 3: The bashtop developer chose a catchy visual presentation of the displayed values; this hardly changed in the successors, bpytop and btop++.



```
¹cpu  menu  preset *                     13:19:16                              - 1000ms +
                                                                         @        3.2 GHz
                                                                         CPU            0%
                                                                         C0             1%
                                                                         C1             0%
                                                                         LAV: 0.34 0.51 0.43
up 40d 17:11
²mem                      disks               io    798 accounts-dae     terminate kill signals          hide
 Total:        7.77 GiB   root           23.9 GiB   0.0%                   Status:   Elapsed:     IO/R:      IO/W:
 Used:         1.51 GiB   IO%                                              Sleeping  40d 17:11
  19%                     Used:  74%     17.6 GiB   C
                          Free:  26%     6.25 GiB   P                          M:3.5%              283 MiB
 Available:    6.25 GiB   swap           3.99 GiB   U                     C
  81%                     Used:   0%     1.50 MiB                         M /usr/lib/accountsservice/accounts-daemon
                          Free:100%      3.99 GiB                         D
 Cached:       3.65 GiB   boot            973 MiB   ⁴proc  filter                per-core  reverse  tree  < cpu lazy >
  47%                     IO%                           Pid: Program:  Command:                    User:   MemB        Cpu% ↑
                          Used:  28%      271 MiB        798 accounts  /usr/lib/accountsservic  root    283M         0.0
 Free:         1.45 GiB   Free:  72%      701 MiB    1198907 btop      btop                     jlut+   3.3M         0.5
  19%                     layout-linux    299 GiB        803 dbus-dae  /usr/bin/dbus-daemon --  mess+    11M         0.0
                          Used:  97%      291 GiB    1228404 bash      bash -c while true; do   tlei+   3.2M         0.0
                          Free:   3%     8.48 GiB     532095 vmtoolsd  /usr/bin/vmtoolsd        root    7.1M         0.0
³net  192.168.1.170  sync  auto  zero  <b ens160 n>  1228389 sshd      sshd: tleichtenstern@pt  tlei+   6.2M         0.0
94K                       download                   1177218 systemd-  /lib/systemd/systemd-ud  root    6.1M         0.0
                      ▼ 2.40 KiB/s  (19.2 Kibps)    1189390 snapd     /usr/lib/snapd/snapd     root     39M         0.0
                      ▼ Top:        (44.3 Mibps)          1 systemd   /lib/systemd/systemd --  root     13M         0.0
                      ▼ Total:          5.98 GiB    1177585 systemd-  /lib/systemd/systemd-jo  root     89M         0.0
                      ▲ 17.8 KiB/s  (143 Kibps)         701 multipat  /sbin/multipathd -d -s   root     18M         0.0
                      ▲ Top:        (9.06 Mibps)         11 rcu_sche                           root      0B         0.0
                      ▲ Total:          12.5 GiB    1137085 sshd      sshd: jluther@pts/2      jlut+   6.9M         0.0
                        upload                      1218876 kworker/                           root      0B         0.0
94K                                                 1226350 kworker/                           root      0B         0.0 ↓
                                                    ↑ select ↓ info ↵ terminate kill signals                        1/233
```

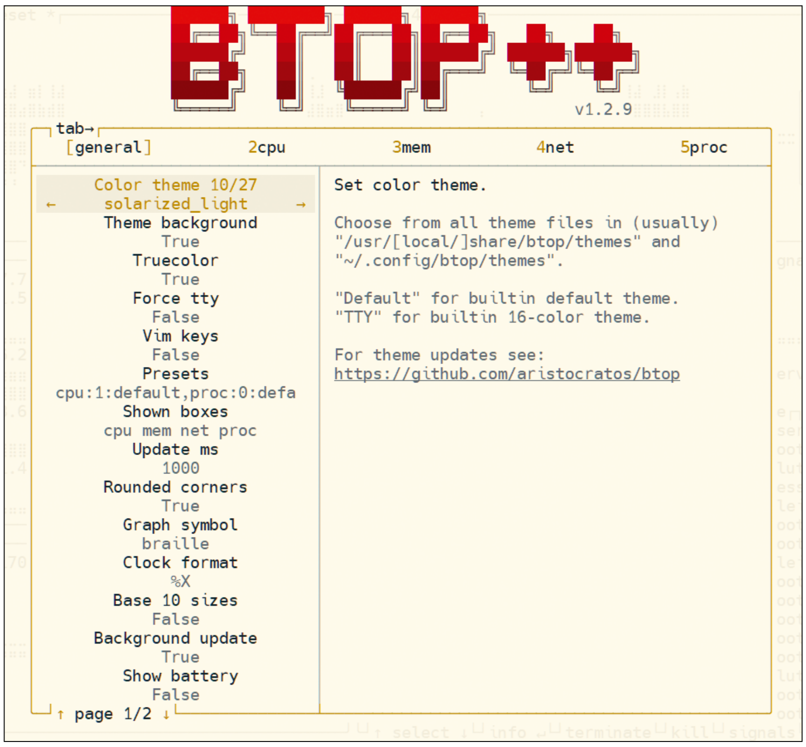
Figure 4: The btop standard interface uses a black background and darker hues, but you can easily change this by selecting a different theme.

will find it in `~/.config/btop/` along with the logfiles and the individual themes.

The Help menu lists the keyboard commands for controlling btop. The most important control options and short explanations are listed in Table 1.

Table 1: Keyboard Shortcuts

Key(s)	Function
Esc/*M*/Shift+M	Displays the main menu
F1/*H*/Shift+H	Displays the keyboard layout
F2/*O*/Shift+O	Displays the *Options* submenu
Ctrl+C/*Q*/Shift+Q	Quits the application
+/*A*/Shift+A	Extends the update interval by 100ms
-/*S*/Shift+S	Shortens the update interval by 100ms
Up arrow/Down arrow	Moves the cursor through the process list
Enter key	Shows details for the selected process
Home/End	Jumps to the beginning/end of the process list
E	Switches to the tree view of the process list
R	Rotates the order of the process list
F	Enter a string to filter processes
C	Removes active filters
1	Shows/hides the CPU box
2	Shows/hides the RAM and HDD box
3	Shows/hides the Network box
4	Shows/hides the process list



```
tab→
 [general]      2cpu      3mem      4net      5proc

    Color theme 10/27       Set color theme.
 ←  solarized_light   →
    Theme background        Choose from all theme files in (usually)
         True               "/usr/[local/]share/btop/themes" and
       Truecolor            "~/.config/btop/themes".
         True
       Force tty            "Default" for builtin default theme.
         False              "TTY" for builtin 16-color theme.
       Vim keys
         False              For theme updates see:
       Presets              https://github.com/aristocratos/btop
cpu:1:default,proc:0:defa
      Shown boxes
    cpu mem net proc
       Update ms
         1000
    Rounded corners
         True
     Graph symbol
        braille
     Clock format
          %X
    Base 10 sizes
         False
   Background update
         True
     Show battery
         False
 ↑ page 1/2 ↓
```

Figure 5: You can either configure btop in the graphical menu or edit it in the configuration file in your home directory directly.

Presets

Now let's take a look at how to use btop to find the information and details you need, as well as to filter and manipulate processes.

You can see the *cpu*, *menu*, and *preset* buttons on the left in the top line (Figure 6). We have already looked at the menu. The three presets are presets for the layout that influence the graphical display. You can define up to nine presets yourself in the *general* tab to customize the display in a granular way.

Pressing the *1* to *4* keys hides respective areas to free up space for the other displays. Pressing the key again shows the display box in question once more. The *1* key hides the CPU area, *2* removes the values for RAM and hard disks, and *3* causes the network data to disappear. If you only want a large-scale view of the process list, just press *1*, *2*, and *3*. Or, conversely, press *4* to remove the process area.

Editing Processes

To sort, filter, and manipulate the data displayed in btop, such as the process list, click on the small buttons in the headers or type the letter highlighted in the button. For example, to filter the running processes, you can click the *Filter* button just above the process list, or press *F*. Pressing *E* displays the processes as a tree list, while pressing *R* reverses the order. For a breakdown of the processes by CPU core, press *C*.

If you select a process and then press the Enter key, btop++ displays the details above the list. They include the status, runtime, RAM usage, owner, and parent process. You can terminate unruly processes here by pressing *T* or *terminate*. If that doesn't work, you send a kill signal to the process by pressing *K* or clicking on *kill*. (Signals are simple system messages to a running process, usually used to stop or terminate the process [8]. Typing the `kill -l` command on the console displays all the signals you can send to a process.) You can send other signals to a process by pressing *S* or clicking on *signals* (Figure 6).

All told, the way btop++ visualizes the process properties and uses signals is far easier and also safer compared with htop. (Other potential alternatives to btop++ include the Python-based Glances [9] and the Rust-based bottom [10].)

Conclusions

After a long history of development, btop++ has become by far the best tool for system

The Author

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

monitoring and process management on the console. The displays for the CPU, RAM, storage media, and network are clear and easily understandable. Manipulating processes is easier than ever. The simple keyboard controls become second nature after a few hours of use, and all of the functions can also be controlled using the mouse.

If the color scheme of the interface reminds you more of a computer game than serious work, you can change btop++'s look to display, say, gray hues, by choosing one of the various themes. The tool's configurability extends to every detail and shows how dedicated the developer is to his program. The readme on the GitHub project page [11] offers more than adequate documentation. ■■■

Info

[1] bashtop: *https://github.com/aristocratos/bashtop*

[2] bpytop: *https://github.com/aristocratos/bpytop*

[3] btop++: *https://github.com/aristocratos/btop*

[4] Installing btop++: *https://github.com/aristocratos/btop#installation*

[5] Terminal emulators: *https://github.com//termstandard/colors#terminal-emulators*

[6] btop++ screenshots: *https://github.com/aristocratos/btop#screenshots*

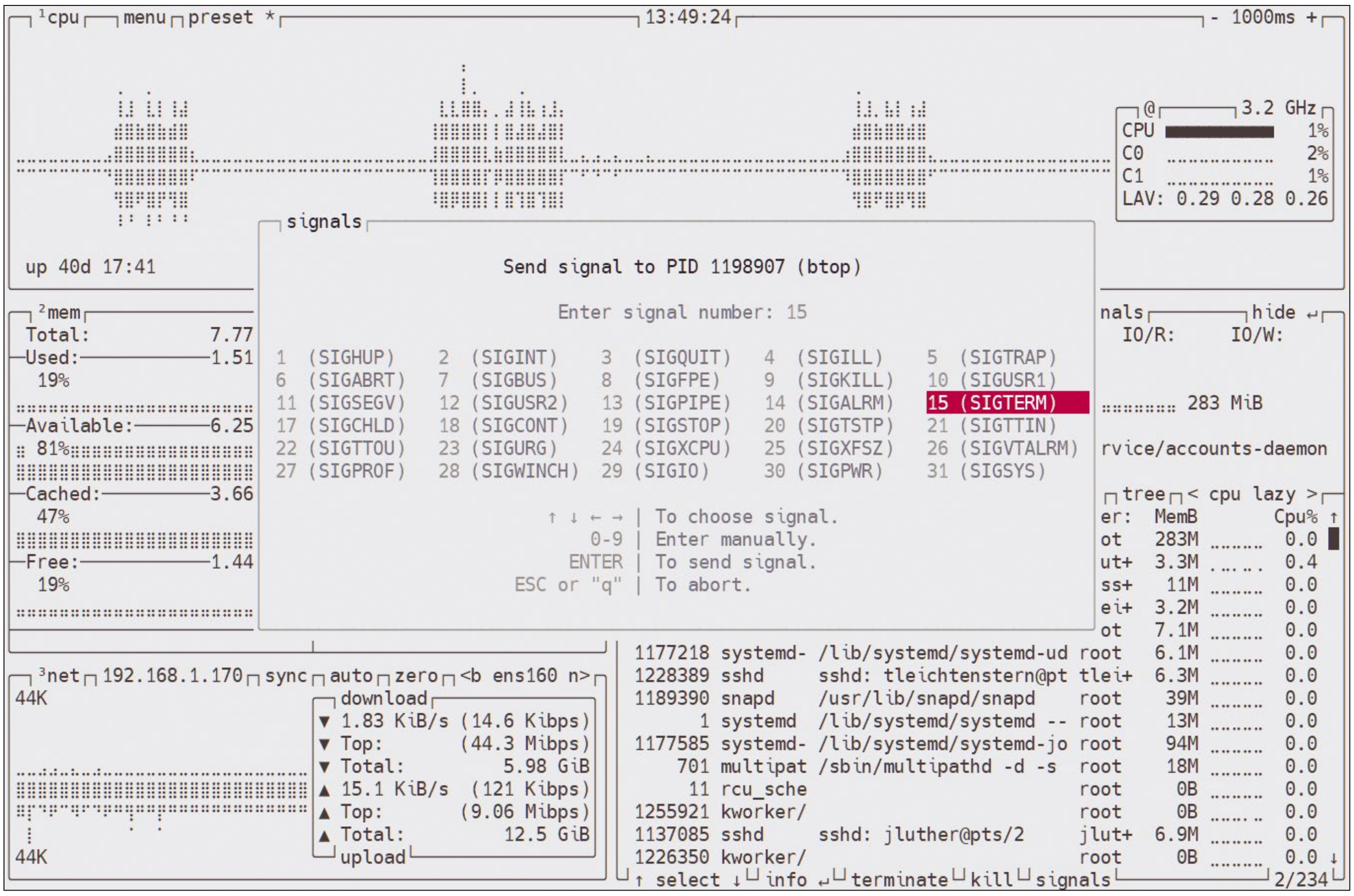
[7] "Notice (Text rendering issues)": *https://github.com/aristocratos/btop#notice-text-rendering-issues*

