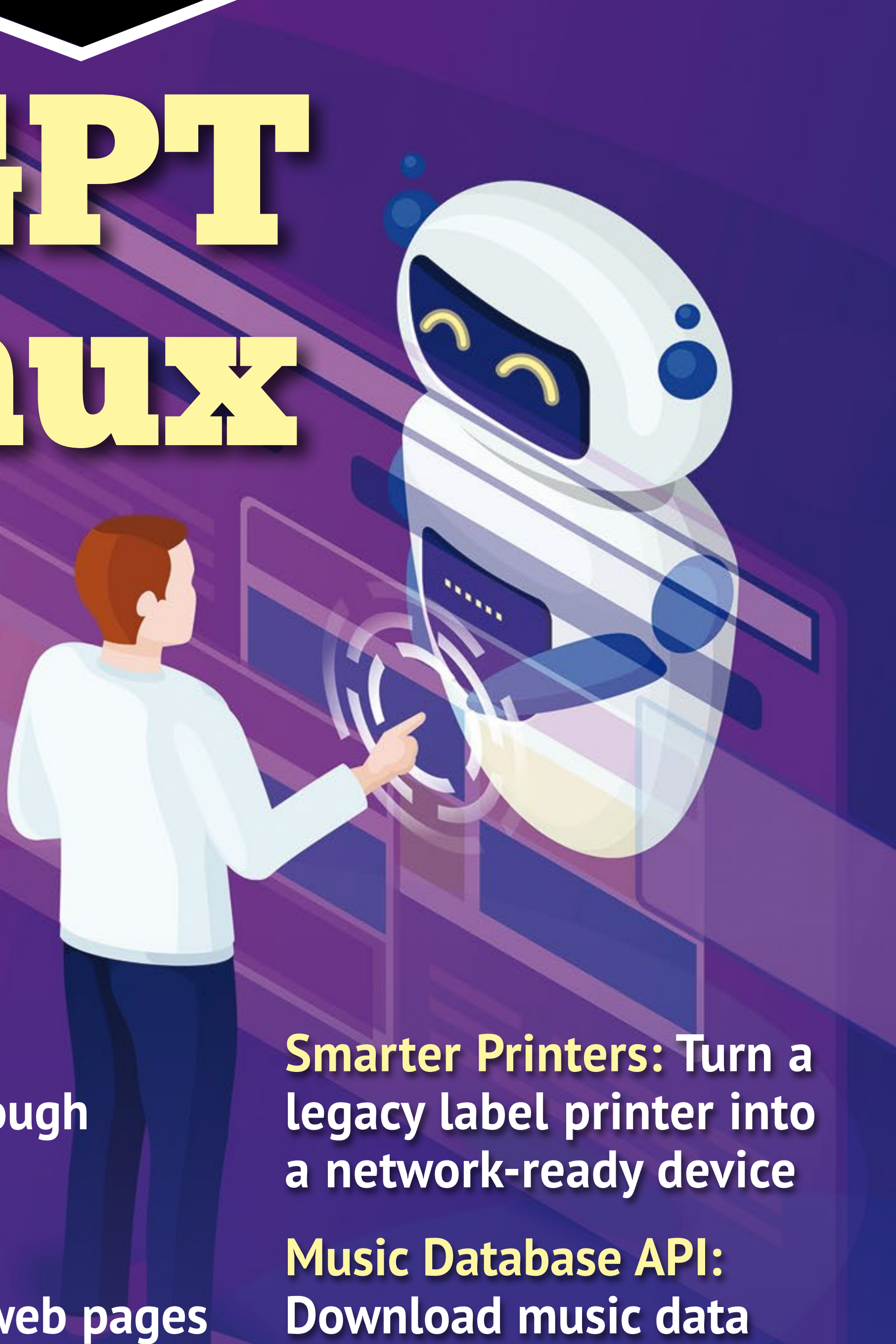
Timely Blender: Build a model of a clock

LINUX MAGAZINE

ISSUE 276 – NOVEMBER 2023

ChatGPT on Linux

Client apps for accessing the ChatGPT API



ProxyChains
Private surfing through proxies

Xidel
Extract data from web pages

Web Serial USB
Control a Rasp Pi over a USB interface

Smarter Printers: Turn a legacy label printer into a network-ready device

Music Database API: Download music data with a well-crafted URL

10 FANTASTIC FOSS FINDS!

LINUX NEW MEDIA
The Pulse of Open Source

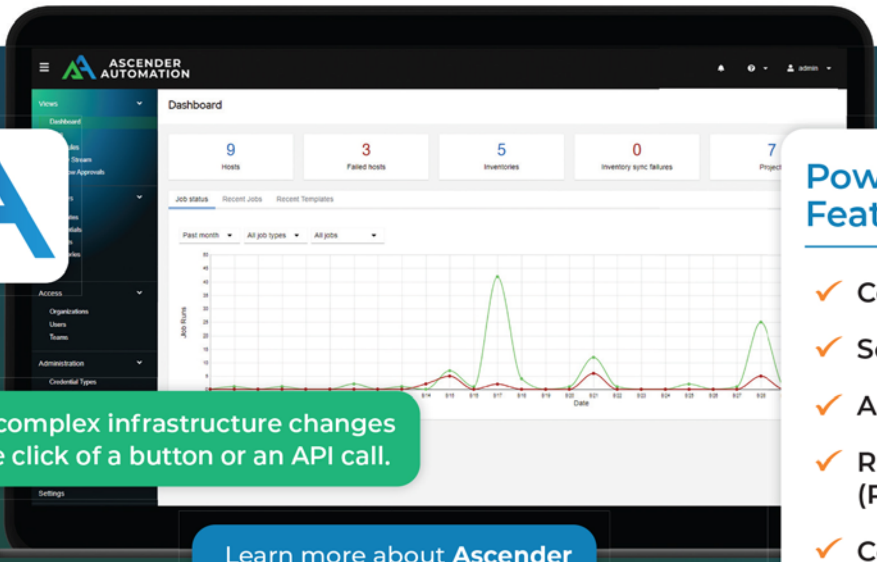
WWW.LINUX-MAGAZINE.COM

DO THE IMPOSSIBLE

INTRODUCING

Ascender Enterprise Infrastructure Automation

Ascender is a platform that provides all of the structure to implement enterprise infrastructure automation via turnkey Ansible playbooks—effectively, securely, and at scale.



Enable complex infrastructure changes with the click of a button or an API call.

Learn more about **Ascender** at www.ciq.com/Ascender

Powerful Automation Features

- ✓ Centralized Management
- ✓ Scheduling and Job Monitoring
- ✓ Auditing and Reporting
- ✓ Role-Based Access Control (RBAC)
- ✓ Collaboration and Version Control
- ✓ Workflow Automation

Watch Ascender Overview Demo

Secure, Reliable, and Open Infrastructure Solutions

Mountain	**Ascender**	**Fuzzball**	**Rocky Linux**	**Apptainer**	**Warewulf**
Lifecycle Management	Infrastructure Automation	Performance Computing	Community Enterprise Linux	Application Containers	Cluster Management

www.CIQ.com

SOMEWHERE IN BETWEEN

Dear Reader,

No one would accuse us of being a business and finance magazine, but we do cover technology, and the tides of tech often follow the tidings of business events. I sometimes wonder if our discussions of business sound as fluffy to business people as a business magazine's discussion of tech sounds to us. The reference frames are so different, and the truth is usually hidden somewhere in between.

The big news this month was the Initial Public Offering (IPO) of chip designer ARM Holdings. ARM is known for its unusual niche in the computer hardware universe. They don't make chips themselves, but they license their designs to other manufactures, including Qualcomm, Broadcom, Samsung, and many more.

ARM is well known for its presence in smartphones and other mobile devices. ARM-designed devices are thought to inhabit nearly 99 percent of all smartphones. The ARM processors, it seems, use less power than the alternatives. This emphasis on low power is even helping ARM make some headway in the high-performance computing (HPC) market. Even in the all in, full-power world of HPC, power efficiency is important, because power usage and cooling (which is often related to power usage) are often limiting factors in the design.

I was encouraged to hear that ARM was offering an IPO because they are an established company with an actual history and a catalog of successful products. I don't know how many IPOs we've reported on through the years for startups that are projected to make a profit in five years if a lot of things that are also "projected" happen according to projections. A stable company with stable products entering the stock exchange? Real tech? What could be healthier?

A close look at this one, though, shows that this IPO defies any simple explanation. First of all, the Japanese holding company SoftBank, the current owners of ARM, placed only 9 percent of ARM shares in the IPO. One of the main reasons for most IPOs is to raise money for the company to develop and expand (as well as to make the current owners rich). Putting up only 9 percent of the shares limits how much money you can raise, and it also limits how rich you can make the current owners.

And then there is the matter of the $8.5 billion in loans SoftBank took out with 75 percent of ARM shares as collateral. This mini-IPO is apparently tied in somehow with the terms of the loans. Why did they borrow the money? A tech watcher would say that ARM's work is concentrated on the design end, and it takes a long time to roll out new chip designs. They are thinking about what they'll need in order to have designs ready five years from now. And then there is the overall decline in smartphone sales (ARM's cash cow) and the trade uncertainties in China, where most smartphones are made. Will ARM need the money to branch into other areas?

A business watcher, however, would say, "Silly techies, this isn't about ARM – it is about SoftBank." ARM is an asset, which SoftBank is free to leverage, sell, or deploy in any way it can to improve the overall position of the portfolio. Some of SoftBank's priorities have little to do with ARM and its chip designs. In fact, SoftBank tried to sell ARM to NVIDIA just last year, and they would have succeeded if the US Federal Trade Commission had not put on the brakes.

As a business matter, SoftBank has plenty of uses for the $8.5 billion that don't have anything to do with grinding out new chip designs. For instance, the company is known to be interested in the emerging AI field and has even discussed a possible alliance with OpenAI, creators of ChatGPT. Selling only 9 percent of the shares keeps the supply of shares low, which helps to prop up the share price and, in turn, ARM's overall valuation. Many commentators have noted that the ARM share price was significantly overvalued based on its price to earnings ratio, which adds to the weirdness of the whole event.

Does the ARM IPO indicate an investment in ARM, or is it a signal that SoftBank is trying to extricate itself from ARM? Probably a little of both, or you could say, somewhere in between.

It's always somewhere in between.

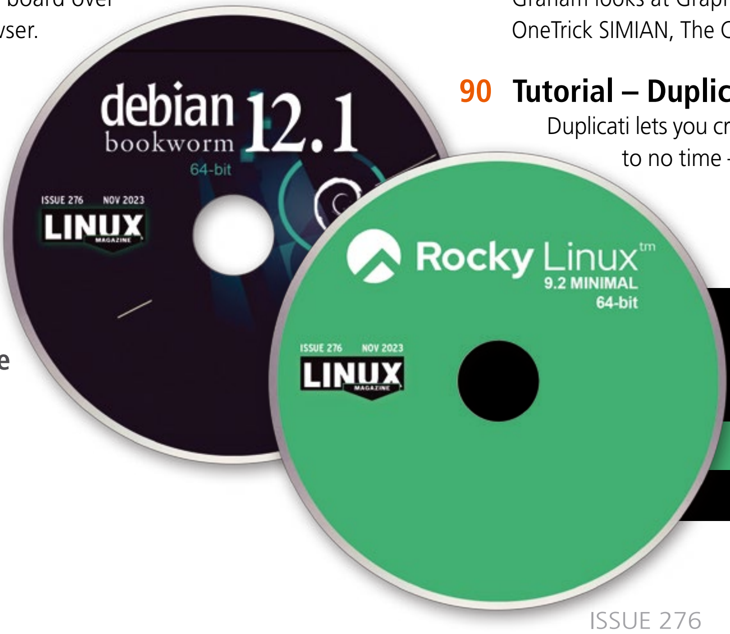
Joe

Joe Casad,
Editor in Chief

LINUX MAGAZINE

NOVEMBER 2023

@linux_pro
@linuxpromagazine
Linux Magazine
@linuxmagazine



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

Rocky Linux 9.2 and Debian 12.1

Two Terrific Distros on a Double-Sided DVD!

Rocky Linux 9.2

64-bit

Rocky Linux is one of the leading community-based drop-in replacements for Red Hat Enterprise Linux (RHEL). The 9.2 release can be used as an upgrade from earlier 9.x releases, but users of 8.x releases are advised to make a fresh install.

Changes in Rocky Linux 9.2 mostly involve improving compatibility with RHEL, including greater reliability on RHEL 9 than on CentOS Stream 9 and changes and additions to container packages, including the images for Microsoft Azure. In addition, 9.2 features enhanced tools for creating installation images; upgrades to packages in the basic tool chain, such as GCC and binutils, and in container tools; and numerous security fixes. Like RHEL, all these changes are geared towards the enterprise, although veterans of all descriptions will find the upgrade worth exploring.