[8] SIG: *https://en.wikipedia.org/wiki/Signal_(IPC)*

[9] Glances: *https://github.com/nicolargo/glances*

[10] bottom: *https://github.com/ClementTsang/bottom*

[11] btop++ readme: *https://github.com/aristocratos/btop#readme*



```
 ¹cpu   menu  preset *                 13:49:24                                     - 1000ms +
                                                                               @          3.2 GHz
                                                                              CPU ▬▬▬▬▬▬▬▬▬▬   1%
                                                                              C0  ..........   2%
                                                                              C1  ..........   1%
                                                                              LAV: 0.29 0.28 0.26
                         signals
 up 40d 17:41                          Send signal to PID 1198907 (btop)

 ²mem                                   Enter signal number: 15                  nals          hide ↵
 Total:             7.77                                                          IO/R:      IO/W:
 Used:              1.51   1  (SIGHUP)    2  (SIGINT)    3  (SIGQUIT)   4  (SIGILL)    5  (SIGTRAP)
  19%                      6  (SIGABRT)   7  (SIGBUS)    8  (SIGFPE)    9  (SIGKILL)   10 (SIGUSR1)
                           11 (SIGSEGV)   12 (SIGUSR2)   13 (SIGPIPE)   14 (SIGALRM)   15 (SIGTERM)    283 MiB
 Available:         6.25   17 (SIGCHLD)   18 (SIGCONT)   19 (SIGSTOP)   20 (SIGTSTP)   21 (SIGTTIN)
  81%                      22 (SIGTTOU)   23 (SIGURG)    24 (SIGXCPU)   25 (SIGXFSZ)   26 (SIGVTALRM) rvice/accounts-daemon
                           27 (SIGPROF)   28 (SIGWINCH)  29 (SIGIO)     30 (SIGPWR)    31 (SIGSYS)
 Cached:            3.66                                                                          tree < cpu lazy >
  47%                                  ↑ ↓ ← → | To choose signal.                       er:  MemB         Cpu% ↑
                                           0-9 | Enter manually.                         ot   283M ........ 0.0
 Free:              1.44                 ENTER | To send signal.                         ut+  3.3M ........ 0.4
  19%                               ESC or "q" | To abort.                               ss+   11M ........ 0.0
                                                                                         ei+  3.2M ........ 0.0
                                                                                         ot   7.1M ........ 0.0
                                                    1177218 systemd- /lib/systemd/systemd-ud root   6.1M ........ 0.0
 ³net  192.168.1.170  sync  auto  zero  <b ens160 n> 1228389 sshd     sshd: tleichtenstern@pt tlei+ 6.3M ........ 0.0
 44K                      download                  1189390 snapd    /usr/lib/snapd/snapd     root    39M ........ 0.0
                          ▼ 1.83 KiB/s (14.6 Kibps)       1 systemd  /lib/systemd/systemd --  root    13M ........ 0.0
                          ▼ Top:       (44.3 Mibps) 1177585 systemd- /lib/systemd/systemd-jo root    94M ........ 0.0
                          ▼ Total:        5.98 GiB      701 multipat /sbin/multipathd -d -s   root    18M ........ 0.0
                          ▲ 15.1 KiB/s  (121 Kibps)      11 rcu_sche                          root     0B ........ 0.0
                          ▲ Top:       (9.06 Mibps) 1255921 kworker/                          root     0B ........ 0.0
                          ▲ Total:        12.5 GiB  1137085 sshd     sshd: jluther@pts/2      jlut+ 6.9M ........ 0.0
 44K                        upload                  1226350 kworker/                          root     0B ........ 0.0 ↓
                                                    ↑ select ↓  info ↵  terminate  kill  signals              2/234
```

Figure 6: Sending signals to a selected process: In addition to the PID and the process name, you will see the applicable signals and notes on handling the function.

FOSSPicks

Sparkling gems and new releases from the world of Free and Open Source Software

Graham was lucky enough to attend the first Ubuntu Summit in a decade, held in Prague in the Czech Republic. If he goes next year, he promises to run a session on making open source beer. BY GRAHAM MORRISON

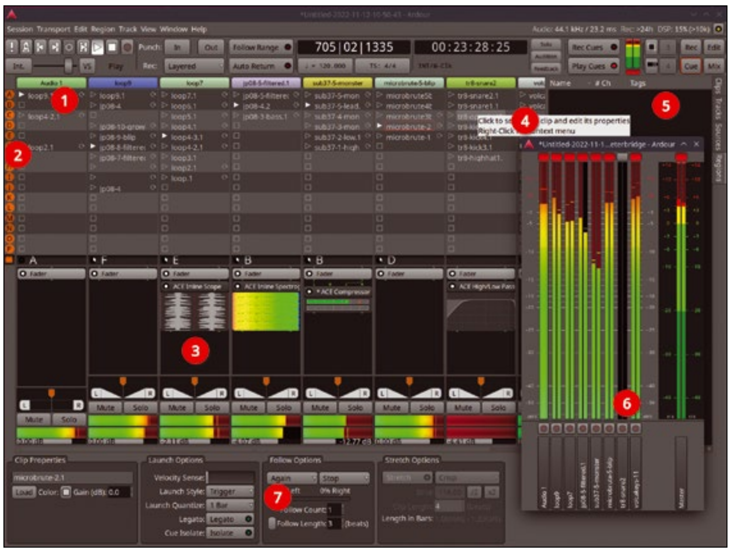
Audio workstation

Ardour

There has been a steady influx of new audio software for Linux over the past few years, and PipeWire looks set to make the Linux audio stack professionally performant and flexible. When it comes to digital audio, there's one application in particular that rules them all, and that's Ardour. Ardour is almost 20 years old now, and over that time it has become the most powerful open source digital audio workstation you can use. It's installed on macOS, Windows, and Linux desktops everywhere, from recording studios in Africa to universities in South America. Like Blender, it's powerful because it's flexible, and (also like Blender) because this power comes from its modularity, there is a learning curve. But if you're interested in recording more than one thing at a time, this shouldn't put you off. In Ardour, your audio hardware's inputs and outputs can be freely patched into and out of tracks and channels, routed through software effects, or configured to provide surround sound. Even without a single input and only headphones for output, you can create MIDI and audio tracks with virtual synthesizers or software drum machines, editing and mixing parts together across the horizontal timeline common to many DAWs. In many ways, this flexibility mimics a recording studio with its infinite audio sources and destinations to rewire, reconfigure, and experiment with. The recording studio metaphor has also helped make Ardour an excellent tool for professional recording engineers and student engineers in training, especially because it's a way of working that's already well established in software such as Pro Tools.

The release of Ableton Live 20 years ago, however, challenged that metaphor and changed what musicians and recording engineers want from their audio workstations. Rather than attempting to emulate a mixer and multitrack tape recording, Ableton instead focused on crafting music from "clips." Clips are blocks of audio, such as four bars of a drum loop, a repeating melody, or a series of chords, and Ableton lets you place these in a grid with variations of the same clip in the same column, or track. Clips that accompany each other are arranged in the rows across neighboring columns, and the entire row is known as a scene. With a single click, often performed live, the piece of music could seamlessly transition between scenes in a song. The magic sauce is that Ableton keeps everything in time, regardless of a clip's original tempo, pitch, or even groove. This simple idea and spreadsheet-like presentation turned Ableton Live into the DJ's tool of choice, ideal for remixes and improvisation.

Ableton Live still dominates the clip-launching DAWs, but it now has some serious competition from Bitwig Studio (which offers native Linux support, unlike Ableton) and even Apple's Logic, which has been augmented with its own clip arranger. Ardour can be added to that list, too, because clip launching is a huge addition for version 7, built atop a four-year project to change the timing algorithms used internally by Ardour. The clip launcher hides behind a new "Cue" mode and transforms the main UI into a place where you can drag and drop loops and samples from the new clip library or from outside the application entirely. Each drag and drop will create a new channel, with clips aligned vertically. These can be edited from a new panel to set the trigger and stretch modes, as well as the option to trigger a new clip from the current one. A horizontal strip of clips can be triggered as a scene, and scenes can even be triggered from the more traditional timeline view. It all works extremely well and feels like a natural extension to the recording, editing, and mixing views which are only a single click away, finally bringing Ableton into the 21st century.



1. **Clips:** Ardour now supports clips for creative loops, live performance, and recording. 2. **Scenes:** All clips in the same row can be played together, and everything stays in time. 3. **Effects:** The same channels and tracks are used in the background, including their insert effects. 4. **Context help:** Hover over an area to get some help on its function. 5. **Clip library:** Create a library of clips and regions to use across other projects. 6. **Monitoring:** Clips coexist with the standard Ardour functionality. 7. **Properties:** Set lengths, stretch options, priority, and clip triggers from the new properties pane.

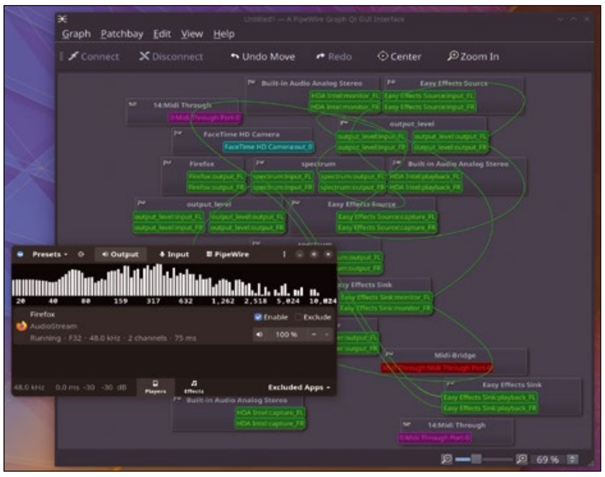
Project Website
https://ardour.org

Audio router

qpwgraph

PipeWire, the audio handling solution to replace PulseAudio, is now in enough mainstream distributions – including Pop!_OS, Fedora, and Ubuntu – to be considered the new standard. It already works incredibly well and offers much better audio performance and configurability than the old defaults. However, PipeWire lacks the desktop controls and smooth integration that come only from being the default for a decade, which is the time it seemed to take for PulseAudio to finally become integrated. It's also why most distributions will plug PipeWire directly into a PulseAudio layer to provide compatibility with those long-established control panels and applications, but this misses many of the advantages you get from using PipeWire directly. PipeWire is similar to JACK in both its low latency and its modularity, but you can't access these features without a third-party console, and that's exactly what qpwgraph is – a PipeWire graph Qt GUI interface with a terrible name.