Debian 12.1

64-bit

Debian is one of the oldest and largest distributions, and the foundation of almost two-thirds of all the current distributions, including Ubuntu, Linux Mint, MX Linux, Knoppix, and Pop!_OS. Debian's popularity is due mainly to its strict packaging guidelines and its frequent security updates. Although Debian rarely has the latest packages, it can be trusted for its security on networks and home workstations alike.

Debian 12, codenamed bookworm after a character in the *Toy Story* movies, was released on June 10, 2023. Debian 12.1 is its first update. Like all Debian point releases, Debian 12.1 contains all the security and bug fixes released separately since the previous release plus a few extras – in this case, over 100 changes total, as detailed in the official announcement of the release (*https://www.debian.org/News/2023/20230722.en.html*). The result is a stable and secure operating system for new and existing users alike.

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

NEWS

Updates on technologies, trends, and tools

THIS MONTH'S NEWS

KSMBD Finally Reaches a Stable State

KSMBD (the kernel SMB daemon) is the in-kernel module, developed by Samsung that implements the SMB/CIFS protocol for sharing files and folders over a network. The SMB 3 server could take the place of the traditional Samba software.

KSMBD was originally merged for Linux 5.15 but was tagged as experimental. That came about in 2021, and it has taken some time to get KSMBD to a state that was considered stable. That time has come, and KSMBD is planned for Linux kernel 6.6.

Why is KSMBD important? First off, it promises considerable performance gains and better support for modern features such as Remote Direct Memory Access (RDMA). KSMBD also supports a number of features such as multiple dialects (SMB 2.1, SMB 3.0, SMB 3.1), oplock cache mechanism, compound requests, ACL, and DCE/RPC.

KSMBD also adds enhanced security, considerably better performance for both single and multi-thread read/write, better stability, and higher compatibility.

In the end, hopefully, this KSMBD will also mean easier share setups in Linux without having to jump through the same hoops one must with the traditional Samba setup.

Only time will tell. Until then, you can read more about KSMBD from this LWN article (*https://lwn.net/Articles/865350/*).

Nitrux 3.0.0 Released

Nitrux is a Linux distribution based on Debian that is fairly young but is already making a big impression.

Code-named "ut," Nitrux 3.0.0 brings some serious improvements to the operating system, which include new features, plenty of updates, and a number of performance optimizations.

The biggest changes come by way of the Nitrux Update Tool, which includes a rescue option that allows you to restore the root partition from a live session. This can be a real lifesaver should something go wrong during an update.

There also are plenty of tweaks to the Calamares installer (such as the disabling of auto-login by default), the removal of a deprecated kernel parameter, and a change to the order of execution of certain Calamares modules.

Other updates include Firefox 117.0, Mesa 23.3, Nitrux Update Tool 1.1.3, Kernel Boot 0.0.7, NVIDIA Linux x64 display driver 535.104.05, AMD open source driver for Vulkan 2023.Q3.1, zsync2, libappimageupdte 2.2.0-alpha-1-20230831+nitrux, and MauiKit and MauiKit Frameworks 3.0.1.

You can download an ISO for installation from SourceForge (*https://sourceforge.net/projects/nitruxos/files/Release/ISO/*) and read the full release notes here: *https://nxos.org/changelog/release-announcement-nitrux-3-0-0/*.

Linux From Scratch 12.0 Available

For those who prefer to build their own version of Linux, the Linux From Scratch (LFS) project has released version 12.0. The latest iteration has been released along with Beyond Linux From Scratch (BLFS) 12.0, which includes a systemd variant.

LFS v12.0 updates a number of packages including GCC 13.2, glibc 2.38, GNU Binutils 2.41, and the Linux 6.4.12 upstream kernel. In fact, you'll find 38 packages were updated since the last release. Other updates include libxcrypt and the new Python module flit-core.

As well, LFS now makes use of pkgconf (in place of pkg-config), which is an application that helps to configure compiler and linker flags for development libraries.

There was one major deprecation notice announced with LFS 12.0: Future versions of BLFS will remove the LXDE Desktop environment and the Reiser filesystem, both of which are no longer maintained.

LFS isn't just a system for building your own Linux distribution. It's also a book (or a collection of books) that provides step-by-step instructions for building your own, customized Linux distribution.

You can read the new LFS 12.0 book on line (*https://www.linuxfromscratch.org/lfs/downloads/stable/*) or download a PDF version (*https://www.linuxfromscratch.org/lfs/read.html*).

Linux Kernel 6.5 Released

The latest iteration of the Linux kernel has arrived and it contains some pretty cool features. According to Linus Torvalds (the creator of Linux), this release "...has been going smoothly."

There are plenty of the usual code cleanups, fixes, and upgrades to existing features. But taking the limelight for this release is the initial support for USB4 and WiFi 7.

WiFi 7 supports the 6Ghz band and has a max data rate of 23Gbps, so getting this rolled into the kernel should give Linux a big boost in network speeds. As well, USB4 supports up to 80Gbps data transfer rate, so there's another boost. Remember, however, this is just initial support, so don't expect to see these newer technologies working just yet.

Other changes added to the 6.5 kernel include fixes for Intel P-state CPU scaling, three modes for `amd-pstate` (active, passive, and guided autonomous), Btrfs performance and storage optimizations, rumble support for the latest iterations of Xbox controllers, overclocking support for AMD Radeon RX 7000 GPUs, optimizations for AMD and Intel graphics drivers, more Rust code, support for new hotkeys found on ThinkBook 14s Yoga ITL, and much more.

You can read the full list of changes and fixes in this post by Torvalds (*https://lore.kernel.org/lkml/CAHk-=wgmKhCrdrOCjp=5v9NO6C=PJ8ZTZcCXj09piHzsZ7qqmw@mail.gmail.com/?ref=news.itsfoss.com*).

When the Linux 6.5 kernel hits your distribution will vary. Rolling release distributions like Arch and Fedora (a semi-rolling release) should see it soon. Ubuntu (and its derivatives) should see 6.5 along with Ubuntu 23.10.

UbuntuDDE 23.04 Available

UbuntuDDE makes it possible to enjoy the Deepin Desktop with a base of the latest Ubuntu release. This combination is not only beautiful, it's also reliable and secure.

This new release uses Deepin 23 (from May 2023), which comes from the upstream Deepin Desktop Environment and includes some (but not all) of the usual Deepin tools (such as Music, Move, Calculator, Log Viewer, and Text Editor).

You'll also find Firefox, Gimp, LibreOffice, and Thunderbird along for the ride. As far as the kernel, UbuntuDDE ships with version 6.2.0-27-generic.

For those who need even more software, there's a Software Center (that isn't the default Deepin tool) as well as Snap. With that combination, you can install a plethora of applications (even apps like Spotify and Slack).

As far as the Deepin Desktop Environment is concerned, the 23rd iteration makes for an elegant and user-friendly desktop. For those who like to customize and configure, the Control Center is broken into easy-to-use categories, making the setup of things like user accounts, desktop preferences, printers, etc., even easier.

You can download an ISO for UbuntuDDE from the official download page (*https://ubuntudde.com/download/*) and read through the release notes (*https://ubuntudde.com/blog/ubuntudde-remix-23-04-lunar-release-note/*) to find out everything that went into making this new version.

MORE ONLINE

Linux Magazine
www.linux-magazine.com

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

Where Does Job Output Go?
• Jeff Layton
Where does your job data go? The answer is fairly straightforward, but I add some color by throwing in a little high-level background about what resource managers are doing and evolve the question to include a discussion of where data "should" or "could" go.

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

Test Your System to Help Fight Phishing Attacks
• Holger Reibold
The Gophish phishing framework lets you set up your own phishing campaigns to identify vulnerabilities and make users aware of these dangers.

Run Your Own Chat Server
• Andreas Stolzenberger
The Matrix open standard implements secure, decentralized communication and comes with bridges to popular chat apps.

Creating Load for Fun and Profit
• Federico Lucifredi
Standard loads are essential to benchmarking.

Star Labs Reveals a New Surface-Like Linux Tablet

Star Labs has reconfigured its StarLite tablet into a laptop in the same vein as the Microsoft Surface.

The new StarLite has a 12.5" LED-backlit 10-point touch display with IPS technology and runs at 2880x1920 resolutions with 276 pixels/inch. The included processor is a 1.00GHz quad-core Intel Alder Lake N200 with a turbo boost of up to 3.7GHz and a 6MB smart cache.

Internal storage is 512GB Gen3 PCIe SSD and is configurable up to 1TB or 2TB Gen3 PCIe SSD. RAM is 16GB of 4800MHz LPDDR5 (which is onboard and not upgradable).

Connectivity includes micro HDMI, 2 X USB Type C 3.2 Gen 2 with Power Delivery 3.0, microSD memory card reader, and a 3.5mm headphone jack. Wireless comes by way of an Intel Wi-Fi 5 9560 chip that supports 802.11ac up to 1.73Gbps and is also 802.11ac/a/b/g/n compatible. Finally, the StarLite supports Bluetooth 5.1.

As far as battery life is concerned, the StarLite delivers up to 12 hours on a full charge by way of a 38-watt-hour lithium-polymer battery and a 65W USB-C power adapter. The included keyboard is backlit with media keys, function lock, and international layouts.

The StarLite can be configured with Ubuntu, elementary OS, Linux Mint, Manjaro, Zorin OS, MX Linux, Xubuntu, Kubuntu, and more.

The StarLite can be purchased now for $498 (for the basic configuration) from the Star Labs website (https://us.starlabs.systems/pages/starlite?shpxid=4fa7bf99-f174-4af1-891c-c0ae40913a7e).

SUSE Going Private (Again)

Marcel holds 79 percent of the SUSE shares, and EQT Private Equity has announced that it will launch a voluntary public purchase offer to the remaining shareholders prior to the delisting. The offering price for those shares is EUR16.

Both the SUSE Management and Supervisory boards support this move as it will allow the company to shift its focus to its operation priorities and long-term strategies.

SUSE's CEO, Dirk-Peter van Leeuwen, said of the move, "I believe in the strategic opportunity of taking the company private – it gives us the right setting to grow the business and deliver on our strategy with the new leadership team in place." He continued, "EQT Private Equity's and SUSE's partnership in a private setting has been fruitful before and we are excited about the long-term potential of the company and our continued collaboration".

The timetable for the transaction looks like this: The offer document will soon be published by Marcel, which will be followed by a four-week acceptance period. The settlement of the offer is to occur in the first half of October 2023. During Q4 of 2023, a general meeting will be held to resolve the merger with an unlisted Luxembourg entity. Once all of this is finished, SUSE will then be delisted.

You can read a detailed description of the entire process from the official SUSE blog (*https://www.suse.com/news/EQT-announces-voluntary-public-purchase-offer-and-intention-to-delist-SUSE/*).

Devuan GNU+Linux Latest Release Available

The developers of the systemd-free Devuan GNU+Linux distribution have made a new release available for installation. This release offers four important updates, which start with it being based on Debian Bookworm (version 12). Next comes the kernel (version 6.1), which offers better support for newer hardware. Then, there's rootless `startx`, which uses *libseat1*. Finally, you'll find the Wayland GUI without the systemd `elogind`, which is used to track user sessions.

This release defaults to the Xfce 4.18 desktop but users can install Gnome 43 and/or KDE Plasma 5.27 from the distribution's repositories.

You'll also find Firefox ESR 102.13, LibreOffice 7.4.7, the Parole Media Player (version 4.16), and plenty of other applications.

The one thing to keep in mind is the installer isn't nearly on par with what you'll find with the likes of Ubuntu. Because of that, this distribution is probably best suited for experienced Linux users.

Devuan is a fork of Debian, created by a collective of veteran Unix admins. First announced in 2015, the distribution came about due to the controversy surrounding systemd.

You can download the latest version of Devuan from the official site (*https://www.devuan.org/get-devuan*) and read more about the release from the official announcement (*https://www.devuan.org/os/announce/daedalus-release-announce-2023-08-14*).

CIQ, Oracle, and SUSE Form Alliance to Thwart Near-Closing of the RHEL Source

CIQ, Oracle, and SUSE have come together to create the Open Enterprise Linux Association (OpenELA) (*https://openela.org/join/*). Because these organizations have business models that depend upon maintaining compatibility with RHEL, they created OpenELA to share resources and work communally on a solution that will provide downstream compatibility.

This new organization describes itself as a "community repository for enterprise Linux sources." What's at the heart of the OpenELA is that the source will be available with no subscriptions, passwords, or barriers. In fact, the group goes so far as to say, "Freeloaders welcome."

OpenELA's stated mission is to support continued access to source code that provides RHEL binary compatibility. To do this, OpenELA will offer open and free enterprise-grade Linux source code required to build systems compatible with RHEL 8 and RHEL 9.

OpenELA does not mince words when they say, "From Enterprise Linux downstream derivatives, to organizations that depend on Enterprise Linux, to vendors and individuals who just love being part of something amazing, YOU ARE INVITED!"

It's clear that this effort is for everyone who depends on RHEL but either needs help to afford the licensing for Red Hat's distribution or has soured on RHEL because they placed the source behind a paywall.

Currently, OpenELA is just getting off the ground. For more information, you can email them (*info@openela.org*) or join the OpenELA Team on Slack (*https://join.slack.com/t/openela/shared_invite/zt-20lv40mgy-fg9Sv6Uv7Hiqg8~WLZExZg*).

Rolling Release Rhino Linux Available

If you've used Linux long enough, chances are good you've heard of a rolling release, which is a distribution that is constantly up to date. Thanks to a continuous stream of updates, instead of having to upgrade from one major release to the next, it's all taken care of in the normal upgrade process.

Rhino Linux brings this to Ubuntu.

Rhino ships with a customized version of Xfce (version 4.18), called Unicorn, and uses a package manager called Pacstall, which is similar to Arch's AUR. Pacstall is a meta-package manager that combines Apt, Pacstall, Flatpak, and Snap into an easy-to-use GUI. There is also a customized Calamares installer and an app called Your System, which is a GUI tool that allows you to view system information at a glance.

The Unicorn desktop is customized with a handful of open source software components that were either borrowed from other distributions or created by the Rhino team.

Rhino Linux is aimed at typical users, but it is also developer-friendly, with plenty of software available for numerous use cases.

You can read all about the official Rhino Linux release here and download an ISO for installation from the official download page (*https://rhinolinux.org/releases/*).

Zack's Kernel News

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

By Zack Brown

Removing the Thorn

Sometimes in Linux kernel development, the best thing to come out of a discussion thread is an update to the documentation.

David Howells recently reported a problem that had been found by Matt Whitlock. Kernel code using the `splice()` system call to pipe data from a file into a running process could potentially see some data corruption. The `splice()` call could return successfully while the data it sent into the pipe could potentially be modified by another system call before the target process could finish reading from that pipe.

To fix this problem, David posted a patch that changed the data's owner before sending it into the pipe. That way, nothing would be able to modify that data. In the kernel, changing owners of RAM pages to protect them this way is called "stealing."

However, this was nowhere near the end of the story. Miklos Szeredi pointed out that "removing spliced pages from the cache is basically guaranteed to result in a performance regression for any application using splice." He added, "The above behavior is well known, so it's not likely to be a problem."

But Matt replied, "it's not well-known, as it's not documented. If the splice(2) man page had mentioned that pages can be mutated after they're already ostensibly at rest in the output pipe buffer, then my nightly backups wouldn't have been incurring corruption silently for many months."

Miklos quoted the "Notes" section of the `splice()` man page, which said that `splice()` did not actually copy data. Instead, `splice()` created a pointer to the data and only copied those pointers, not the data itself. This would result in the behavior David and Matt had seen – data sent into a pipe that could nevertheless be modified after `splice()` returned.

Matthew Wilcox complained that this was not at all clear from the documentation or from any semantics he would expect from a system call that was specifically supposed to copy data. He asked where he could find the library calls that would actually copy data, given that `splice()` didn't appear to do this.

At this point, Linus Torvalds brought the hammer down on the discussion, saying:

"It's called 'read()' and 'write()'.

"Seriously.

*"The *ONLY* reason for splice() existing is for zero-copy. If you don't want zero-copy (aka 'copy by reference'), don't use splice.*

"Stop arguing against it. If you don't want zero-copy, you use read() and write(). It really is that simple.

"And no, we don't start some kind of crazy 'versioned zero-copy with COW'. That's a fundamental mistake. It's a mistake that has been done – several times – and made perhaps most famous by Hurd, that made that a big thing.

*"And yes, this has been documented *forever*. It may not have been documented on the first line, because IT WAS SO OBVIOUS. The whole reason splice() is fast is because it avoids the actual copy, and does a copy-by-reference.*

"That's still a copy. But a copy-by-reference is a special thing. If you don't know what copy-by-reference is, or don't want it, don't use splice().

"I don't know how many different ways I can say the same thing.

"IF YOU DON'T WANT ZERO-COPY, DON'T USE SPLICE.

"IF YOU DON'T UNDERSTAND THE DIFFERENCE BETWEEN COPY-BY-VALUE AND COPY-BY-REFERENCE, DON'T USE SPLICE.

*"IF YOU DON'T UNDERSTAND THE *POINT* OF SPLICE, DON'T USE SPLICE.*

"It's kind of a bit like pointers in C: if you don't understand pointers but use them anyway, you're going to have a hard time. That's not the fault of the pointers. Pointers are very very powerful. But if you are used to languages that only do copy-by-value, you are going to think they are bad things."

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**.

Matt replied, "Thanks for being so condescending. Your reputation is deserved."

He also said that, at least on a cursory glance at the documentation, "it is not unreasonable to believe that the point of splice is to avoid copying between user-space and kernel-space," rather than the point of `splice()` being speed.

And in that case, he argued, `splice()` was the right choice because, "If you use read() and write(), then you're making two copies. If you use splice(), then you're making one copy (or zero, but that's an optimization that should be invisible to the user)."

To which Linus replied:

"No. It really isn't.

*"It is an optimization that is INHERENT IN THE INTERFACE and has been there literally since day #1. It was *never* invisible. It was the *point*.*

"You getting this use case wrong is not an excuse to change reality. It is, at most, a good reason to clarify the documentation.

"The 'without copying between kernel address space and user address space' is about the ability to not copy AT ALL, and yes, let's by all means clarify that part."

Linus also added, "When documentation and reality collide, documentation loses. That's how this works."

He went on to propose another way to get what Matt wanted:

"If you want 'one-copy', what you can do is:

- mmap() the file data (zero copy, not stable yet)

- use 'write()' to write the data to the network. This will copy it to the skbs before the write() call returns and that copy makes it stable."

Matt had had enough and replied:

"This entire complaint/discussion/argument would have been avoided if splice(2) had contained a sentence like this one from sendfile(2):

"'If out_fd refers to a socket or pipe with zero-copy support, callers must ensure the transferred portions of the file referred to by in_fd remain unmodified until the reader on the other end of out_fd has consumed the transferred data.'

"That is a clear warning of the perils of the implementation under the hood, and it could/should be copied, more or less verbatim, to splice(2)."

Linus agreed with that proposal, saying, "Internally in the kernel, the two really have always been more or less of intermingled. In fact, I think splice()/sendfile()/tee() could – and maybe should – actually be a single man-page to make it clear that they are all facets of the same thing."

The discussion skewed off at that point, into speculative considerations of additional data copying methods like the one Linus identified using `mmap()` and `write()`.

But clearly, the `splice()` documentation will be updated.

This was a fascinating discussion, and I hope that Matt discovered that his backups were corrupt before he actually needed to restore his data from those backups. David's patch may have been replaced by a single sentence in the documentation, but it's a single sentence that could save users from a lot of pain.

What Do You Do When There's Nothing to Do?

The automated bug-discovering tool `syzbot` posts anything it finds to the Linux Kernel Mailing List. Recently it reported a bug in Linux's Hierarchical File System (HFS) support. HFS is a very obsolete Apple filesystem that dates back to 1985. Even Apple doesn't support it anymore. Its big claim to fame was that it supported subdirectories – hence, "hierarchical."

Using `git bisect` (a powerful tool to find the exact patch that caused a problem), `syzbot` laid the blame at the feet of Arnd Bergmann, in a patch he submitted in November 2021. Arnd, in his own defense, said the patch was a simple attempt to get rid of a compiler warning after Linus Torvalds announced that no more compiler warnings would be tolerated in kernel code.

"My patch was a mechanical conversion from '/* panic? */' to 'WARN_ON()' to work around a compiler warning, and the previous code had been in there since the 2004 HFS rewrite by Roman Zippel," Arnd said.

Linus joined the discussion right away, noting, "since 2004, as you point out – you have to go back to before the git days to even see any development in this area."

Linus took a stab at debugging the `syzbot` report and untangled it to some extent, though he pointed out that this bug was very unlikely to ever be triggered. He remarked, "Looks like this is syzbot just mounting a garbage image (or is it actually some real hfs thing?)."

He went on to say:

"I suspect this code is basically all dead. From what I can tell, hfs only gets updates for

(a) syzbot reports

(b) vfs interface changes

"and the last real changes seem to have been by Ernesto A. Fernandez back in 2018."

Linus also reported, "Hmm. Looking at that code, we have another bug in there, introduced by an earlier fix for a similar issue [....] So we should probably fix that too." He whipped up a patch and sent it to the mailing list for testing.

As for whether HFS itself was truly dead, Arnd replied, "There is clearly no new work going into it, and most data exchange with MacOS would use HFS+, but I think there are still some users."

Jeffrey Walton spoke up, saying, "I've been running Linux on an old PowerMac and an old Intel MacBook since about 2014 or 2015 or so. I have needed the HFS/HFS+ filesystem support for about 9 years now [....] There's never been a problem with Linux and the Apple filesystems. Maybe it speaks to the maturity/stability of the code that already exists. The code does not need a lot of attention nowadays."

John Paul Adrian Glaubitz also said:

"HFS/HFS+ is also used by PowerPC users on PowerMac hardware for both exchanging data between MacOS and Linux as well as for booting Linux using GRUB with a HFS/HFS+ /boot partition.

"Debian's grub-installer creates an HFS partition from which GRUB loads the kernel and the initrd. In order to be able to update kernel and initrd, the running Linux system needs to be able to read/write HFS/HFS+ partitions which is used for the /boot partition."

Viacheslav Dubeyko took a look at the bug(s) and at Linus's patch, but he was not able to reproduce the issue. When debugging, the ability to cause the same problem to occur at will is generally a necessary first step. But Viacheslav said that as far as he could tell, the data corruption reported by `syzbot` "had happened somehow earlier. The reported issue is only a side effect of volume corruption. The real issue of HFS volume corruption had taken place before. And it was a silent issue somehow." But Matthew Wilcox pointed out that "Syzbot generates deliberately-corrupted (aka fuzzed) filesystem images. So basically, you can't trust anything you read from the disc."

Viacheslav replied, "If the volume has been deliberately corrupted, then no guarantee that file system driver will behave nicely. Technically speaking, inode write operation should never happened for corrupted volume because the corruption should be detected. [...] If we would like to achieve such nice state of HFS/HFS+ drivers, then it requires a lot of refactoring/implementation efforts."

However, he said he didn't think it was really worth the trouble, because very few people used these filesystems in Linux.

Michael Schmitz also looked at Linus's untested patch and wondered "whether the missing fd.entrylength size test in the HFS_IS_RSRC(inode) case was due to the fact that a file's resource fork may be empty?" However, Linus replied, "But if that is the case, then the subsequent hfs_bnode_read would return garbage, no?"

Linus continued, "But I really don't know the code, so this is all from just looking at it and going 'that makes no sense'. Maybe it _does_ make sense to people who have more background on it." Michael replied, "But I don't really understand the code too well either. I'll have to see for myself whether or not your patch does cause a regression on HFS filesystems such as the OF bootstrap partition used on PowerPC Macs."

Michael did a quick test and reported that Linus's patch seemed to work fine.

But the story doesn't end there.

Dmitry Vyukov remarked at some point during the conversation, "Most popular distros will happily auto-mount HFS/HFS+ from anything inserted into USB (e.g. what one may think is a charger). This creates interesting security consequences for most Linux users. An image may also be corrupted non-deliberately, which will lead to random memory corruptions if the kernel trusts it blindly."

To which Matthew said flatly, "Then we should delete the HFS/HFS+ filesystems. They're orphaned in MAINTAINERS and if distros are going to do such a damnfool thing, then we must stop them."

For John Paul Adrian, that seemed like going way overboard. He replied:

"Both HFS and HFS+ work perfectly fine. And if distributions or users are so sensitive about security, it's up to them to blacklist individual features in the kernel.

"Both HFS and HFS+ have been the default filesystem on MacOS for 30 years and I don't think it's justified to introduce such a hard compatibility breakage just because some people are worried about theoretical evil maid attacks.

"HFS/HFS+ [is] mandatory if you want to boot Linux on a classic Mac or PowerMac and I don't think it's okay to break all these systems running Linux."