If you've ever used Catia or QJackCtl to access JACK interconnectivity, you'll already be familiar with the view qpwgraph presents when first launched. Every detected input and output is shown as a node in a graph, with connections to show where audio data flows. If you launch Firefox and use it to play sound, for example, PipeWire and the graph automatically insert new nodes and adapt to show the new audio pathway. You can then drag and drop any connection to change the pathway, which is great if you want to run some effects or analysis, or route the audio both to an output and to a recording application.



The Qt interface is system agnostic, but if you prefer Gnome, Helvum does PipeWire interconnectivity for Gnome users.

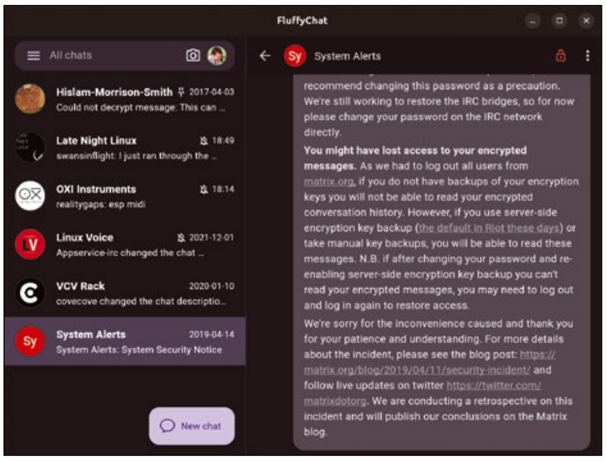
PipeWire with qpwgraph does all this natively, without any further configuration. If you're a Gnome user, the equally brilliant Helvum presents the same graph view interface within a GtkWindow that integrates perfectly with your desktop. Either way, both applications are a gateway to the wonderful world of PipeWire that don't require any system-wide compromises on the way.

Project Website
https://gitlab.freedesktop.org/rncbc/qpwgraph

Matrix client

FluffyChat

We'd be remiss to skip through this set of FOSSPicks without finding a chat client for the fediverse. With all the changes at Twitter HQ, there's been unprecedented demand for Mastodon clients, its instances, and even for running self-hosted servers. But Matrix too has seen a huge influx of new users, and the chat platform has seen a few new open source clients appear to go alongside the tried and tested Element. FluffyChat is one such client, and in its own words, it's "Open. Nonprofit. Cute." FluffyChat's cuteness comes from both its name and the way it's been designed. The "F" in its name is likely because FluffyChat has been written in the toolkit du jour, Flutter, and the client has been built with a clean user interface that's immediately familiar but also modern, fast, and flexible.
Flutter has been developed by Google, and as a result, it's very good at creating dynamic, minimal, and beautifully animated user interfaces that adapt easily across any form factor and desktop size. This is exactly what you get with FluffyChat on the desktop, but this also allows FluffyChat to coexist as a web application, as a cross-platform client, and as an Android application. They all feature the same responsive user interface that shows contacts and grouped chats on the left and the contents of any conversations on the right. You can change the accent color, switch between light and dark themes, and even create a backup of your conversations. All of this is done through Flutter's trademark smooth transitions and beautifully animated sliding panels. It makes Matrix a joy to use and is a great reminder that outside the contentious world of toots and tweets, there are stable and federated chat platforms that are quietly getting on with it.



Hislam-Morrison-Smith 2017-04-03
Could not decrypt message: This can ...

Late Night Linux 18:49
swansinflight: I just ran through the ...

OXI Instruments 18:14
realitygaps: esp midi

Linux Voice 2021-12-01
Appservice-irc changed the chat ...

VCV Rack 2020-01-10
covecove changed the chat descriptio...

System Alerts 2019-04-14
System Alerts: System Security Notice

recommend changing this password as a precaution. We're still working to restore the IRC bridges, so for now please change your password on the IRC network directly.

You might have lost access to your encrypted messages. As we had to log out all users from matrix.org, if you do not have backups of your encryption keys you will not be able to read your encrypted conversation history. However, if you use server-side encryption key backup (the default in Riot these days) or take manual key backups, you will be able to read these messages. N.B. if after changing your password and re-enabling server-side encryption key backup you can't read your encrypted messages, you may need to log out and log in again to restore access.

We're sorry for the inconvenience caused and thank you for your patience and understanding. For more details about the incident, please see the blog post: https://matrix.org/blog/2019/04/11/security-incident/ and follow live updates on twitter https://twitter.com/matrixdotorg. We are conducting a retrospective on this incident and will publish our conclusions on the Matrix blog.

Matrix is another open source federated chat platform that features groups and end-to-end encryption.

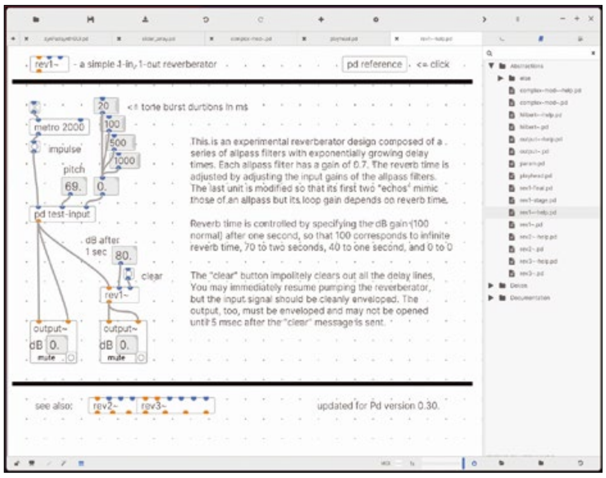
Project Website
https://fluffychat.im

Audio plugin

PlugData

The world of audio processing is incredibly complex. For that reason, many of us rely on other, cleverer people to develop the algorithms and software, and to package those up as plugins we can use with our favorite audio software. But there are also a few applications that can help with algorithm development without requiring a postgraduate degree in digital signal processing. And the best of these is Pure Data. Pure Data is a visual programming language that was designed for audio and control message processing. You join inputs and outputs together with a little scripting, but you can equally load and hack on hundreds of preexisting patches, algorithms, and documented examples. It's a powerful tool for generating your own sounds and effects, and it became the inspiration for the industry standard (non-open source and non-Linux) Max by Cycling '74. This in turn even led to the creation of Max for Live, an enviable integrated plugin and programming environment for Ableton Live on Windows and macOS.

Despite the availability of patches and modifications, Pure Data patches still need Pure Data running, and this solution doesn't work too well with other audio applications. Which is where PlugData can help. It's a version of Pure Data that wraps its functionality inside the JUCE plugin framework so your patches can be built and used directly as plugins. It will turn any audio plugin host into a modular and programmable audio processor, much like the aforementioned Max for Live, and is something we've never had access to in the Linux audio world. Lots of objects are included, and you can import your own patches and do everything you normally can in Pure Data. It works as you would expect, including MIDI and OSC data processing, making PlugData a fully fledged version of Pure Data where it's most useful – directly inside the audio path of a recording or producing system.



This is an experimental reverberator design composed of a series of allpass filters with exponentially growing delay times. Each allpass filter has a gain of 0.7. The reverb time is adjusted by adjusting the input gains of the allpass filters. The last unit is modified so that its first two "echos" mimic those of an allpass but its loop gain depends on reverb time.

Reverb time is controlled by specifying the dB gain (100 normal) after one second, so that 100 corresponds to infinite reverb time, 70 to two seconds, 40 to one second, and 0 to 0

The "clear" button impolitely clears out all the delay lines, You may immediately resume pumping the reverberator, but the input signal should be cleanly enveloped. The output, too, must be enveloped and may not be opened until 5 msec after the "clear" message is sent.

PlugData is the open source equivalent of Max for Live for whichever open source audio software you want to use.

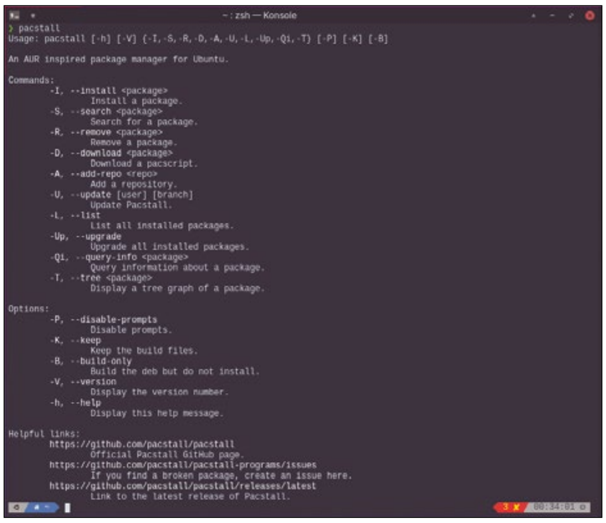
Project Website
https://github.com/timothyschoen/PlugData

Package manager

Pacstall

Arch Linux has transformed the Linux distribution landscape and has single handedly made minimal distributions cool again. It did this without oversimplifying the installation process or by making assumptions about what most users may need. In a world of easy installations and live distribution testing, Arch put the onus on the user, helped hugely by the wonderful Arch wiki. Arch Linux has since thrived and become default on Valve's Steam Deck and spawned a few derivatives. But one of the best things still about Arch is the Arch User Repository (AUR). This is a very Arch-like take on third-party packaging, again putting the responsibility on the user by asking them to check over the packages they install. Consequently, AUR packages are easy to audit and are incredibly diverse, and the AUR is often the first place to look for new software packages. Which is why the AUR is also central to writing these very pages, because it helps us test everything we write about.

Pacstall is a project that attempts to bring the convenience and diversity you find in the AUR to Ubuntu. It doesn't use the AUR itself, but it does aggregate lots of different package formats, including binary, Git, AppImage, and deb, and lets you build packages manually, just like `makepkg` with Arch. In this way, you always get the latest version available from Git or even build packages from forks or branches. The command line syntax will also feel familiar, with packages installed via `pacstall -I` followed by the package name. There aren't too many packages currently, but they do include complex open source and proprietary packages, including Android Studio, Bitwig, and Bitwarden. More important, however, the packages are created much like AUR packages, with a simple descriptive script that's easily accessible and easily reproducible locally. Not only does this mean you can be sure of what you're really getting, you can also create your own packages with very little difficulty.



```
❯ pacstall
Usage: pacstall [-h] [-V] {-I,-S,-R,-D,-A,-U,-L,-Up,-Qi,-T} [-P] [-K] [-B]

An AUR inspired package manager for Ubuntu.

Commands:
        -I, --install <package>
                Install a package.
        -S, --search <package>
                Search for a package.
        -R, --remove <package>
                Remove a package.
        -D, --download <package>
                Download a pacscript.
        -A, --add-repo <repo>
                Add a repository.
        -U, --update [user] [branch]
                Update Pacstall.
        -L, --list
                List all installed packages.
        -Up, --upgrade
                Upgrade all installed packages.
        -Qi, --query-info <package>
                Query information about a package.
        -T, --tree <package>
                Display a tree graph of a package.

Options:
        -P, --disable-prompts
                Disable prompts.
        -K, --keep
                Keep the build files.
        -B, --build-only
                Build the deb but do not install.
        -V, --version
                Display the version number.
        -h, --help
                Display this help message.

Helpful links:
        https://github.com/pacstall/pacstall
                Official Pacstall GitHub page.
        https://github.com/pacstall/pacstall-programs/issues
                If you find a broken package, create an issue here.
        https://github.com/pacstall/pacstall/releases/latest
                Link to the latest release of Pacstall.
                                                                 3 ✘  00:34:01
```

Pacstall is installed with a Bash script, but once installed, it can be used to add many more experimental packages to your system.