Matthew shot back, "If they're so popular, then it should be no trouble to find somebody to volunteer to maintain those filesystems. Except they've been marked as orphaned since 2011 and effectively were orphaned several years before that (the last contribution I see from Roman Zippel is in 2008, and his last contribution to hfs was in 2006)."

In a later email he went on to say, "There are bugs in how this filesystem handles intentionally-corrupted filesystems. That's being reported as a critical bug because apparently some distributions automount HFS/HFS+ filesystems presented to them on a USB key. Nobody is being paid to fix these bugs. Nobody is volunteering to fix these bugs out of the kindness of their heart. What choice do we have but to remove the filesystem, regardless of how many happy users it has?"

Linus came down on the other side of that issue. He replied to Matthew, saying:

"We have tons of sane options. The obvious one is 'just don't mount untrusted media'.

"Now, the kernel doesn't know which media is trusted or not, since the kernel doesn't actually see things like /etc/mtab and friends. So we in the kernel can't do that, but distros should have a very easy time just fixing their crazy models.

*"Saying that the kernel should remove a completely fine filesystem just because some crazy use-cases that nobody cares about are broken, now *that* [is] just crazy.*

"Now, would it be good to have a maintainer for hgs [HFS]? Obviously. But no, we don't remove filesystems just because they don't have maintainers.

"And no, we have not suddenly started saying 'users don't matter'."

The discussion continued among various developers, but the decision was made. However, at one point Matthew offered some perspective:

"Google have decided to subject the entire kernel (including obsolete unmaintained filesystems) to stress tests that it's never had before. IOW these bugs have been there since the code was merged. There's nothing to back out. There's no API change to blame. It's always been buggy and it's never mattered before.

"It wouldn't be so bad if Google had also decided to fund people to fix those bugs, but no, they've decided to dump them on public mailing lists and berate developers into fixing them."

Eric W. Biederman offered some additional perspective:

"There are no known linux filesystems that are safe to mount when someone has written a deliberately corrupted filesystem on a usb stick.

"Some filesystems like ext4 make a best effort to fix bugs of this sort as they are discovered but unless something has changed since last I looked no one makes the effort to ensure that it is 100% safe to mount any possible corrupted version of any Linux filesystem.

"If there is any filesystem in Linux that is safe to automount from an untrusted USB stick I really would like to hear about it."

It's amazing to me that Linux continues to run on every piece of hardware that anyone actually uses, supporting all sorts of protocols and filesystems that have any users at all, even when the companies, like Apple, who created those things have long since stopped supporting them. It would have been the easiest thing in the world for Linus to throw up his hands and rip out that ancient code, but no. There are people using that code, so it stays in. Bug fixes will be accepted, and someday maybe even a maintainer will step forward to stand behind those remaining users until the end. ■■■

MSPGLOBAL | in association with Acronis

November 14-16, 2023 | Nürburgring, Germany

For MSPs ISVs MSSPs VARs DMRs Systems / Software Integrators IT professionals

Drive Digital Transformation

This international event connects managed service providers and IT professionals with the tools, insights, and personal connections to grow their businesses and deliver maximum value to their customers.

Hands-on masterclasses, keynotes, expert panels. Elite networking through parties, concerts, and on-track racin gevents.

Register for **FREE** with the Code:

Linux

reg.mspglobal.com

Highlight speakers include:

Bob Geldof
Rock star, Activist

Patrick Pulvermüller
CEO, Acronis

Sanna Marin
Former Prime Minister & Head of Government of Finland

... And the leading voices in the MSP industry

mspglobal.com

Accessing ChatGPT from the desktop or the Linux command line

Smart Assistant

Do you think ChatGPT only works in your web browser? You can also access the global chat phenomenon from your desktop – or even from the Linux command line.

By Koen Vervloesem

ChatGPT [1] is a chatbot developed by OpenAI [2] based on a large language model (LLM). You can have conversations with it, get answers to your questions, or let it write texts or code. With some guidance, the answers are quite useful. However, always keep in mind that ChatGPT lacks understanding: it is purely based on statistical patterns. Therefore, it's essential to critically evaluate the answers.

To begin using ChatGPT on the web, you need to create an OpenAI account. You can sign up by entering your email address and creating a password, or you can log in with a Google, Microsoft, or Apple account. Additionally, you will need to provide your name and mobile phone number, and you will need to answer questions about your intended use of OpenAI's services.

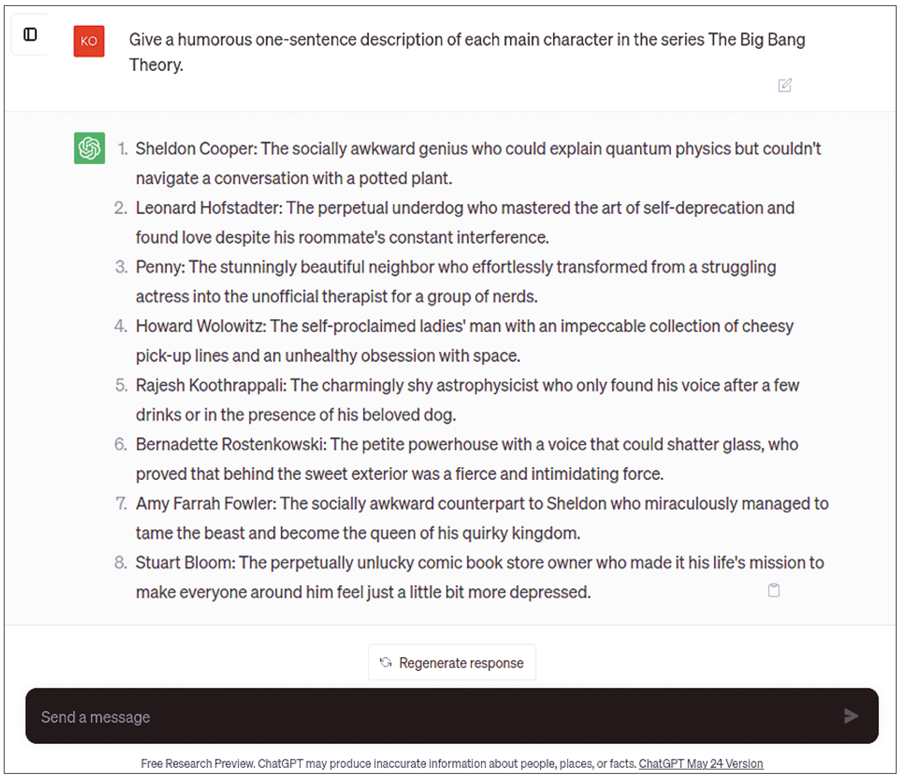
It's important to note that ChatGPT is still a prototype and access to the free version is not guaranteed. The website (Figure 1) often experiences high traffic, resulting in occasional messages asking that you try again later. Furthermore, the free version does not offer the latest version of the language model.

If you upgrade to ChatGPT Plus (for US$20 a month), you will always have access to the service, experience faster response times, and get earlier access to new features and versions of the language model. For example, ChatGPT Plus gives you access to GPT-4, a language model that is much more powerful than the GPT-3.5 model available in the free version.

API Access

In addition to the web interface, OpenAI provides an API that allows external applications to interact with ChatGPT. To access this API, you need to create an API key. Log in to the OpenAI platform site [3] with your OpenAI account, click on your username in the top-right corner, and then click on *View API keys* in the menu that appears. Next, click *Create new secret key*, give the key a name, and select *Create secret key*. Make sure to save this key immediately on your computer – you won't be able to retrieve it again after clicking on *Done* (Figure 2). You can create multiple keys, and it is a good idea to create a separate key for each application, as you can revoke individual keys if needed.

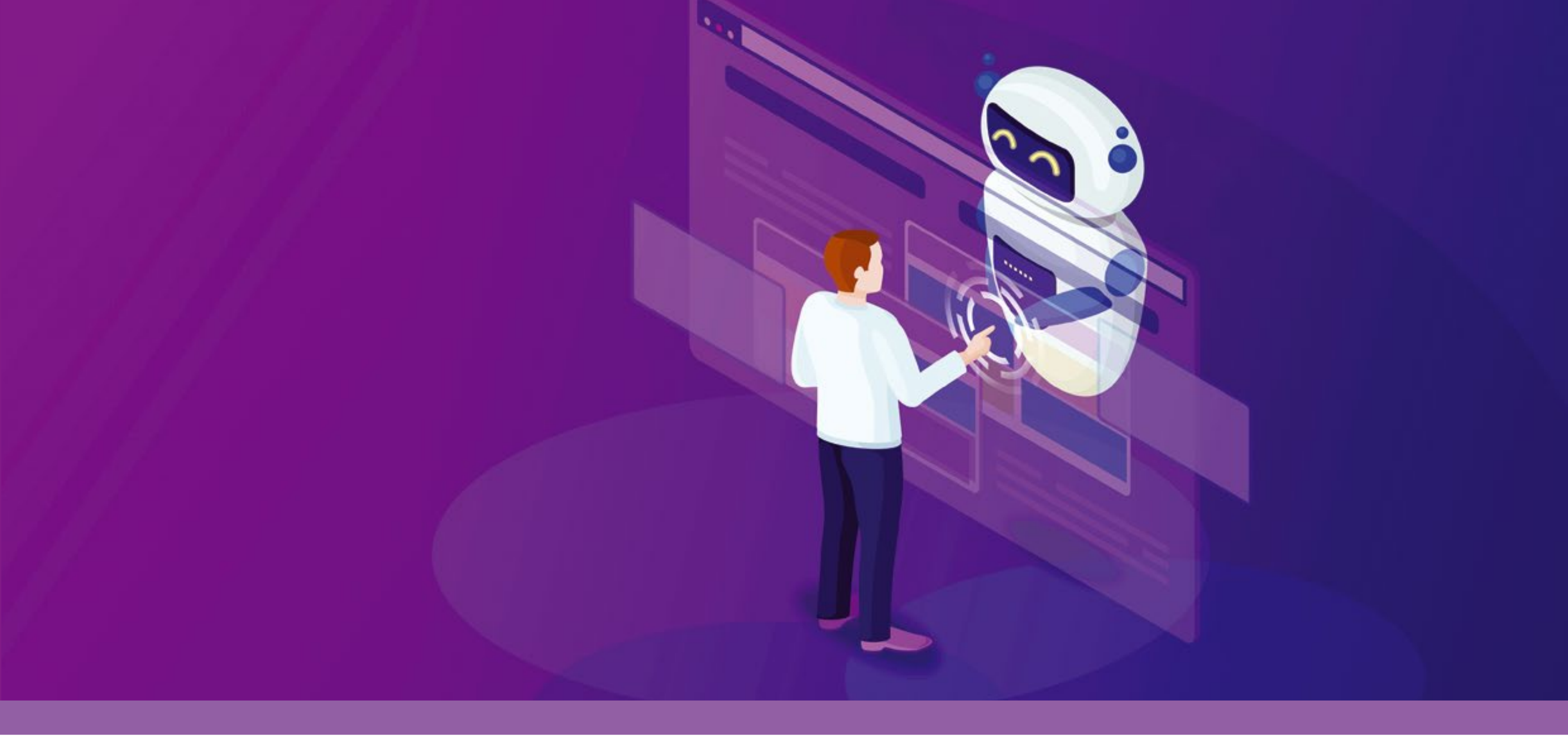
As a new user, you receive an initial credit in your OpenAI account, which is valid for three months. This credit allows you to experiment and evaluate the ChatGPT API. Afterwards, you need to provide credit card details. In many cases, using the API is cheaper than a ChatGPT Plus subscription. With the API, you pay based on the number of tokens used (1,000 tokens correspond to approximately 750 words). See the "Pricing and Models" box for more information.


KO Give a humorous one-sentence description of each main character in the series The Big Bang Theory.

1. Sheldon Cooper: The socially awkward genius who could explain quantum physics but couldn't navigate a conversation with a potted plant.
2. Leonard Hofstadter: The perpetual underdog who mastered the art of self-deprecation and found love despite his roommate's constant interference.
3. Penny: The stunningly beautiful neighbor who effortlessly transformed from a struggling actress into the unofficial therapist for a group of nerds.
4. Howard Wolowitz: The self-proclaimed ladies' man with an impeccable collection of cheesy pick-up lines and an unhealthy obsession with space.
5. Rajesh Koothrappali: The charmingly shy astrophysicist who only found his voice after a few drinks or in the presence of his beloved dog.
6. Bernadette Rostenkowski: The petite powerhouse with a voice that could shatter glass, who proved that behind the sweet exterior was a fierce and intimidating force.
7. Amy Farrah Fowler: The socially awkward counterpart to Sheldon who miraculously managed to tame the beast and become the queen of his quirky kingdom.
8. Stuart Bloom: The perpetually unlucky comic book store owner who made it his life's mission to make everyone around him feel just a little bit more depressed.

Regenerate response
Send a message
Free Research Preview. ChatGPT may produce inaccurate information about people, places, or facts. ChatGPT May 24 Version

Figure 1: Using ChatGPT on the web is easy, but the web version comes with some limitations.



Installing ChatWizard

After setting up your OpenAI account and obtaining an API key, you no longer need to use the website to access ChatGPT. An excellent desktop application for ChatGPT is ChatWizard [5], which works on Linux, Windows, and macOS. Although the developer primarily tests the application on macOS, he does make an effort to resolve any issues on other platforms as well. I have successfully tested ChatWizard on Ubuntu 22.04 LTS by downloading the AppImage of version 0.5.0.

To begin, download `chat-wizard_0.5.0_amd64.AppImage` (or the latest version available) from the ChatWizard releases [6] and make it executable:

```
chmod +x chat-wizard_0.5.0_amd64.AppImage
```

You can then launch the application:

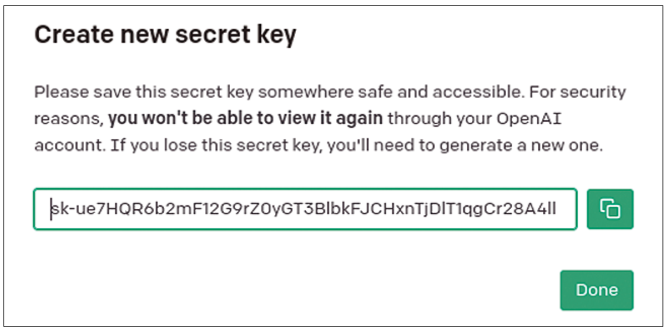
```
./chat-wizard_0.3.0_amd64.AppImage
```

Chatting on the Desktop

First click on the icon at the bottom-left to access ChatWizard's settings. The only field you need to fill in is the *Api Key* (Figure 3). Generate a new API key in your OpenAI account, as described earlier, and paste it here. After you click on the light bulb icon in the top-left corner, ChatWizard opens a *Casual Chat*. To start a conversation, simply type your question in the text field at the bottom (in any language you like), and ChatGPT's response will appear at the top (Figure 4).

If you find that ChatGPT's response is excessively lengthy, and you've gathered enough information, you can interrupt it by clicking *Stop replying*. Upon completing a response, you will see the cost of the API calls for that conversation displayed at the bottom-left. Additionally, you can click on *Export* at the bottom-right to select specific questions and answers for export, generating a PNG file with the result. By mousing over a question or answer, you can access icons to delete, edit, or copy the text to the clipboard. Resending a question is also possible (by clicking on the paper plane icon) to explore different responses to the same query.

Clicking on the three dots at the top-right corner allows you to adjust various settings for ChatGPT in this conversation, such as the language model (GPT-3.5 Turbo by default, but you can switch to GPT-4), "temperature," presence penalty, and frequency penalty. Only adjust the latter three parameters if you have a thorough understanding of their impact (see the box entitled "Adjusting Language Model Parameters" for more information). In most


Create new secret key

Please save this secret key somewhere safe and accessible. For security reasons, **you won't be able to view it again** through your OpenAI account. If you lose this secret key, you'll need to generate a new one.

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

Figure 2: Create a new secret key for every application using the OpenAI API.

Pricing and Models

OpenAI provides several language models, each with its own capabilities. The default model is GPT-3.5 Turbo with a context size of 4,000 tokens. This means that the length of your input (the prompt) and the model's response (its completion) can't exceed 4,000 tokens. There's also a GPT-3.5 Turbo model with a 16,000-token context. OpenAI's latest model, GPT-4, comes in versions with 8,000 and 32,000-token contexts. The OpenAI pricing [4] page lists detailed costs associated with using the API for these models. For example, using GPT-3.5 Turbo 4K incurs a cost of $0.0015 per 1,000 tokens for the input and $0.002 per 1,000 tokens for the model's response. In comparison, GPT-4 8K costs $0.03 per 1,000 tokens for the input and $0.06 per 1,000 tokens for the response. This makes GPT-4 roughly 20 times more expensive than GPT-3.5 Turbo. It is better to use GPT-4 only when you require its broader general knowledge and advanced reasoning capabilities. Previously, access to GPT-4 through the API required joining a waiting list. However, OpenAI has made the model generally available for paying API users as of July 2023.

cases, the default values are sufficient. When selecting the GPT-4 model, keep an eye on the cost displayed at the bottom-left.

Conversations and Prompts

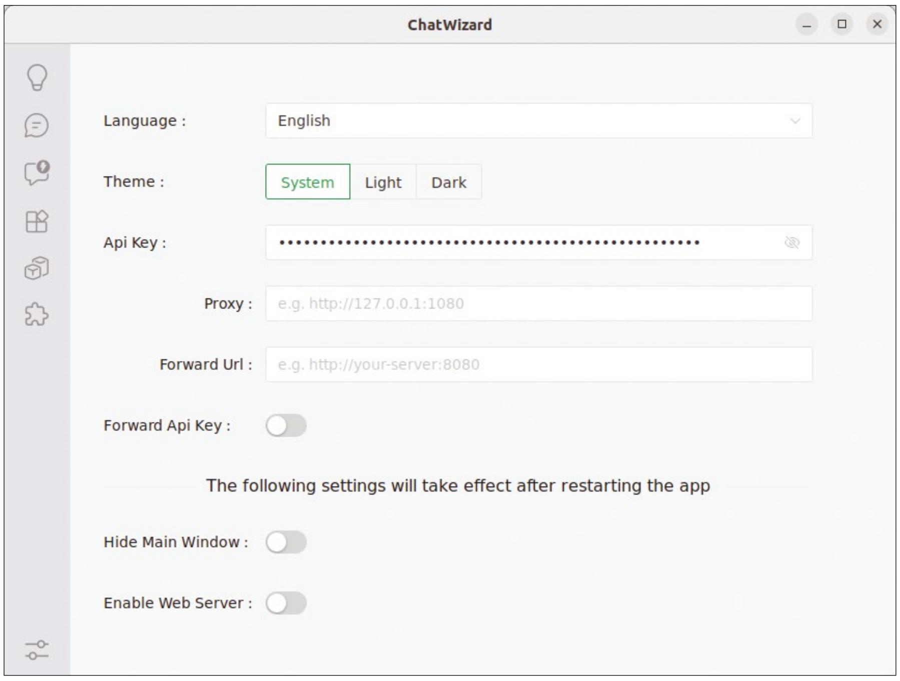
Whereas the *Casual Chat* option allows for quick, free-flowing conversations, ChatGPT gives better results when starting a new chat for each separate topic. After all, the model takes into account the context, including previous questions within the current conversation. To initiate a new chat, click on the speech bubble icon (*Topic Chat*) in ChatWizard and then on the green *New Chat* button. From there, you can start a new conversation in the same way as the casual chat. The cost displayed at the bottom-left now corresponds to that specific conversation. You can rename, pin, archive, and delete any conversation with a right-click within the list on the left-hand side.

Another way to get more out of ChatGPT is by using prompts. Prompts are specific instructions provided at the beginning of a conversation, guiding ChatGPT to behave in a specific way or take on a specific role. Prompts often result in more insightful answers. ChatWizard conveniently supports this feature. Click on the chat icon with the lightning bolt and then on the *New Prompt* button. Give your prompt a name and enter your instructions. You can then right-click on a prompt from your list and choose *New Chat* to begin a conversation with ChatGPT following these instructions. The name of your prompt is displayed at the top-right within ChatWizard, and hovering over it reveals the full prompt.

Creating your own prompts (which is an art in itself) isn't necessary, as ChatWizard offers an online "marketplace" with readily available prompts. Open it by clicking on the icon with the four cubes (Figure 5). By default, ChatWizard displays prompts sourced from Awesome ChatGPT Prompts [7]. Unfortunately these prompts are not alphabetically sorted, and there's no search function within ChatWizard. Therefore, you'll have to scroll through the entire list or search on the Awesome ChatGPT Prompts website. Once you've found a prompt you would like to use, select it to view the full prompt and click on the basket icon at the top-right, or right-click on the prompt's name and choose *Install*. The prompts will then appear in your list of prompts. You can still edit the prompt if necessary before starting a new chat with it.

More Features

ChatWizard has a few more interesting features. For example, clicking on the icon with two boxes allows you to add another model you can choose within a


ChatWizard

Language : English

Theme : System | Light | Dark

Api Key : ••

Proxy : e.g. http://127.0.0.1:1080

Forward Url : e.g. http://your-server:8080

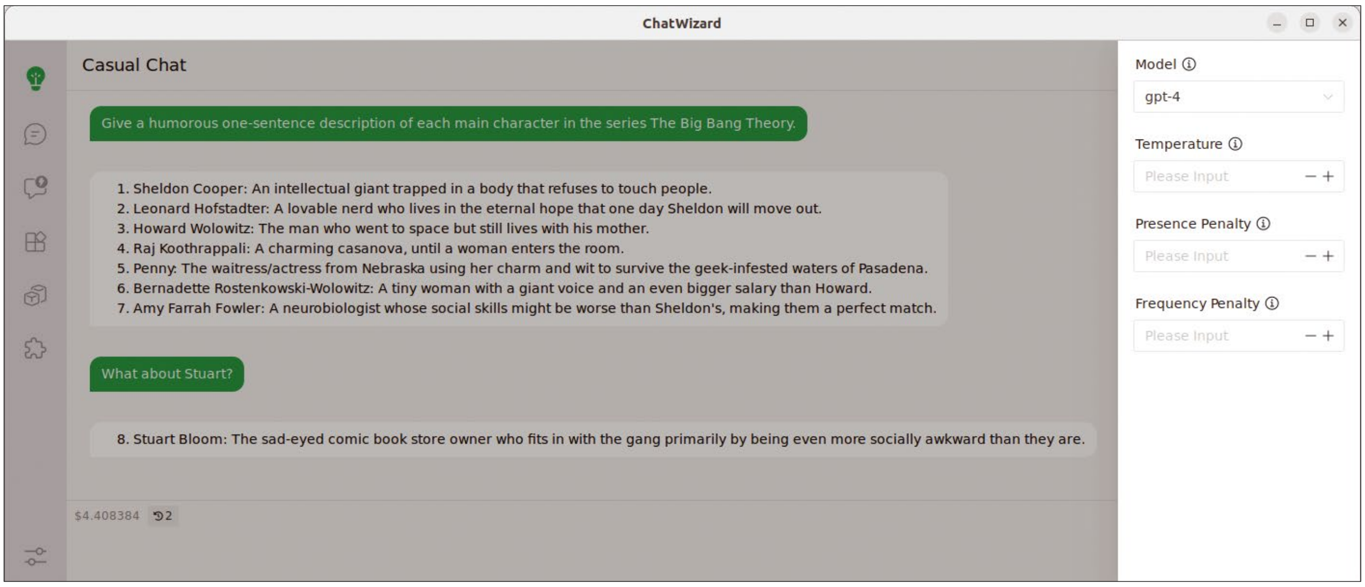
Forward Api Key :

The following settings will take effect after restarting the app

Hide Main Window :

Enable Web Server :

Figure 3: Paste your OpenAI API key in ChatWizard's settings to use the desktop application.


ChatWizard

Casual Chat

Give a humorous one-sentence description of each main character in the series The Big Bang Theory.

1. Sheldon Cooper: An intellectual giant trapped in a body that refuses to touch people.
2. Leonard Hofstadter: A lovable nerd who lives in the eternal hope that one day Sheldon will move out.
3. Howard Wolowitz: The man who went to space but still lives with his mother.
4. Raj Koothrappali: A charming casanova, until a woman enters the room.
5. Penny: The waitress/actress from Nebraska using her charm and wit to survive the geek-infested waters of Pasadena.
6. Bernadette Rostenkowski-Wolowitz: A tiny woman with a giant voice and an even bigger salary than Howard.
7. Amy Farrah Fowler: A neurobiologist whose social skills might be worse than Sheldon's, making them a perfect match.

What about Stuart?

8. Stuart Bloom: The sad-eyed comic book store owner who fits in with the gang primarily by being even more socially awkward than they are.

$4.408384 2

Model: gpt-4

Temperature: Please Input

Presence Penalty: Please Input

Frequency Penalty: Please Input

Figure 4: A casual chat with GPT-4 in ChatWizard.

conversation. You can enter the model name `gpt-3.5-turbo-16k` to use GPT-3.5, with a context that's four times larger than the default model. Additionally, by providing the cost per 1,000 tokens (as listed on the pricing page), the cost displayed at the bottom-left in a conversation is calculated correctly.