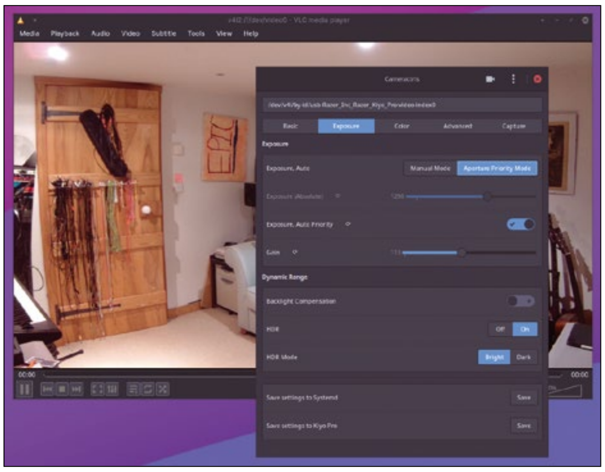
Project Website
https://github.com/pacstall/pacstall

Webcam controls

Cameractrls

For years, many of us were perfectly content to use our perfectly poor webcams for video meetings and idle video chat. This was probably because you could still physically meet up with the people, so your memories of them wouldn't be tainted by the crude color and noise of cheap sensors. Things have changed, and so too have our webcams. The latest models have much better sensors, much less noise, and are capable of resolutions such as 4K at 60 frames per second. Thanks to the V4L2 standard, the majority of these webcams will just work on Linux, but you can't typically configure them to the full extent of their capabilities because their manufacturers usually hide those options within their Windows-only applications.

Enter cameractrls to the rescue. Cameractrls exposes all the options supported by V4L2 and adds to these many of the secret options provided by Logitech, Brio, and Razor Kiyo webcams in their Windows drivers. These options can be configured either through the command line, just as you might with `v4l2-ctl`, or through the bundled GTK and Tk GUI front ends. These two are particularly good because they allow you to configure your webcam graphically while you're using the camera, so you can easily open these and tweak your settings while you're in a video meeting, for example. There's even an option to open the video stream so you can set up a configuration and save it before getting into a call. Most importantly, it is the only way to set those out-of-standard webcam specific features, such as pan and tilt, zoom, autofocus, and HDR settings, which might otherwise be out of reach without a Windows installation. You can then save those settings to the camera itself, or if your camera can't save everything, directly to systemd so they can be restored easily next time the camera is connected.



Cameractrls helps you configure your webcam and utilize its full capabilities.

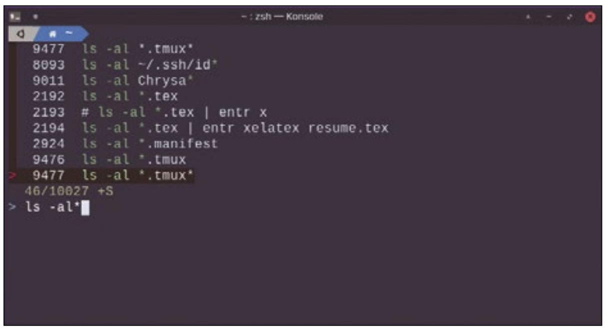
Project Website
https://github.com/soyersoyer/cameractrls

History improvements

hiSHtory

Despite sounding like it's been named after a drunken lecturer saying "history," hiSHtory is worthy of your attention. It updates the functionality of one of the most important aspects of our shell or terminal interactions, their history backlogs. This is something most of us access with up and down cursor keys, while a few of the elite press Ctrl+R to search through their previous commands. The problem with this default terminal history though, is that while the commands you typed are of course still remembered, their context is utterly lost. A command might have been typed into a remote session, even as a different user, in a different environment or location on your filesystem, and it will still be available through your history. The best this history can do is prod your memory to go through the same setup and configuration before running the command you wish to remember. And what happens when you move to a different machine entirely?

hiSHtory improves on this by remembering the context of commands. It can remember the directory you ran a command in, for example, or even whether the command was successful or not. This might not seem like a big deal, but if you've ever searched through your history only to find typos and badly formatted arguments, you'll know they all have equal precedence alongside the singular instance of the one successful command you're looking for. This is precisely where hiSHtory can help. It supplants the two previously mentioned history access commands and replaces them with additional functionality. In particular, you can search for commands containing a specific string, commands referencing a specific hostname, commands that return a certain exit code, or even services run after a certain date. The history is stored locally and can even be synced across devices with end-to-end encryption, so you always have access to your latest typos. It's brilliant and really should become the default history behavior.


```
  9477  ls -al *.tmux*
  8093  ls -al ~/.ssh/id*
  9011  ls -al Chrysa*
  2192  ls -al *.tex
  2193  # ls -al *.tex | entr x
  2194  ls -al *.tex | entr xelatex resume.tex
  2924  ls -al *.manifest
  9476  ls -al *.tmux
> 9477  ls -al *.tmux*
 46/10027 +S
> ls -al*
```
hiSHtory replaces your terminal's history functionality with a history that can be shared across multiple machines.

Project Website
https://github.com/ddworken/hishtory

2D/3D modeling

CadQuery Editor

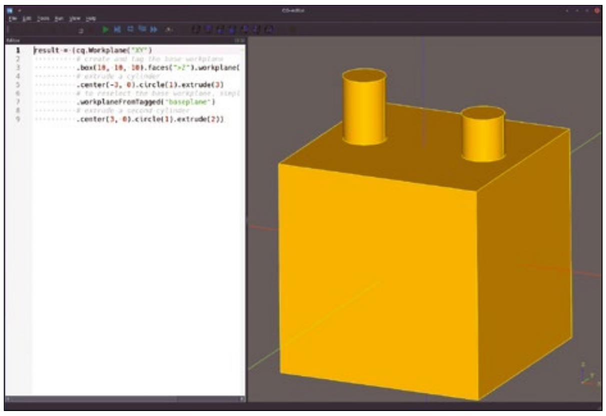
3D printing has brought many of us into contact with Computer Aided Design (CAD) when we might otherwise have remained ignorant, or perhaps have kept our 3D-modeling ambitions to Blender. As a result, Linux now has several excellent CAD tools that range from point-and-click 3D construction (Blender) and parametric modeling (FreeCAD), to programmable solid 3D modeling (OpenSCAD) and even Python libraries (CadQuery). Programmable 3D modeling is particularly well suited to 3D printing because it's often easier to describe what you want algorithmically than it is by dragging and dropping shapes into a canvas. You might start with a cube, for example, but then carve exact shapes out of it using intersection operators with a programmable set of instructions. This allows for objects to be created exactly in code, according to measurements and requirements, and also for parametric modification of a single element without affecting any other elements in the construction.

Blender and even FreeCAD offer their own programmable interfaces to achieve the same thing, but CadQuery has been developed solely around this ability. It's similar to OpenSCAD, which is also entirely driven through a programmable interface with an editor and 3D preview so you can see what you're creating. OpenSCAD uses its own functional programming language with a JavaScript syntax that's easy to learn but can't be used anywhere else. As CadQuery is a Python library, however, you obviously use Python, and that means Python from your favorite editor or development environment. All it takes to create a rectangle, for example, is:

```
cadquery.Workplane("front").↩
  box(2.0, 2.0, 0.5)
```

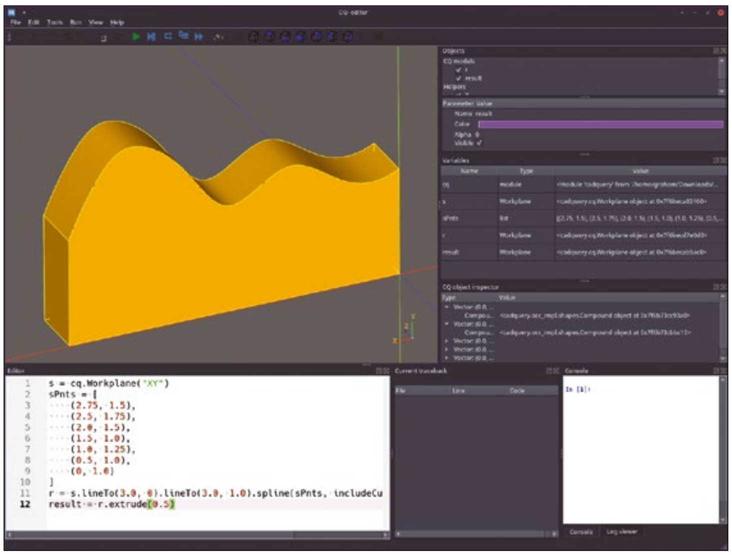
This is more portable and reusable than a custom code engine, but it does have the downside of not having a 3D preview to help with your coding. This is where CadQuery editor helps. It's a GUI application that brings together Python coding with the visual output, in much the same way OpenSCAD does. Alongside a 3D preview of the object you're building, there are panels to provide an interpreter, a traceback and log viewer, and a variable and object explorer. These work a little like debuggers for your own code, and they're useful when what you expect to happen doesn't happen. While modeling, the 3D view updates each time you run your changes, and you can step into or through individual lines in your code much like a debugger to see how an object is constructed, which is something you can't do in OpenSCAD. When you've finished modeling, objects can be exported from the editor as either an STL or STEP file without any further programming.

Writing in Python with CadQuery can make more sense than using OpenSCAD because the scripts you build are fully functional without the editor. This means they can be run independently when you've finished your modeling. One example of this is the brilliant CadQuery VS Code extension, which is almost equivalent to the editor when it comes to capabilities and 3D preview and is a great alternative if you're already using VS Code. Either way, CadQuery very much feels like the next generation of modeling for 3D printing and all kinds of Computer Aided Design.



```
result = (cq.Workplane("XY")
          # create and tag the base workplane
          .box(10, 10, 10).faces(">Z").workplane(
          # extrude a cylinder
          .center(-3, 0).circle(1).extrude(3)
          # to reselect the base workplane, simpl
          .workplaneFromTagged("baseplane")
          # extrude a second cylinder
          .center(3, 0).circle(1).extrude(2))
```

The CadQuery editor is really an IDE for Python: You can step through your scripts and watch how your models are constructed.



```
s = cq.Workplane("XY")
sPnts = [
    (2.75, 1.5),
    (2.5, 1.75),
    (2.0, 1.5),
    (1.5, 1.0),
    (1.0, 1.25),
    (0.5, 1.0),
    (0, 1.0)
]
r = s.lineTo(3.0, 0).lineTo(3.0, 1.0).spline(sPnts, includeCu
result = r.extrude(0.5)
```

CadQuery editor is a great way to visualize the results of your Python code and debug the output as you build 2D and 3D objects.

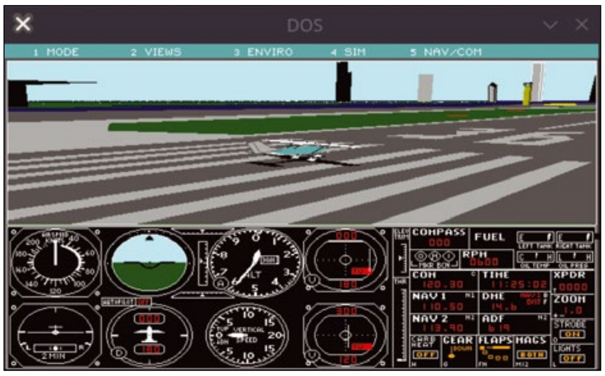
Project Website
https://github.com/CadQuery/CQ-editor

Flight simulator

FSHistory

Despite the CPU and RAM limitations of 1980s home computers, they were capable of remarkable things. And nowhere was this more evident than in the video games they ran. Games developers were able to push computer capabilities way beyond what was ever expected from a decade that started off with Breakout and ended with Ghouls 'n Ghosts. The most impressive games threw entire 3D worlds at the CRT, all from 8-bit integers stored in just 32,768 bytes of RAM. There was no better example of this than Elite, and then Microsoft Flight Simulator, first released in November 1982. Flight Simulator let you fly a variety of aircraft between various real-world airports in 3D with a moderate degree of instrument and flight mechanics realism, especially for the time. Sometimes you only got a few frames a second, and at other times the simulator became completely unresponsive, but it nevertheless felt like the future – something the 2020 edition of Microsoft Flight Simulator is only truly getting close to when you use a HOTAS and virtual reality headset (and a $2,000 graphics card).

Amazingly, especially considering the publisher, the decompiled version of Flight Simulator 4 is available under an MIT license, and FSHistory adds the supporting code and infrastructure necessary to make this build on modern hardware. In a humbling indication of progress, it takes just a few seconds to build FSHistory, and launching it takes an instant longer. The game now runs at hundreds of frames a second and is unlikely to ever become unresponsive, but it still feels like an authentic experience, requiring careful mouse and keyboard control to get anywhere. Because the code is identical, so too is the game, from the clunky menu system to the controls, graphics options, and windows configuration. You can't even resize the window because you couldn't in 1989 either. But if you can adapt your system to make it fit and you used to play Flight Simulator, it's the perfect nostalgia trip, virtually, from Oakland International runway number 28.