Another notable feature, still in early development, is the plugins market. Clicking on the jigsaw icon gives you access to this market. You need to install `chat-wizard-cli` from the ChatWizard releases to use the plugins. Currently, two plugins are available: one that automatically generates Git commit messages based on changes made in the staging area and another one that allows you to chat on the command line.

On the Command Line

ChatWizard offers a command-line plugin to use ChatGPT, but several other alternatives are also available. One of the distinctive features of the command line is the ability to pipe output from one command to the input of another one. Simon Willison's `llm` [8], a command-line client for ChatGPT and other large language models, excels as a tool to pipe another application's output to ChatGPT.

Install `llm` as a Python application using the `pip` package manager for Python packages:

```
pip install llm
```

Next, create a new OpenAI API key and copy it. Then run the following command:

```
llm keys set openai
```

When prompted, paste the API key and press Enter. The key will now be used for all API calls made by the `llm` command.

Asking Questions

Asking questions to ChatGPT is easy with the `llm` command. Just run `llm` with the desired prompt as its argument:

```
llm "Ten Italian cities"
```

By default, the response appears token by token, similarly to the ChatGPT website. If you prefer to see the entire response at once (after a delay), disable streaming as follows:

```
llm --no-stream "Ten Italian cities"
```

The default model is GPT-3.5 Turbo; however, you can switch to GPT-4 by specifying the model:

```
llm --model gpt4 "Ten Italian cities"
```

Alternatively, you can use GPT-3.5 Turbo with a context of 16,000 tokens by specifying the model as chatgpt-16k.

Piping Command Output to llm

You can also pipe the output of another command to `llm`. The latter then uses this output as its prompt:

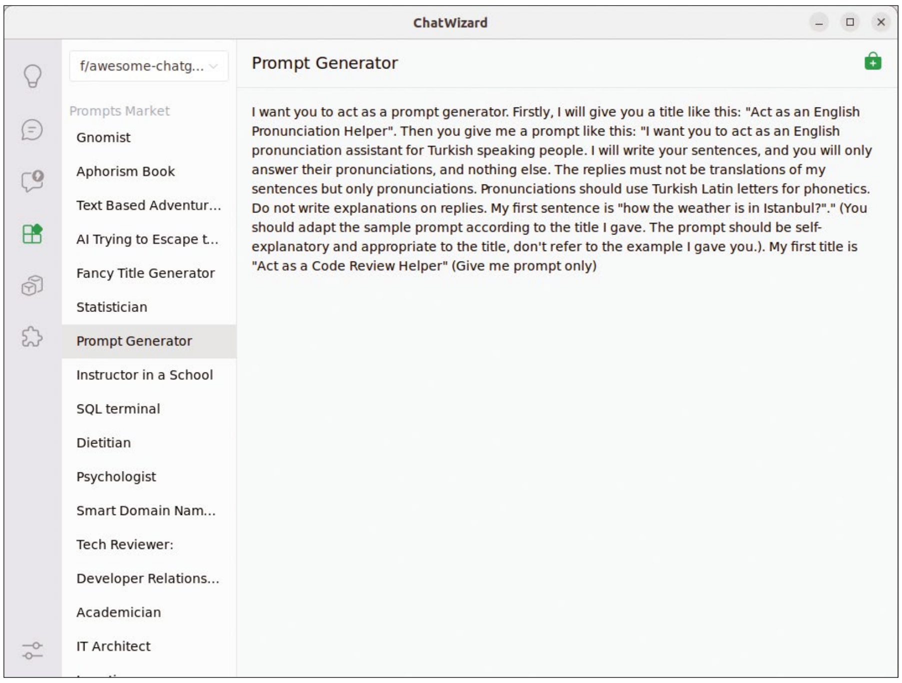
```
lsb_release --id --short | llm
```

As the `lsb_release` command feeds your distribution's name to `llm`, the latter describes this distribution in a few sentences.

Another way to use the output of commands in a prompt is by using `$(command)` inside a doubly quoted string. For example, if you're in a Git repository and you want to understand the changes in a specific commit, simply ask `llm`:

Adjusting Language Model Parameters

ChatGPT's models have three important parameters that you can adjust to modify the model's behavior. Using ChatWizard, you can experiment with different values for these parameters. Temperature ranges from 0 to 2, where 0 means that the answer is the same every time, and higher values introduce more randomness. If left unset, the default value is 1, resulting in moderate variation. Presence penalty and frequency penalty both range from -2 to 2 and have a default value of 0. A positive presence penalty reduces the likelihood of previously used tokens reappearing in the response, favoring discussions on new topics. With a positive frequency penalty, the model is less likely to reuse tokens that have already been used multiple times. This reduces the chances of the model repeating a sentence verbatim. Adjusting these parameters is rarely necessary; however, they can be useful if you notice the model providing repetitive answers to specific questions. In such cases, assigning both parameters a value between 0.1 and 1 can help reduce repetition.



I want you to act as a prompt generator. Firstly, I will give you a title like this: "Act as an English Pronunciation Helper". Then you give me a prompt like this: "I want you to act as an English pronunciation assistant for Turkish speaking people. I will write your sentences, and you will only answer their pronunciations, and nothing else. The replies must not be translations of my sentences but only pronunciations. Pronunciations should use Turkish Latin letters for phonetics. Do not write explanations on replies. My first sentence is "how the weather is in Istanbul?"." (You should adapt the sample prompt according to the title I gave. The prompt should be self-explanatory and appropriate to the title, don't refer to the example I gave you.). My first title is "Act as a Code Review Helper" (Give me prompt only)

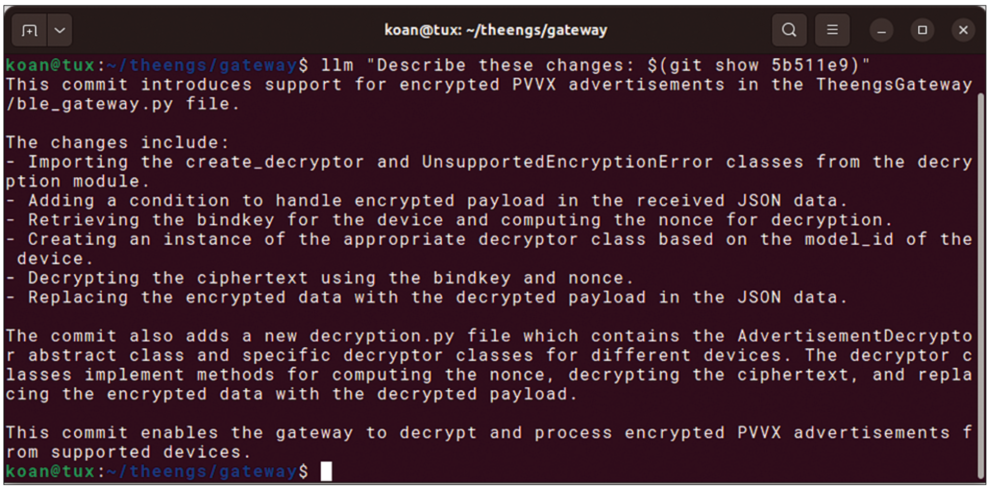
Figure 5: ChatWizard sources a list of readily available prompts from the Awesome ChatGPT Prompts website.

```
llm "Describe these changes: $(git show5b511e9)"
```

This command shows the changes in a particular Git commit and adds them to the prompt sent to the `llm` command (Figure 6).

You can also add a system prompt (comparable to ChatWizard's prompts) to guide the language model's behavior. With a system prompt, the previous command to describe changes in a Git commit can be asked like this:

```
git show 5b511e9 | llm --system "Give a high-level description
  of the changes in this Git diff output"
```



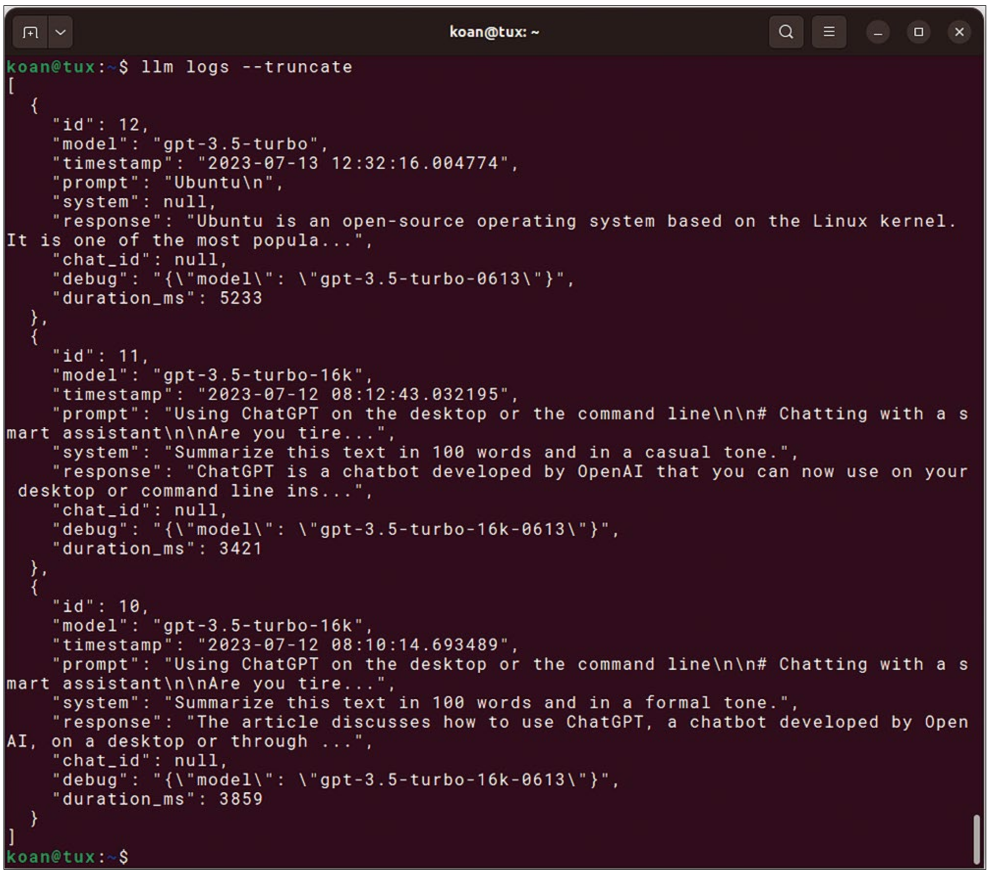
```
koan@tux: ~/theengs/gateway
koan@tux:~/theengs/gateway$ llm "Describe these changes: $(git show 5b511e9)"
This commit introduces support for encrypted PVVX advertisements in the TheengsGateway
/ble_gateway.py file.

The changes include:
- Importing the create_decryptor and UnsupportedEncryptionError classes from the decry
ption module.
- Adding a condition to handle encrypted payload in the received JSON data.
- Retrieving the bindkey for the device and computing the nonce for decryption.
- Creating an instance of the appropriate decryptor class based on the model_id of the
 device.
- Decrypting the ciphertext using the bindkey and nonce.
- Replacing the encrypted data with the decrypted payload in the JSON data.

The commit also adds a new decryption.py file which contains the AdvertisementDecrypto
r abstract class and specific decryptor classes for different devices. The decryptor c
lasses implement methods for computing the nonce, decrypting the ciphertext, and repla
cing the encrypted data with the decrypted payload.

This commit enables the gateway to decrypt and process encrypted PVVX advertisements f
rom supported devices.
koan@tux:~/theengs/gateway$ 
```

Figure 6: Ask `llm` to describe changes in a Git commit.



```
koan@tux: ~
koan@tux:~$ llm logs --truncate
[
  {
    "id": 12,
    "model": "gpt-3.5-turbo",
    "timestamp": "2023-07-13 12:32:16.004774",
    "prompt": "Ubuntu\n",
    "system": null,
    "response": "Ubuntu is an open-source operating system based on the Linux kernel.
It is one of the most popula...",
    "chat_id": null,
    "debug": "{\"model\": \"gpt-3.5-turbo-0613\"}",
    "duration_ms": 5233
  },
  {
    "id": 11,
    "model": "gpt-3.5-turbo-16k",
    "timestamp": "2023-07-12 08:12:43.032195",
    "prompt": "Using ChatGPT on the desktop or the command line\n\n# Chatting with a s
mart assistant\n\nAre you tire...",
    "system": "Summarize this text in 100 words and in a casual tone.",
    "response": "ChatGPT is a chatbot developed by OpenAI that you can now use on your
 desktop or command line ins...",
    "chat_id": null,
    "debug": "{\"model\": \"gpt-3.5-turbo-16k-0613\"}",
    "duration_ms": 3421
  },
  {
    "id": 10,
    "model": "gpt-3.5-turbo-16k",
    "timestamp": "2023-07-12 08:10:14.693489",
    "prompt": "Using ChatGPT on the desktop or the command line\n\n# Chatting with a s
mart assistant\n\nAre you tire...",
    "system": "Summarize this text in 100 words and in a formal tone.",
    "response": "The article discusses how to use ChatGPT, a chatbot developed by Open
AI, on a desktop or through ...",
    "chat_id": null,
    "debug": "{\"model\": \"gpt-3.5-turbo-16k-0613\"}",
    "duration_ms": 3859
  }
]
koan@tux:~$
```

Figure 7: After logging all prompts and responses, you can continue conversations with the `llm` command.

Conversations

Thus far, you've posed questions and received responses from `llm` without engaging in a back-and-forth conversation. `llm` does support conversations (taking into account the context), although you need to enable logging of all prompts and responses first. Therefore, create a database with:

```
llm init-db
```

Once logging is enabled, `llm` will record the prompts and responses. You can view these logs with:

```
llm logs
```

This command displays the three most recent logs as a JSON array. However, the full output can be long, especially when piping output from commands to the `llm` command. You can truncate the display with (Figure 7):

```
llm logs --truncate
```

To continue the previous conversation, just add a prompt and include the `--continue` option:

```
llm --continue "How can this
  code still be improved?"
```

When you execute another `llm` command without the `--continue` argument, a new conversation starts. If you want to continue a specific conversation, look up its `chat_id` property in the output of `llm logs` and specify it as follows:

```
llm --chat 2 "How do I add support
  for another encrypted device?"
```

To display more than three conversations, add the `--count` argument:

```
llm logs --count 10 --truncate
```

Creating Prompt Templates

If you frequently use the same prompt, you can create prompt templates for `llm`. For example, suppose you want to create a prompt template to summarize texts. Create a template called *summary* with:

```
llm templates edit summary
```

This command opens your default editor with the template file. The initial content is as follows:

```
prompt:
```

This is a YAML [9] file with `prompt` as a key, and you now need to add your prompt as a YAML string value for this key. You use the `$input` template variable as a placeholder for the input text:

```
prompt: 'Summarize this: $input'
```

After saving the file, you can use the template on a text piped to the `llm` command:

```
cat text_to_summarize.txt | llm --template summary
```

Under the hood, `llm` combines the input piped into the command with the prompt from the template and processes it.

Other Options for Prompt Templates

Instead of a user prompt, you can also set a system prompt in a prompt template. For example, create a template to describe Git diffs:

```
llm templates edit describe-diff
```

And then let it use this system prompt:

```
system: Give a high-level description of the changes ⮐
  in this Git diff output
```

To use this template, run the following command:

```
git show 5b511e9 | llm --template describe-diff
```

You can combine the templates with other `llm` command options, such as the model:

```
git show 5b511e9 | llm --template describe-diff --model gpt4
```

However, you can also set a default model for a template:

```
model: gpt4
system: Give a high-level description of the changes ⮐
in this Git diff output
```

Additionally, you can use both a system prompt and a user prompt in a template.

For prompts that span multiple lines, use YAML multi-line strings. For example:

```
system: |
    Give a high-level description of the changes in this ⮐
      Git diff output.
    For each file, explain the changes in this file, ⮐
      and the overall goal.
    End with an overall explanation of all the changes ⮐
      across all files.
```

If you don't need to preserve newlines exactly, use `system: >` instead. This collapses indented text into a single string with newlines transformed into spaces.

Using Additional Template Variables

In addition to the `$input` variable that represents the user's input, you can define additional variables in a template, which you then need to provide to the `llm` command as extra parameters. For example, suppose you want to create a template to summarize a text with a specific word count and tone:

```
llm templates edit summary
```

The template looks like this:

```
system: Summarize this text in $words words and in a ⮐
  $tone tone.
```

To summarize a text using this template, run the following command:

```
cat text_to_summarize.txt | llm --template summary ⮐
  -p words 100 -p tone formal
```

You can modify the variables on the command line to generate longer or shorter summaries or change the tone.

Conclusion

Linux users are just beginning to consider how to integrate ChatGPT into their daily life. The ChatWizard desktop tool is a great way to get started, and it adds some convenient features for managing prompts and conversations. The text-based tool `llm` brings the power of the Linux command line to AI chat, letting you to pipe output of other commands directly to ChatGPT. ■■■

Info

[1] ChatGPT: *https://chat.openai.com*

[2] OpenAI: *https://openai.com*

[3] OpenAI platform: *https://platform.openai.com*

[4] OpenAI pricing: *https://openai.com/pricing*

[5] ChatWizard: *https://github.com/lisiur/ChatWizard*

[6] ChatWizard releases: *https://github.com/lisiur/ChatWizard/releases/*

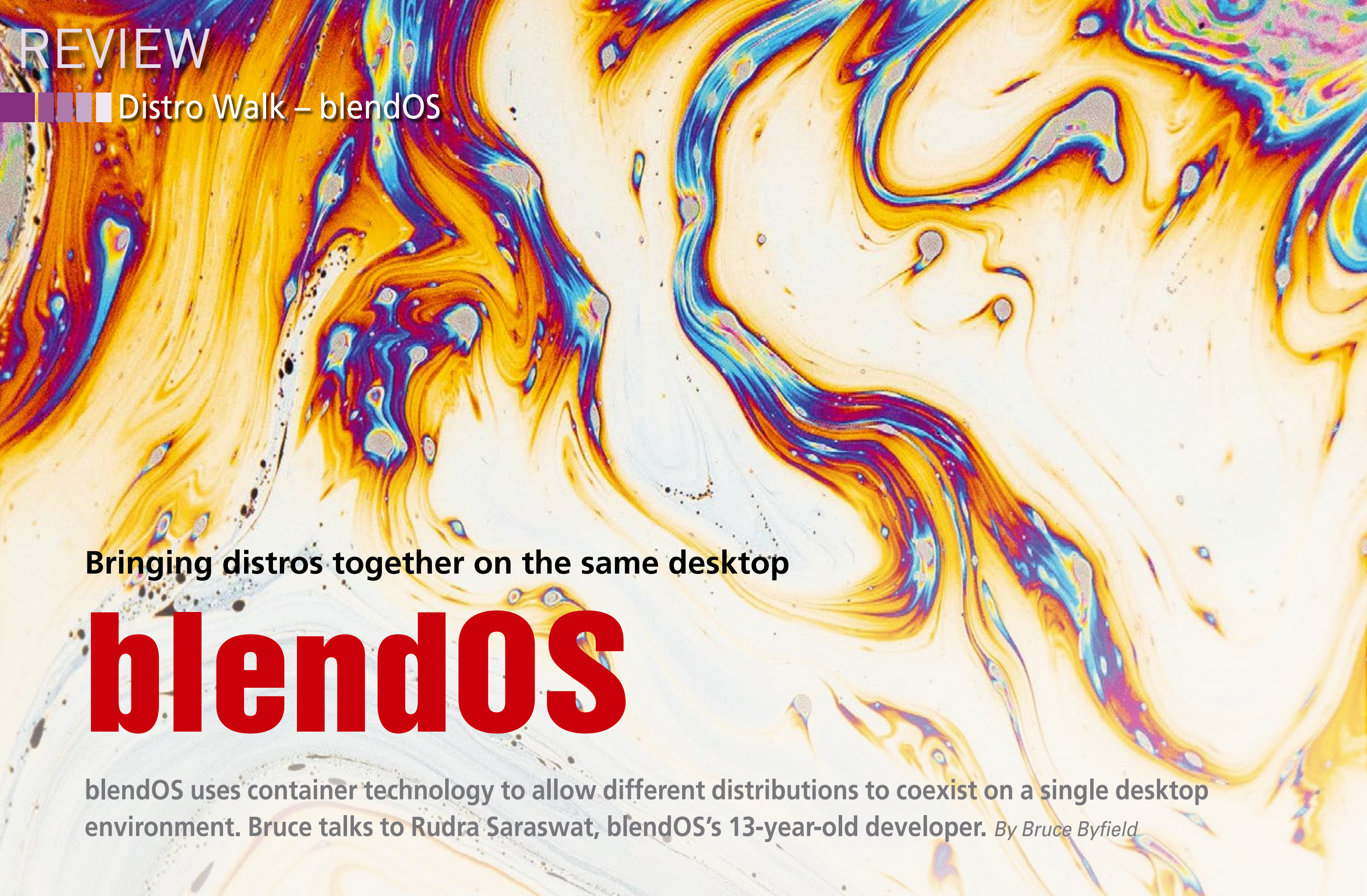
[7] Awesome ChatGPT Prompts: *https://prompts.chat*

[8] Simon Willison's llm: *https://llm.datasette.io*

[9] YAML: *https://yaml.org*

Author

Koen Vervloesem has been writing about Linux and open source, computer security, privacy, programming, artificial intelligence, and the Internet of Things for more than 20 years. You can find more on his website at *koen.vervloesem.eu*.



Bringing distros together on the same desktop

blendOS

blendOS uses container technology to allow different distributions to coexist on a single desktop environment. Bruce talks to Rudra Saraswat, blendOS's 13-year-old developer. *By Bruce Byfield*

Although several hundred Linux distributions already exist, new technologies and purposes are constantly spurring the creation of more. One of the most ambitious new efforts to emerge recently is blendOS [1], which uses container technology to allow packages from different distributions, as well as Android, to coexist in a single desktop environment (Figure 1). The result is ideal both for developers who package for different distributions and for everyday users who want the latest releases.

As I write, blendOS is little more than a year old, with the latest version released in May 2023. Based on Arch Linux, blendOS currently supports Debian, Ubuntu, Fedora, Arch Linux, Kali Linux, AlmaLinux, and Rocky Linux, with the potential to support even more distributions. blendOS also features a minimalist install (Figure 2), graphical tools to create containers and connect to Android stores, and clear and comprehensive online documentation [2].

blendOS was created by Rudra Saraswat (Figure 3), a 13-year-old developer who is also involved in the continued development of Unity, the desktop discontinued by Ubuntu, as well as its port to Arch Linux, plus Gamebuntu, Ubuntu Web, Ubuntu Remixes, UbuntuEd, an alternative Snap store called lol, and una, a version of Arch's AUR for Ubuntu and Debian. In many ways, blendOS can be seen as a natural development of Saraswat's other efforts at cross-distribution development. Fortunately, Saraswat was finishing exams when contacted, so he had time to briefly answer questions.

Linux Magazine (LM): Tell us about your involvement with free software and your experiences as a young contributor.

Rudra Saraswat (RS): I have been quite active in the free software community since about 2019 when I was nine, working on a bunch of tools/shell scripts and posting them on the Ubuntu Community Discourse. I did work on a few tiny operating system kernels prior to that back in 2017/2018 and made them public on GitHub, but as expected, they didn't gain much traction, so I wouldn't term that as my debut. Ubuntu Unity, which I released when I was 10, was my first project of that sort that got popular, and it's now an official Ubuntu flavor.

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/*.