FSHistory is a version of Flight Simulator 4 from 1989 that will build on modern hardware. It is even hidden as an Easter egg in the 2020 version of Flight Simulator.

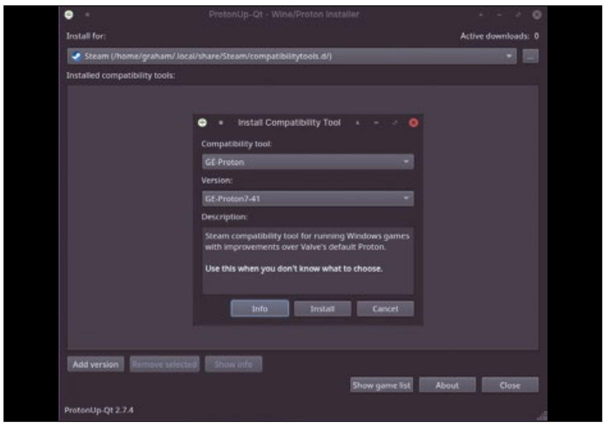
Project Website
https://github.com/s-macke/FSHistory

Compatibility tool

ProtonUp-Qt

While Valve is still preparing its Steam Deck Linux distribution for general release, there's a lot of Valve's gaming technology that can be used without waiting. Top of this list is Proton, the specially patched version of Wine that brings incredible Windows games performance to the native Linux desktop. Proton's performance has been central to Steam Deck's success, and you've been able to use it on your standard Linux desktop for a while. The problem is that there are lots of versions of Proton, and they all have their own quirks and performance impact on different games. This is the problem that ProtonUp solves, helping you download and manage multiple versions of Proton, especially Proton-GE – a community-built, bleeding-edge version of Proton – alongside lots of other compatibility tools. It helps you get the best possible performance from your games, regardless of whether you're using a Steam Deck or a desktop Linux PC. It's a little like Luxtorpeda, which we've looked at previously, only you need to configure the Proton versions manually for the games you have installed. But Luxtorpeda can even be installed through ProtonUp itself.

ProtonUp-Qt is a graphical equivalent to the ProtonUp command, making it easier to use and more accessible to Linux desktop users. Being built with Qt, it's ideal on Steam Deck in desktop mode, but it's equally useful on any Linux desktop. It works like a package manager, and you simply choose what you'd like to install from the available packages. Alongside Proton and Luxtorpeda, you can also install versions of Wine-GE and Boxtron, all of which enhance game compatibility and performance in Steam. You can then select whichever compatibility tool you want (or ask Luxtorpeda to do this for you) from the Properties page for your selected games in Steam, just as you do for default Proton builds. It'll even work with Lutris and Heroic Games Launcher, and it's a great way to squeeze the most out of your Linux gaming system.


ProtonUp-Qt - Wine/Proton Installer

Install for: Steam (/home/graham/.local/share/Steam/compatibilitytools.d/)

Active downloads: 0

Installed compatibility tools:

Install Compatibility Tool

Compatibility tool: GE-Proton

Version: GE-Proton7-41

Description: Steam compatibility tool for running Windows games with improvements over Valve's default Proton.

Use this when you don't know what to choose.

Easily install a cutting edge version of Proton and other gaming tools with ProtonUp-Qt.

Project Website
https://davidotek.github.io/protonup-qt/

Manage your private finances with Skrooge

Financial Manager

Skrooge is an accounting program that organizes and gives you a better overview of your private finances.

BY DANIEL TIBI

Anyone who wants to bring order to their personal finances and reveal potential savings will appreciate a good financial management program. Skrooge [1] was developed specifically for managing private finances. It helps you to record your income and expenses, arrange them by categories, and evaluate them graphically. The project name is based on the character Ebenezer Scrooge from Charles Dickens's novella *A Christmas Carol*.

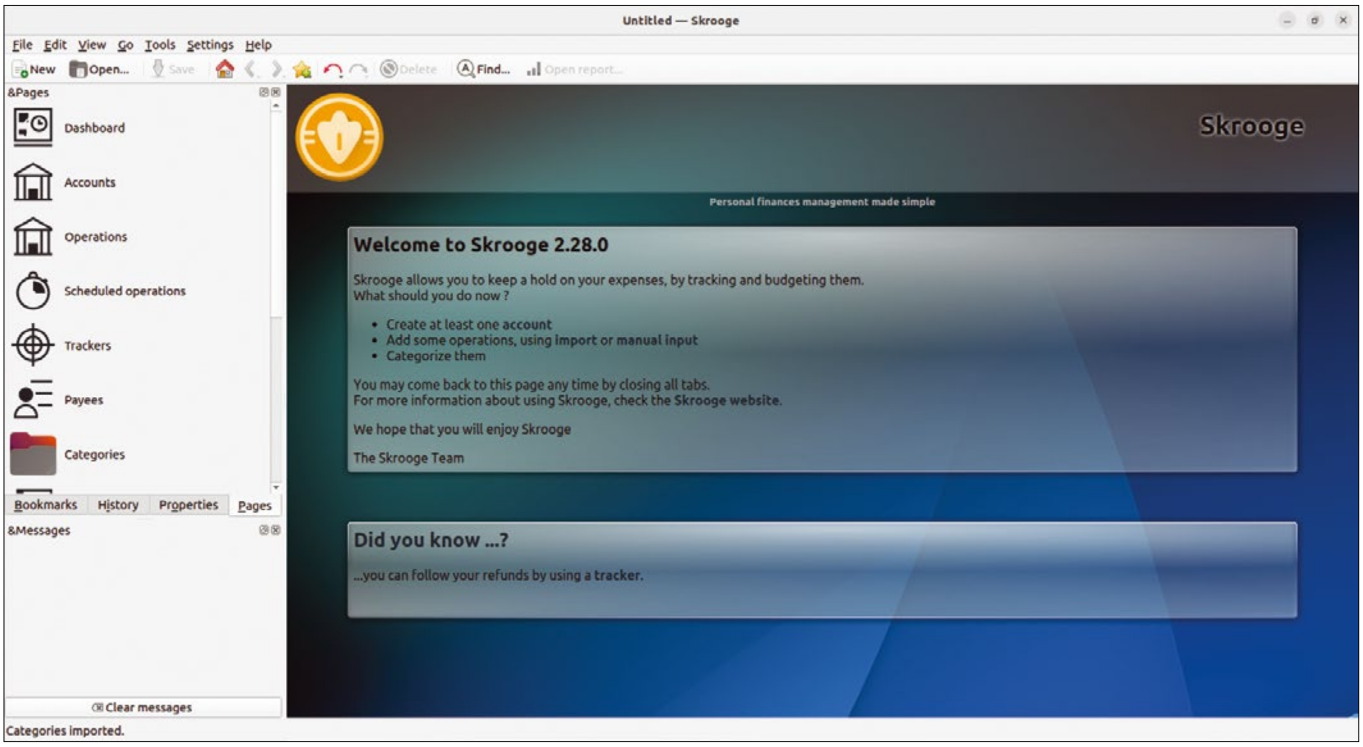
The current version of Skrooge is 2.28.0 from July 30, 2022. The program was originally developed for KDE, but it also runs on other desktop environments. Most distributions already have Skrooge in their package sources, so you can install it conveniently via the package manager. However, the repositories do not always give you the latest version. You can get this as an AppImage, Flatpak, Snap, or as a source code package from the project's download page [2].

After completing the install, you can launch Skrooge by clicking on the appropriate icon, in the shell by typing the `skrooge` command, or use the following command for Flatpak:

```
flatpak run org.kde.skrooge
```

Skrooge Setup

When first launched, the main window of Skrooge appears (Figure 1). In the left sidebar you can see an overview of the functions. Selecting one of them opens a new tab with the selected function on the right side of the main program window. Because the window uses tabs, several functions can be opened at the same time. To return to the initial window, just close all the tabs.



Welcome to Skrooge 2.28.0

Skrooge allows you to keep a hold on your expenses, by tracking and budgeting them.
What should you do now ?

- Create at least one account
- Add some operations, using import or manual input
- Categorize them

You may come back to this page any time by closing all tabs.
For more information about using Skrooge, check the Skrooge website.

We hope that you will enjoy Skrooge

The Skrooge Team

Did you know ...?

...you can follow your refunds by using a tracker.

Categories imported.

Figure 1: The main window in Skrooge is divided into two areas. You can see the functions on the left, which appear as new tabs on the right when you click on them.

If you are already using a different financial management tool and want to switch to Skrooge, it is worth trying to transfer the data via the menu item *File* | *Import* | *Import…*. Because the program supports the common formats QIF and OFX/QFX, and can also read GnuCash and KMyMoney files, switching usually does not cause any difficulties. If you get stuck during the setup, you can access the user manual via the *Help* | *Skrooge Handbook* menu.

Whether you are starting from scratch or importing data, the first thing to do is to save your financial management data in a file. To do this, select *File* | *Save as…* and specify the storage location. The software uses its own storage format with a file suffix of `.skg`. To protect the data from undesirable access, assign a password via *File* | *Change password…*. You can also change a previously assigned password in the same way.

Adding Accounts

First, add all your accounts in Skrooge. This means your checking account(s), credit card and savings accounts, and securities accounts. You can also create a separate account for cash. To do this, click on *Accounts* in the left sidebar. A new tab appears with an empty account overview. Enter your account data in the fields at the bottom (Figure 2).

The bold fields (*Bank*, *Account*, *Type*, and *Initial Balance*) are mandatory, while the others are for information purposes only and can be left blank. Enter your bank's name in the *Bank* field. If the bank exists in the Skrooge database, its logo will appear automatically. Use the *Account* field to assign a name for the account, such as *MyBank*. Choose the account type in the *Type* selection field; checking accounts belong in the *Current* category. Then enter your current account balance in the *Initial balance* field. If needed, you can change the currency type, for example, ¢ instead of $. This completes the data input for the mandatory values. You can optionally add your account number and the details of your bank branch. Proceed in the same way with each additional account; you don't need the bank details for a *Wallet* account type.

The *Minimum limit* and *Maximum limit* fields, each of which you can enable by checking the box to the left of the field, offer an interesting function. You are notified if the balance drops below or exceeds the defined thresholds. This means you can keep track of when you need to transfer money to your checking account to avoid going into the red, for example, or if there is a surplus balance in your checking account that you could invest.

After entering all the data, click *Add* bottom right to create the account. You can edit the data of an existing account by pressing *Edit*. After creating all the accounts you need, you can hide the fields in the lower part of the window by clicking *Edit* and then show them again later on in the same way, if needed.

Defining Categories

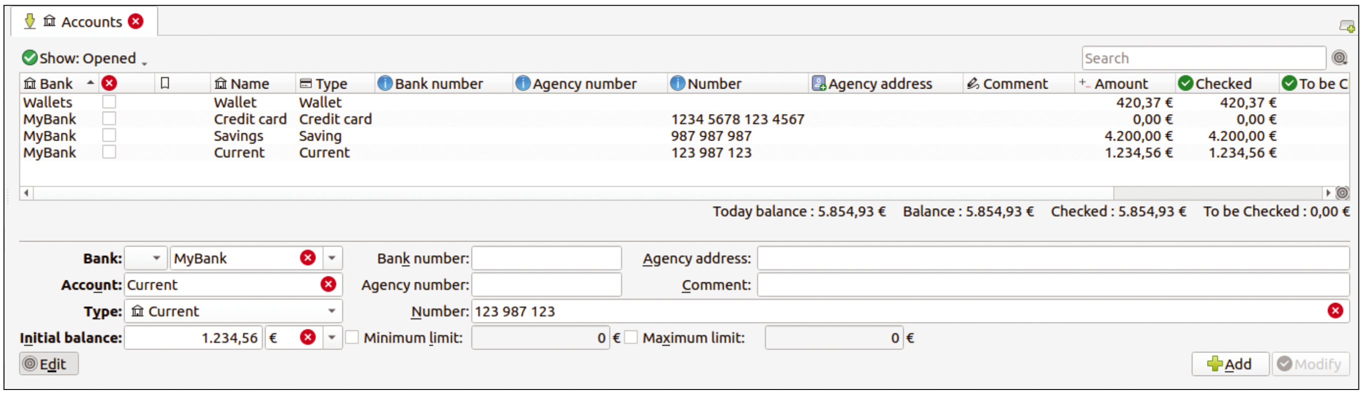
The second step is to create categories for your income and expenses. To do this, click *Categories* in the left sidebar. Skrooge does not come with predefined categories, so you will need to define them yourself. This will give you a category list that precisely meets your requirements.

You can create a new category by entering its name in the *Name* field in the lower area and then clicking the plus sign next to it (Figure 3). For more clarity, you can create subcategories as needed. You do not need to commit to a final list; the categories can be edited later at any time.

When you post entries, the total is listed after each category. This gives you a quick overview of how much you spent (and on what) or how much you earned.

Adding Payees

The final step is to add your sources of income and payees for your payments. These can include your employer, your landlord, and stores where you regularly shop. Payees can help you systematize your finances just like categories.



Bank			Name	Type	Bank number	Agency number	Number	Agency address	Comment	Amount	Checked	To be C
Wallets			Wallet	Wallet						420,37 €	420,37 €	
MyBank			Credit card	Credit card			1234 5678 123 4567			0,00 €	0,00 €	
MyBank			Savings	Saving			987 987 987			4.200,00 €	4.200,00 €	
MyBank			Current	Current			123 987 123			1.234,56 €	1.234,56 €	

Today balance : 5.854,93 € Balance : 5.854,93 € Checked : 5.854,93 € To be Checked : 0,00 €

Bank: MyBank
Account: Current
Type: Current
Initial balance: 1.234,56 €
Bank number:
Agency number:
Number: 123 987 123
Minimum limit: 0 €
Maximum limit: 0 €
Agency address:
Comment:

Figure 2: The account overview shows a summary of your accounts and the account balances. You can use the fields at the bottom to add new accounts.

This means that you can see what you spent money on or earned, and also where it went or came from. Creating payees is only worthwhile if you receive money from them or pay money to them on a regular basis. You do not necessarily need to specify a payee when posting an entry in your digital budget book. You do not need to create a payee for stores where you only rarely shop.

Click *Payees* in the left sidebar to open an overview of the payees. You add a new payee by entering the data in the area at the bottom. You need to at least enter a name, and optionally an address. If you always have income or expenses of the same type for a payee, such as food from the baker's shop, you can enter the details directly in the appropriate category. Clicking *Add* lets you add the payee to the list. If new payees are added later on, the list can be expanded at any time. As soon as you make entries, the corresponding amount appears to the right of each payee.

Income and Expenses

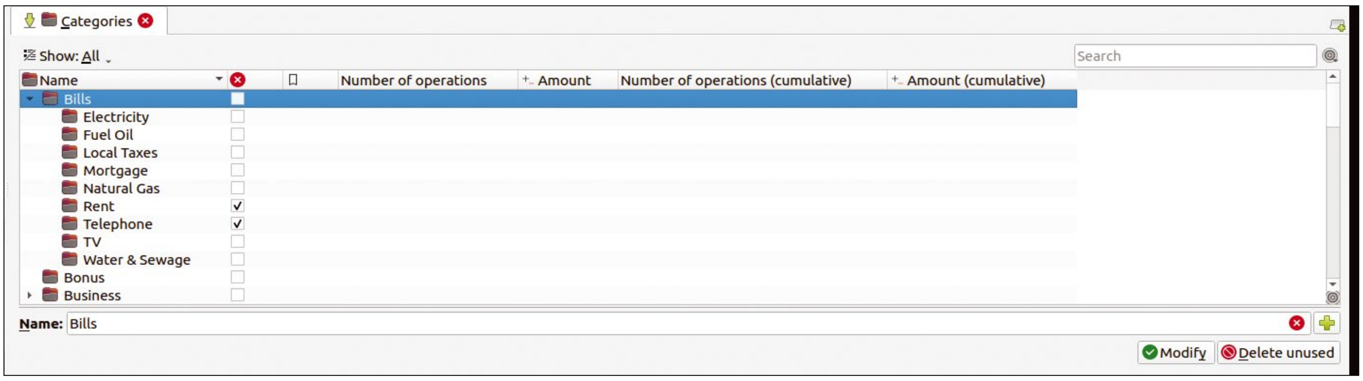
To enter revenues and expenses, click on *Operations* in the left sidebar. Following the usual approach, a new tab appears showing your income and expenses. You can enter new operations in the fields in the area at the bottom (Figure 4).

Just select the right account and enter the date and amount, remember to prefix expenses with a minus sign. You also select suitable payees and categories from the list boxes. Both help keep track of your finances, but neither is mandatory. If necessary, you can also add a comment. To transfer the operation to your budget book, press *Add* after entering all the data. You will now see a new line in the overview with the data you entered.

A tip on entering income: Record your gross salary, not the net amount. You need to state the income tax, social security contributions, and other deductions as expenses. This helps you keep track of your finances and have the correct amounts for a tax return at hand.

If an operation belongs to different categories – for example, if you buy groceries, personal care hygiene items, and magazines from the supermarket – press the *Split* button. You can then split the total amount across any number of categories.

Transferring money from one account to another is a special kind of a operation. For example, if you withdraw money from an ATM, you have an outpayment in your checking account and cash intake at the same time. To enter a operation of this type, press the *Transfer* button at the bottom of the page. This doesn't mean a bank transfer from your own current account to another, but a transfer between two of your own accounts. Enter the outpayment account in the *Account* field, for example, *Current account* in the case of a cash withdrawal from an ATM. The target account belongs in the *To account* field, such as *Wallet*.



Categories

Show: All

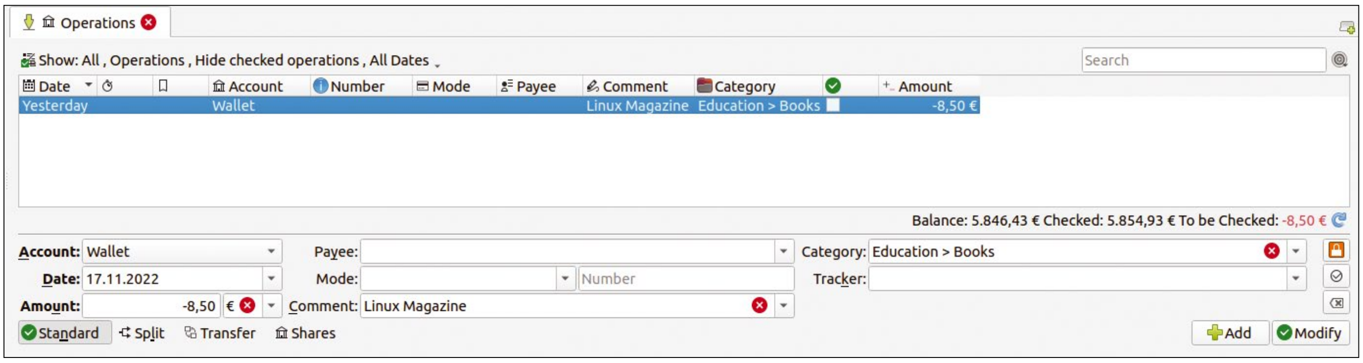
Search

Name			Number of operations	Amount	Number of operations (cumulative)	Amount (cumulative)
Bills	☐					
Electricity	☐					
Fuel Oil	☐					
Local Taxes	☐					
Mortgage	☐					
Natural Gas	☐					
Rent	✓					
Telephone	✓					
TV	☐					
Water & Sewage	☐					
Bonus	☐					
Business	☐					

Name: Bills

Modify | Delete unused

Figure 3: You can classify your income and expenses in categories.



Operations

Show: All , Operations , Hide checked operations , All Dates

Search

Date			Account	Number	Mode	Payee	Comment	Category		Amount
Yesterday			Wallet				Linux Magazine	Education > Books	☐	-8,50 €

Balance: 5.846,43 € Checked: 5.854,93 € To be Checked: -8,50 €

Account: Wallet
Payee:
Category: Education > Books
Date: 17.11.2022
Mode:
Number
Tracker:
Amount: -8,50 €
Comment: Linux Magazine

Standard | Split | Transfer | Shares | Add | Modify

Figure 4: You can then assign your income and expenses – known as operations in Skrooge – to categories and payees.

Skrooge opens a separate input screen for purchasing securities when you press the *Shares* button. You can split the total amount into the actual amount attributable to the shares, plus any fees and taxes.

Regular Operations

Certain incomes and expenses recur regularly, like salary or rent. You do not need to repeatedly enter these operations. For example, after entering a rental payment, you can right-click on the line with the operation and choose *Schedule* from the context menu. A new tab opens where you enter the details of the regular operation (Figure 5).

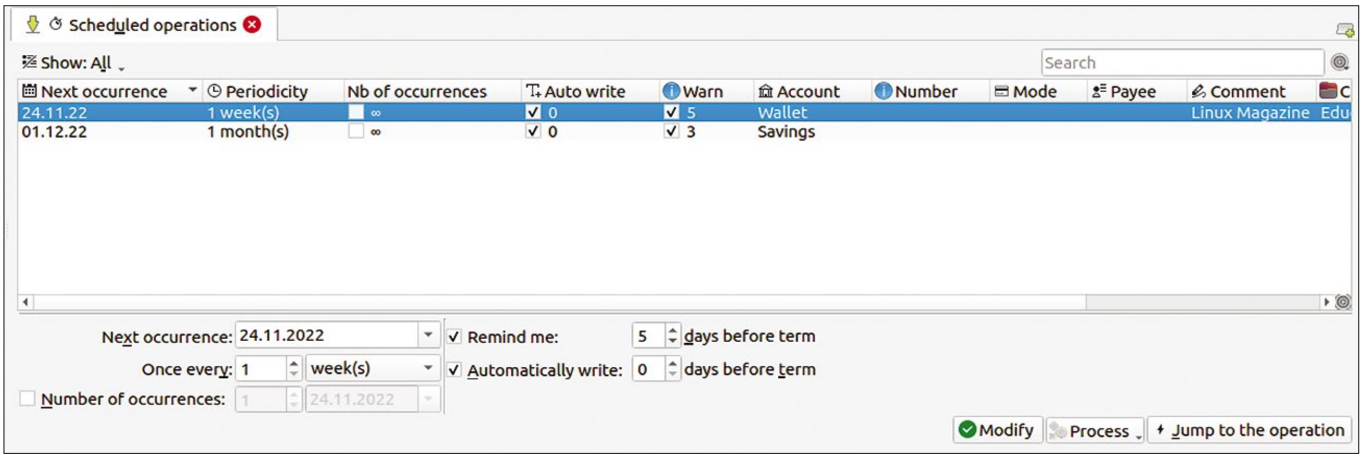
Start by entering the date when the operation is next due. Then enter the interval, once a month, for example. Optionally, add a closing date when Skrooge will post the operation for the last time. To do this, check the *Number of occurrences* box and specify the closing date. If the schedule is valid until further notice, do not check the box. You can also choose to be reminded of the operation any number of days in advance.

If you check the *Automatically write*, Skrooge will automatically post the item in your budget book. Then state how many days in advance you want the item to be posted. If you choose *0* here, Skrooge posts the item on the due date.

Synchronizing Operations

When you receive a bank statement, or if you count the money in your wallet, you need to check the account balance or cash amount against your records in Skrooge. To do this, click *Accounts* in the left sidebar and double-click the account you want to reconcile. A new tab opens with the entries for this account. Bottom right you will see a blue circular arrow. Clicking it switches to reconciliation mode. A *Final balance* field now appears in the bottom line; use it to transfer the account balance from your bank statement or the cash amount (Figure 6).

Skrooge now compares this amount with the account balance resulting from the operations in the budget book and computes the difference if the totals do not tally. If there is a difference in your current account, you may have forgotten to enter an operation. Differences with cash are more common, because you may not be given a receipt for minor cash outlay or might not enter


Scheduled operations

Show: All

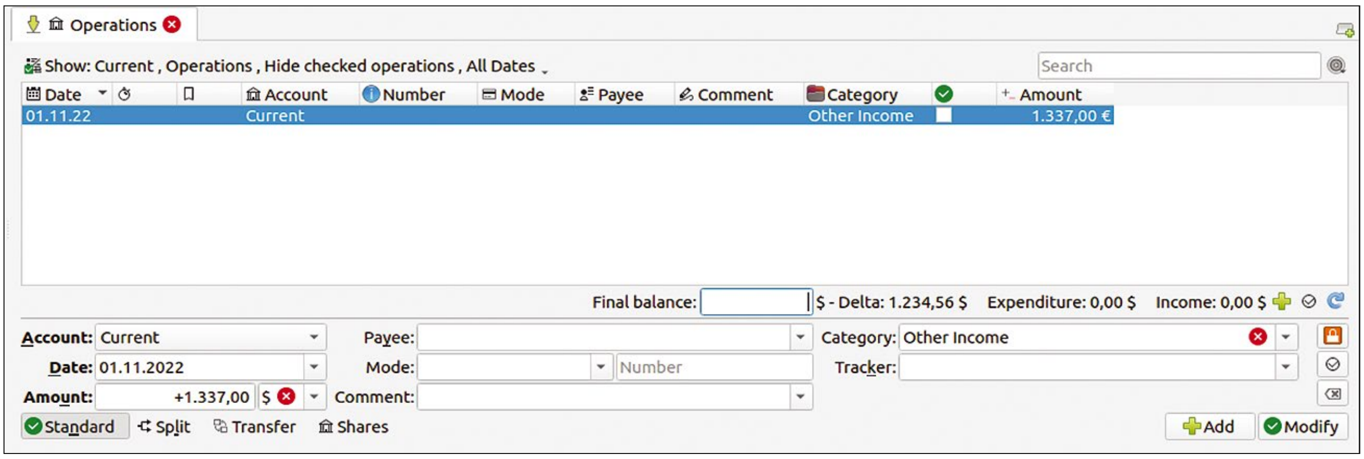
Search

Next occurrence	Periodicity	Nb of occurrences	Auto write	Warn	Account	Number	Mode	Payee	Comment	C
24.11.22	1 week(s)	☐ ∞	✓ 0	✓ 5	Wallet				Linux Magazine	Edu
01.12.22	1 month(s)	☐ ∞	✓ 0	✓ 3	Savings					

Next occurrence: 24.11.2022
Once every: 1 week(s)
☐ Number of occurrences: 1 24.11.2022
✓ Remind me: 5 days before term
✓ Automatically write: 0 days before term

Modify | Process | Jump to the operation

Figure 5: You only need to enter regular income and expenses once. To do this, you need to create a scheduled operation to tell Skrooge to post the item automatically.


Operations

Show: Current , Operations , Hide checked operations , All Dates

Search

Date			Account	Number	Mode	Payee	Comment	Category	✓	Amount
01.11.22			Current					Other Income	☐	1.337,00 €

Final balance: $ - Delta: 1.234,56 $ Expenditure: 0,00 $ Income: 0,00 $

Account: Current
Date: 01.11.2022
Amount: +1.337,00 $
Payee:
Mode:
Number
Comment:
Category: Other Income
Tracker:

Standard | Split | Transfer | Shares | Add | Modify

Figure 6: To keep your records up to date, you need to regularly reconcile your budget book with your bank statements and cash in hand.

the operation in your budget book. If there is a difference in the cash account, press the plus button on the right to enter the difference in your budget book. You can collectively post petty cash outlay in everyday life in this way.

Importing Bank Statements

Many banks let you download your bank statements as comma-separated values files (CSV, a file format that stores structured data in a text file and makes it easy to exchange data between different programs). Using the *File* | *Import* | *Import…* menu item you can load your account statements in this format and save yourself the trouble of having to enter the data individually.

Skrooge tries to map the fields in the budget book to the individual columns in the CSV file. If you see an error message, you will need to change the settings for importing CSV files in *Settings* | *Configure Skrooge* | *Import/Export* | *CSV* | *Edit Regular Expressions*. The required settings will vary depending on the structure of the CSV file. To discover more, open the file in a text editor or spreadsheet. In the first line you will see the names of the individual columns. Then transfer the names of the columns to the program so it can assign a value from the CSV file to each field.

Alternatively, you can load your bank statement data via the AqBanking [3] program. If you do not already have AqBanking, you will need to install it on your system up front. Your bank will give you the data required for the setup. After that, you can access AqBanking in Skrooge via the *File* | *Import* | *Import with backends…* menu.

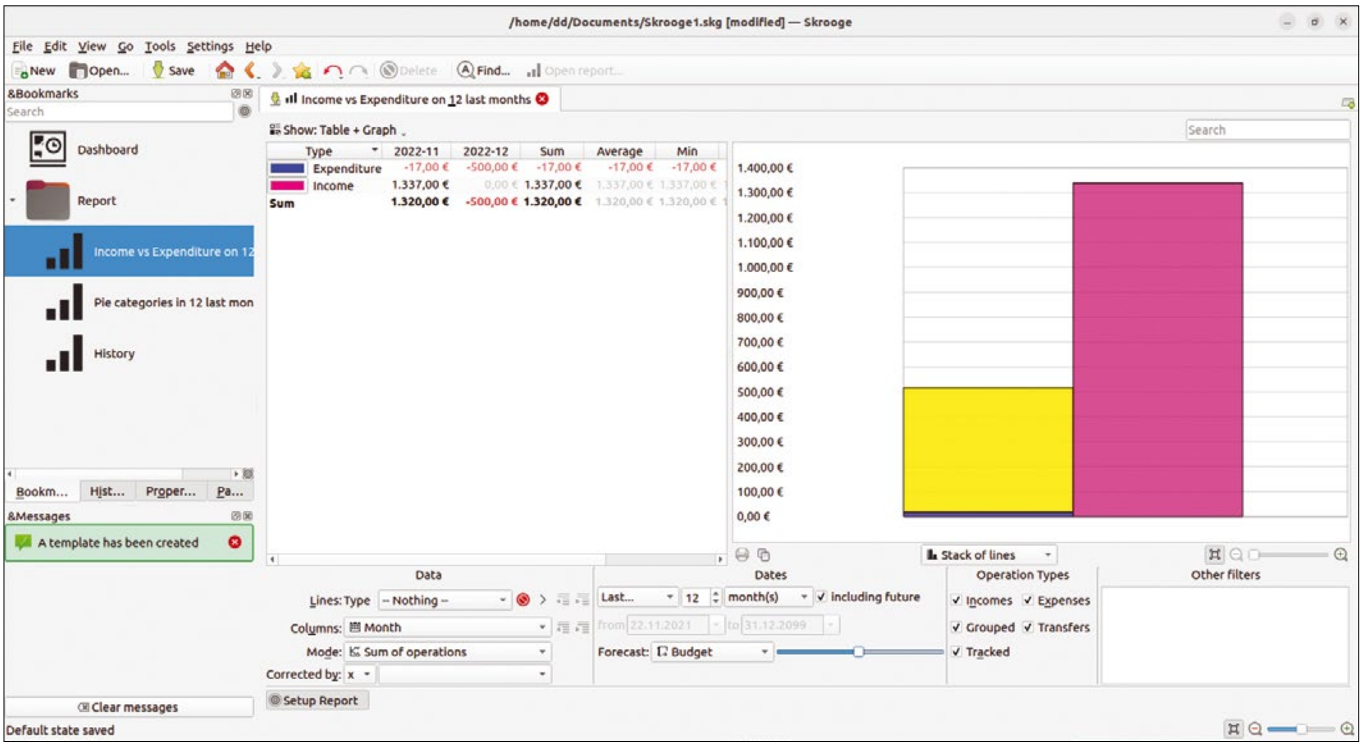
Evaluating Finances

Pressing *Monthly report* in the left sidebar tells Skrooge to open an overview of your income and expenses for the past month or quarter (Figure 7). This helps you keep track of your income and expenses and any changes compared to the previous month or quarter. You can see how much you have left at the end of the month and how much you can afford to set aside. You may also notice some potential for savings. Experience shows that an overview like this is not meaningful until you have collected data for at least three months or possibly even a year, when all annually recurring operations such as tax refunds or additional expenses have been posted.

Budgets and Interest

Once you have recorded and analyzed your finances for some time, you will want to create a budget. To do this, click *Budget* in the left sidebar. You can assign a specific monthly or annual budget to the categories. For example, you define the maximum amount of money you want to spend on eating out every month. You can use the category to keep track of how much of this budget you have already used up in the current month.

You can create your budget yourself or just let the software do it for you. If you choose the latter, press the *Automatic* icon at the bottom of the



Type	2022-11	2022-12	Sum	Average	Min
🟦 Expenditure	-17,00 €	-500,00 €	-17,00 €	-17,00 €	-17,00 €
🟪 Income	1.337,00 €	0,00 €	1.337,00 €	1.337,00 €	1.337,00 €
Sum	1.320,00 €	-500,00 €	1.320,00 €	1.320,00 €	1.320,00 €

Figure 7: Skrooge compiles your income and expenses in a single overview, giving you a better way of keeping track of everything.

Budget window. Skrooge analyzes the operation data of a given year as the basis for automatically creating a budget. For the budget calculations to be realistic, data for a complete year should be available.

If you have taken advantage of potential savings and put money aside, it may be worthwhile investing your excess capital. For an overview of the potential interest returns, press *Simulations* in the left sidebar to open an interest calculator. At the bottom of the window, select *Amortization Table* instead of *Interest* to display the monthly interest and installments for a loan.

Conclusions

The developers have designed Skrooge as a digital budget book. This clearly distinguishes the program from financial managers such as GnuCash [4] or KMyMoney [5]. The program focuses on aspects that are essential for private finances. A systematic approach with categories and payees gives you an overview of what you spend money on and where your earnings come from. The monthly reports present a clear overview of your financial situation and help you identify potential savings. Budget planning also helps you to implement consistent cost control. If you are looking for a simple digital budget book to help you keep track of your personal finances, it's definitely worth taking a closer look at Skrooge. ■■■

Info

[1] Skrooge: *http://skrooge.org/*

[2] Skrooge download: *https://skrooge.org/download/*

[3] AqBanking: *https://github.com/aqbanking/aqbanking*

[4] GnuCash: *https://www.gnucash.org*

[5] KMyMoney: *https://kmymoney.org*

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.

SCaLE20x

Date: March 9-12, 2023

Location: Pasadena, California

Website: *https://www.socallinuxexpo.org/blog/scale-20x*

The 20th Annual Southern California Linux Expo will take place on March 9-12, 2023 at the Pasadena Convention Center in Pasadena, California. SCaLE is the largest community-run open-source and free software conference in North America.

FOSS Backstage

Date: March 13-14, 2023

Location: Berlin, Germany + Online

Website: *https://23.foss-backstage.de*

Join us at FOSS Backstage for two days of exciting talks, workshops, and panels with the most relevant experts from the Free and Open Source Software world. Attend in Berlin or Online. Find out about new tools, best practices, and the latest ideas about the future of FOSS.

Events

FOSDEM 2023	Feb. 4-5	Brussels, Belgium	https://fosdem.org/2023/
ITEXPO	Feb. 14-17	Fort Lauderdale, Florida	https://www.itexpo.com/east/
DeveloperWeek	Feb. 15-23	San Francisco, California + Virtual	https://www.developerweek.com/
GeekBeacon Festival	Feb. 18	Virtual Event	https://gbfest.org/
FAST'23	Feb. 20-23	Santa Clara, California	https://www.usenix.org/conference/fast23
SCaLE 20x	Mar. 9-12	Pasadena, California	https://www.socallinuxexpo.org/blog/scale-20x
Cassandra Summit	Mar. 13-14	San Jose, California + Virtual	https://events.linuxfoundation.org/
FOSS Backstage 2023	Mar. 13-14	Berlin, Germany + Online	https://23.foss-backstage.de/
Everything Open 2023	Mar. 14-16	Naarm (Melbourne), Australia	https://2023.everythingopen.au/
Women in CyberSecurity	Mar. 16-18	Denver, Colorado	https://www.wicys.org/events/wicys-2023/
CloudFest	Mar. 21-23	Europa-Park, Germany	https://www.cloudfest.com/
KubeCon + CloudNativeCon Europe 2023	Apr. 17-21	Amsterdam, Netherlands	https://events.linuxfoundation.org/
Cloud Expo Europe	May 10-11	FrankfurtFrankfurt, Germany	https://www.cloudexpoeurope.de/en
ISC High Performance	May 21-25	Hamburg, Germany	https://www.isc-hpc.com/about-overview.html
DrupalCon Pittsburgh	June 5-8	Pittsburgh, Pennsylvania	https://events.drupal.org/pittsburgh2023
stackconf 2023	Sep 13-14	Berlin, Germany	https://stackconf.eu/
Open Source Monitoring Conference (OSMC)	Nov 9-7	Nuremberg, Germany	https://osmc.de/

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	20, 28, 32, 71
Prof. Timo Baumann	44
Zack Brown	12
Bruce Byfield	6, 36, 50
Joe Casad	3
Mark Crutch	69
Jon "maddog" Hall	70
Swen Hopfe	64
Vincent Mealing	69
Brooke Metcalfe	60
Pete Metcalfe	60
Graham Morrison	82
Mike Schilli	54
Christian Schuler	44
Ferdinand Thommes	16, 40, 77
Daniel Tibi	88
Jack Wallen	8

Contact Info

Editor in Chief
Joe Casad, jcasad@linux-magazine.com

Copy Editors
Amy Pettle, Aubrey Vaughn

News Editors
Jack Wallen, Amber Ankerholz

Editor Emerita Nomadica
Rita L Sooby

Managing Editor
Lori White

Localization & Translation
Ian Travis

Layout
Dena Friesen, Lori White

Cover Design
Lori White

Cover Image
© Timofeev Vladimir, 123RF.com

Advertising
Brian Osborn, bosborn@linuxnewmedia.com
phone +49 8093 7679420

Marketing Communications
Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
4840 Bob Billings Parkway, Ste 104
Lawrence, KS 66049 USA

Publisher
Brian Osborn

Customer Service / Subscription
For USA and Canada:
Email: cs@linuxnewmedia.com
Phone: 1-866-247-2802
(Toll Free from the US and Canada)

For all other countries:
Email: subs@linux-magazine.com

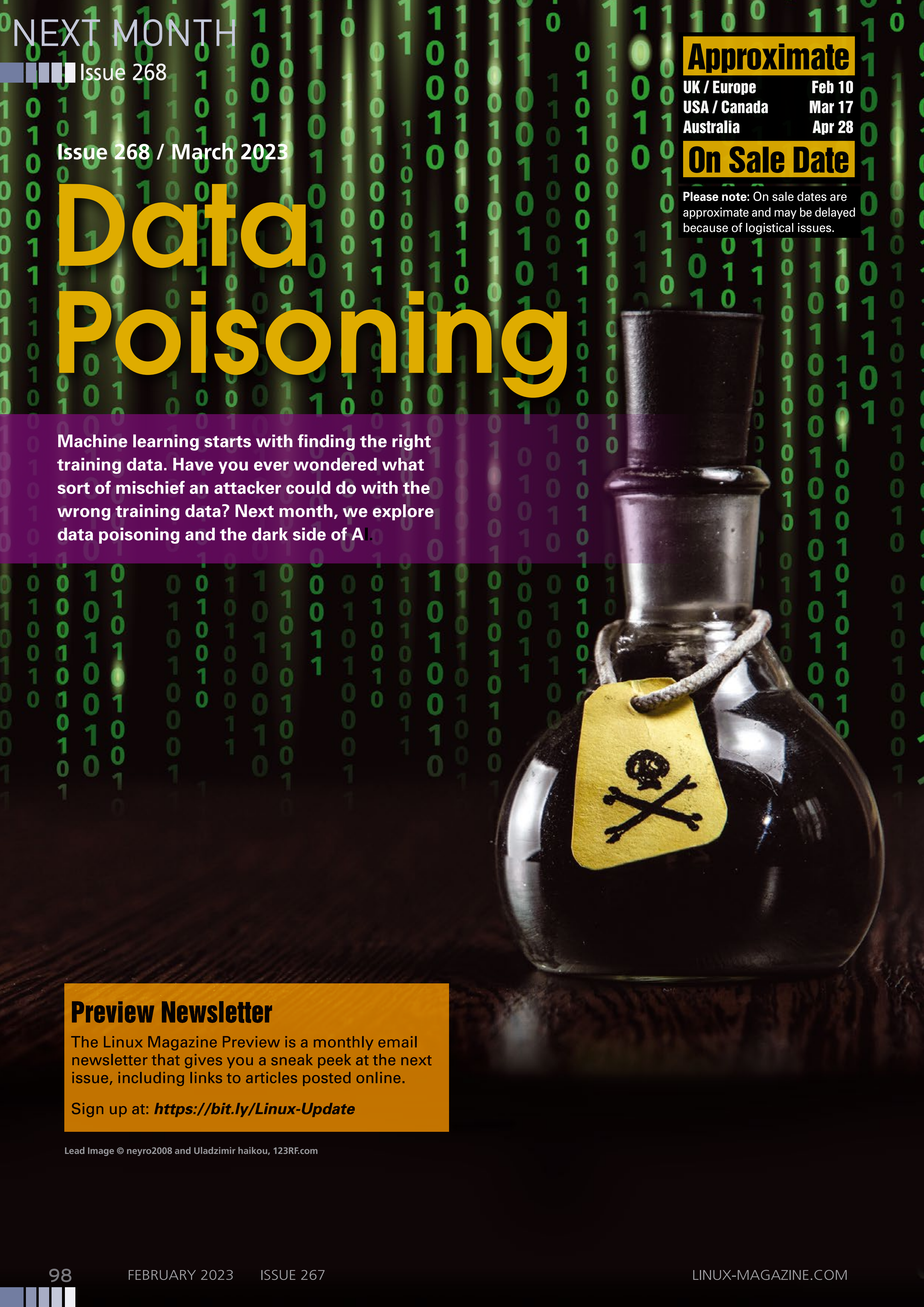
www.linux-magazine.com

Printed in Nuremberg, Germany by Zeitfracht GmbH.

Distributed by Seymour Distribution Ltd, United Kingdom

Represented in Europe and other territories by: Sparkhaus Media GmbH, Bialasstr. 1a, 85625 Glonn, Germany.

Published monthly as Linux Magazine (Print ISSN: 1471-5678, Online ISSN: 2833-3950) by Linux New Media USA, LLC, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA. Periodicals Postage paid at Lawrence, KS and additional mailing offices. Ride-Along Enclosed. POSTMASTER: Please send address changes to Linux Magazine, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA.



Issue 268 / March 2023

Data Poisoning

Machine learning starts with finding the right training data. Have you ever wondered what sort of mischief an attacker could do with the wrong training data? Next month, we explore data poisoning and the dark side of AI.

Approximate On Sale Date

UK / Europe	Feb 10
USA / Canada	Mar 17
Australia	Apr 28

Please note: On sale dates are approximate and may be delayed because of logistical issues.

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***

CLOUDFEST

March 21–23, 2023
Europa-Park, Germany

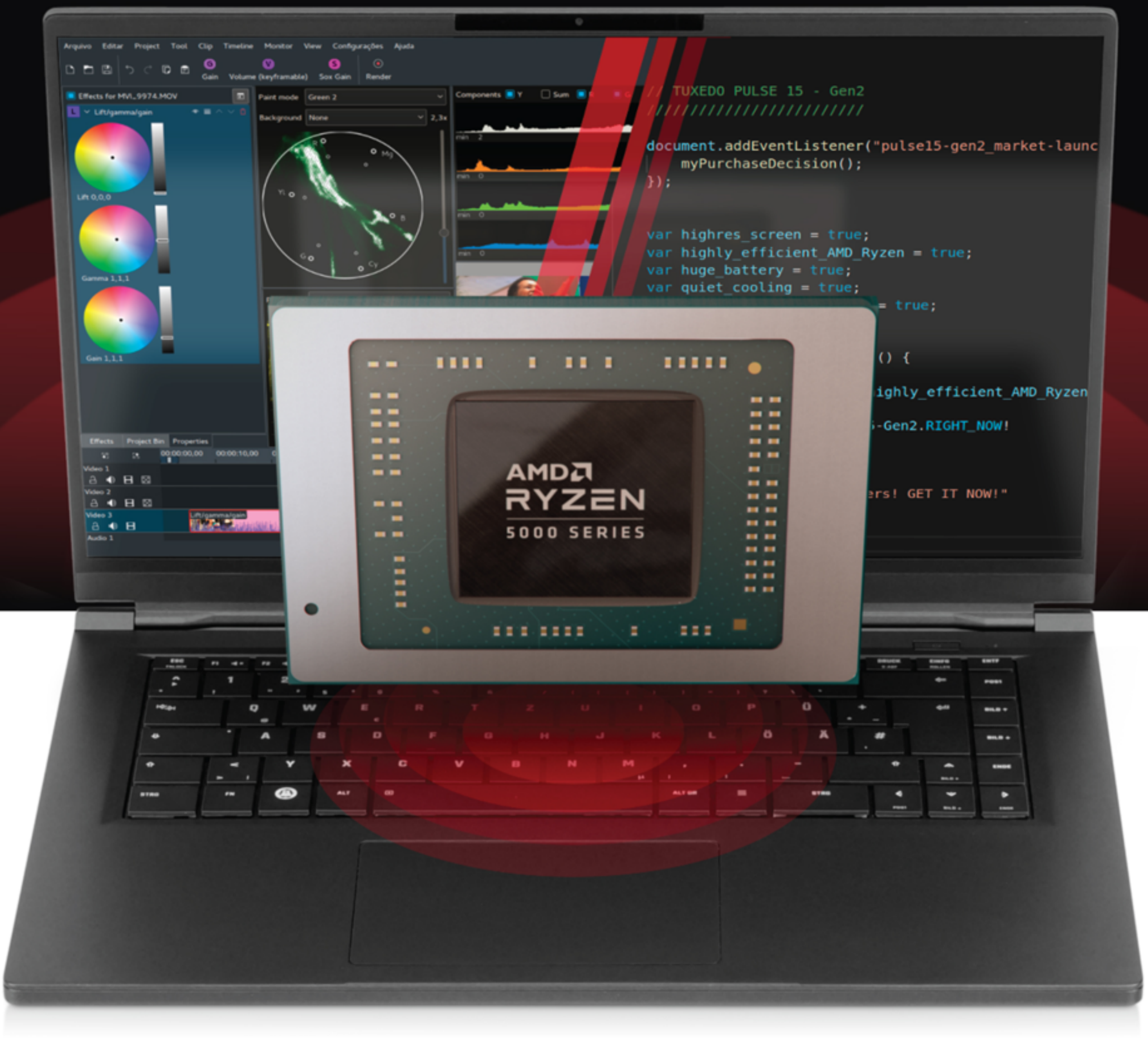
6,000+ Participants
250+ Speakers
150+ Partners
65 Countries

The world's largest cloud industry event is ready to once again take over a spectacular European amusement park to facilitate new partnerships, deep knowledge sharing, and the best parties the industry has ever seen.

scan me!

Start your CloudFest Journey
AND SAVE €399!
With your FREE Code: CF23ADMIN

reg.cloudfest.com



```
// TUXEDO PULSE 15 - Gen2
//////////////////////

document.addEventListener("pulse15-gen2_market-launc
    myPurchaseDecision();
});

var highres_screen = true;
var highly_efficient_AMD_Ryzen = true;
var huge_battery = true;
var quiet_cooling = true;
                                        = true;

                                        () {
                                        ighly_efficient_AMD_Ryzen
                                        -Gen2.RIGHT_NOW!

                                        ers! GET IT NOW!"
```

Qui(e)te powerful!

TUXEDO Pulse 15 - Gen2

AMD Ryzen 7 5700U-35W
8 cores | 16 threads

WQHD display
2560 x 1440 | 165 Hz

Up to 18 h of runtime
91 Wh battery

Rigid magnesium chassis
1,7 cm thin | 1,5 kg light

100%
Linux

5
Year
Warranty

Lifetime
Support

Built in
Germany

German
Privacy

Local
Support

TUXEDO

tuxedocomputers.com