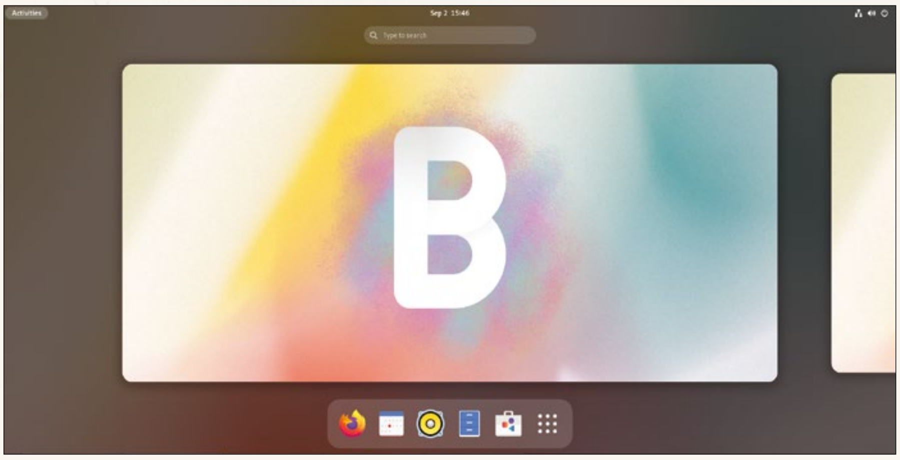
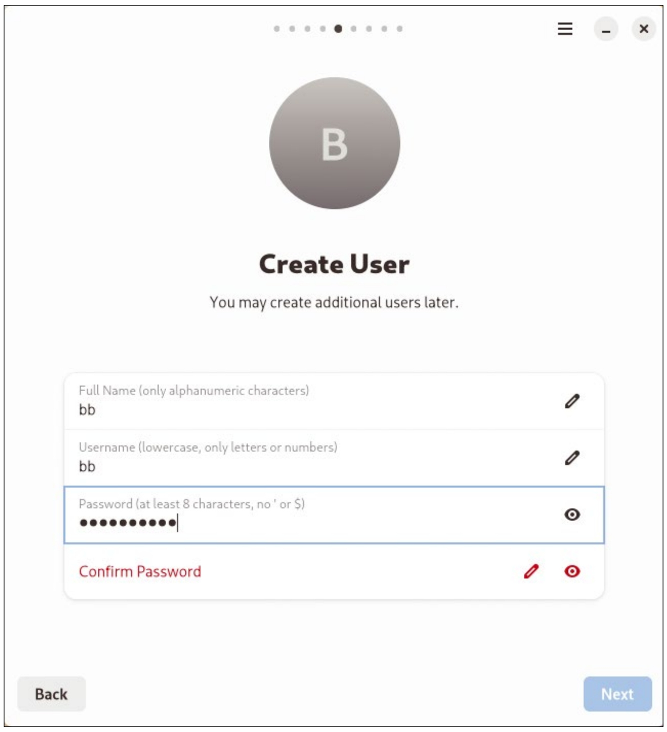


Figure 1: blendOS's appropriately themed default desktop.

Photo by Daniel Olah on Unsplash



Create User

You may create additional users later.

Full Name (only alphanumeric characters)
bb

Username (lowercase, only letters or numbers)
bb

Password (at least 8 characters, no ' or $)
••••••••••

Confirm Password

Back
Next

Figure 2: blendOS's minimalist installer.

It was quite challenging initially, with yours truly getting kicked off quite a few Linux communities (including some local Linux user groups) due to concerns related to my young age, but I eventually found a safe haven in the official Ubuntu communities.

LM: How do you make decisions within the blendOS project? How popular is blendOS?

RS: Decisions are usually based off of the general consensus of the blendOS community. Speaking of which, blendOS has exploded in popularity and is growing at a rapid pace.

LM: What is blendOS's main goal?

RS: blendOS aims to ship a stable, up-to-date base system that allows you to mix-and-match applications from different package repositories and distributions, without the usual risks of dependency hell that come with it, thanks to their containerization. Any applications you install in a container (from a shell inside it) instantly appear on the host (binaries with a suffix consisting of a dot and the name of the container) (Figure 4), with Android apps being supported as well through Waydroid (Figure 5).

LM: What are the technical and structural challenges in supporting so many distributions?

RS: There aren't many challenges supporting this many distributions, since quite a few share package managers and similar package names. I plan to make it possible for one to use any distribution outside of the supported ones with a little bit of tweaking if one has the know-how.

A container allows you to install regular packages from any of the Linux distributions and use them on the host as though they were native packages, allowing for a seamless experience.

There isn't any difference in memory consumption or CPU utilization, since containers are essentially equivalent to jailed (namespaced) chroots. As a result, they're identical in this regard.

LM: What is the relation of blendOS to Arch Linux?

RS: Indeed, blendOS ships Arch as the base underneath, images of which are rolled out as updates once we ensure they're perfectly stable and won't result in system breakage, thus eliminating the greatest disadvantage of rolling release distributions: stability.

LM: How is blendOS structured?

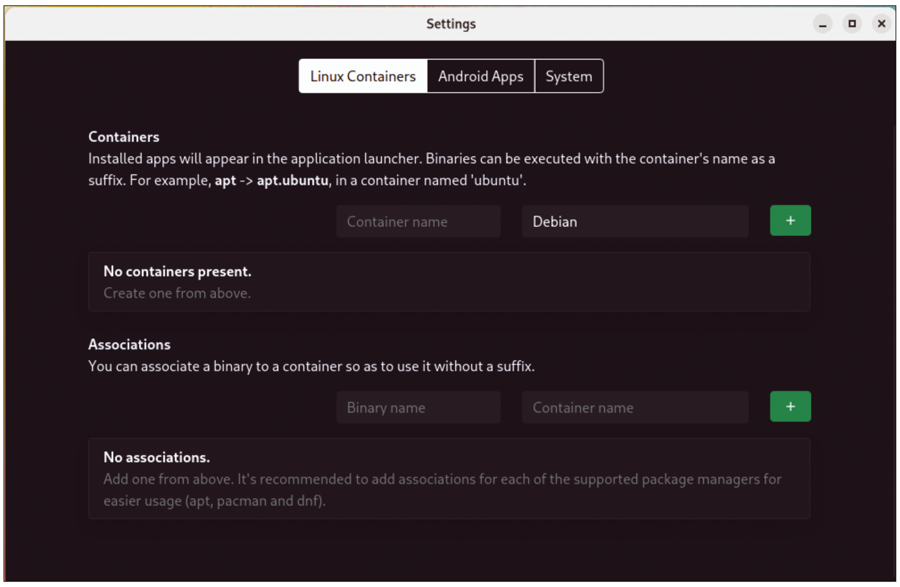
RS: On blendOS, a base system with a particular desktop environment is called a track, and you can therefore switch between desktop environments simply by switching your system track with the command `sudo system track` (Figure 6).

Cadres help with migrating or reproducing blendOS container setups on multiple systems and thus eliminate the effort required to set up all of your containers once again.

LM: blendOS installs with two Android sites that carry free software. Tell us more about blendOS's Android support.

RS: You can install any Android Package Kit (APK) file (not to be confused with Alpine Linux's package format) simply by double-clicking on it,

Figure 3: **At 13, Saraswat is already a veteran contributor to Linux.**



Settings

Linux Containers | Android Apps | System

Containers
Installed apps will appear in the application launcher. Binaries can be executed with the container's name as a suffix. For example, **apt** -> **apt.ubuntu**, in a container named 'ubuntu'.

Container name | Debian | +

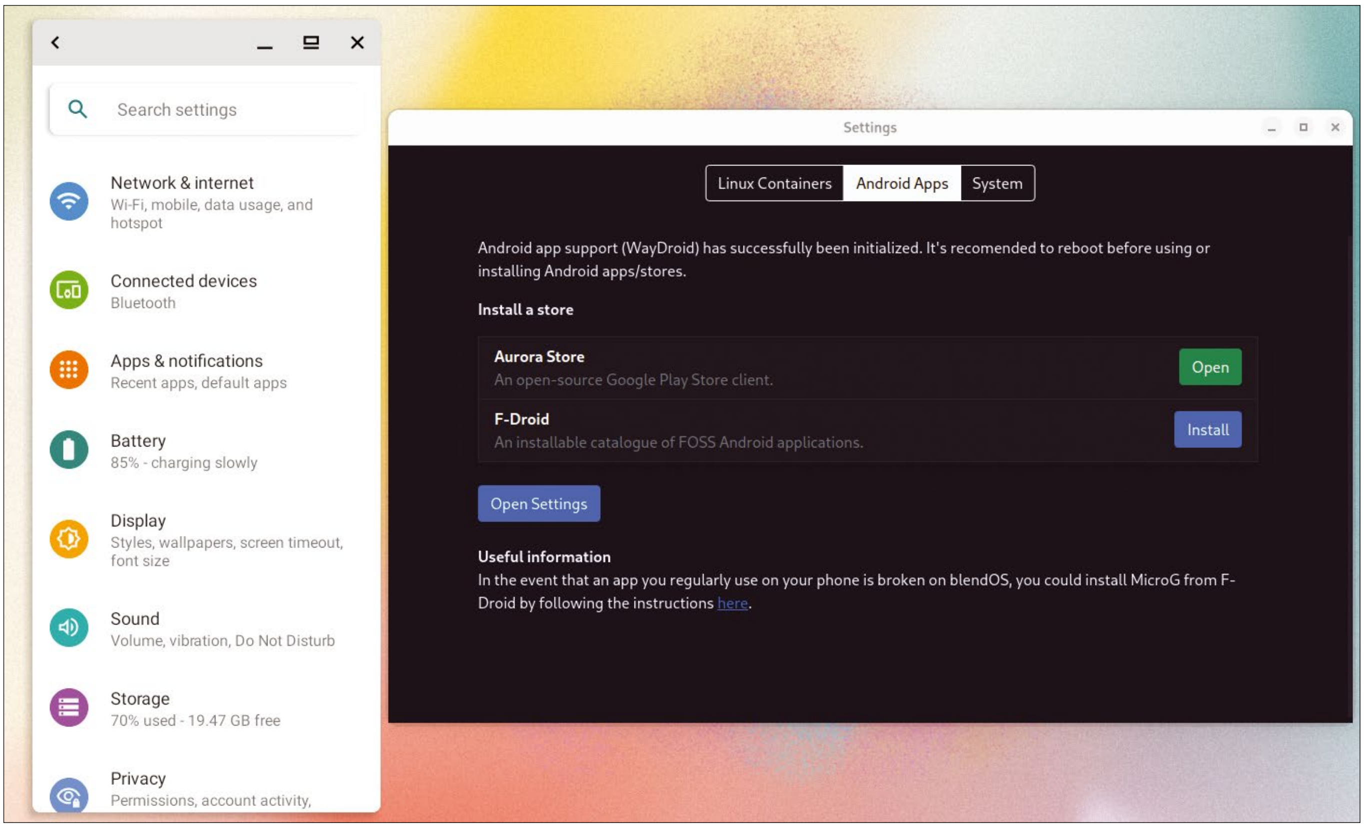
No containers present.
Create one from above.

Associations
You can associate a binary to a container so as to use it without a suffix.

Binary name | Container name | +

No associations.
Add one from above. It's recommended to add associations for each of the supported package managers for easier usage (apt, pacman and dnf).

Figure 4: **blendOS's graphical tool for making containers, including one for each distribution from which packages are taken.**



Android app support (WayDroid) has successfully been initialized. It's recomended to reboot before using or installing Android apps/stores.

Install a store

Aurora Store
An open-source Google Play Store client.
Open

F-Droid
An installable catalogue of FOSS Android applications.
Install

Open Settings

Useful information
In the event that an app you regularly use on your phone is broken on blendOS, you could install MicroG from F-Droid by following the instructions here.

Figure 5: blendOS shows two sites for free software Android apps, but others can be added, even proprietary ones.

therefore allowing you to install any store on blendOS – even the Google Play Store with a couple of hoops, though we personally wouldn't recommend it, since the Aurora Store ships all of the apps available through the Play Store while respecting your privacy.

An example of a useful Android app is WhatsApp, with which you can simply link with your mobile (as the Android app subsystem on blendOS is detected as a tablet). The overall experience is much better than that of WhatsApp Web.

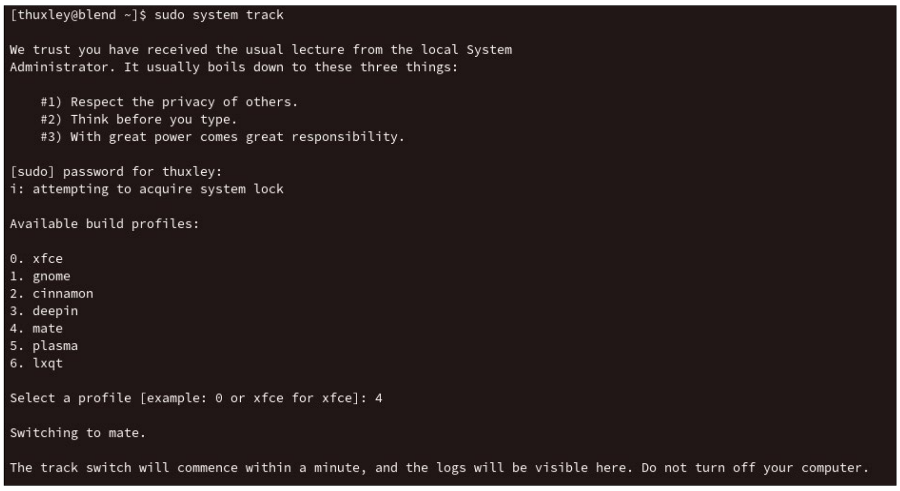
Thanks to the advent of the Waydroid project back in 2020, I do expect this to be the case. One of my earlier projects from back then, Ubuntu Web, also used WayDroid for Android app support, but it's probably going to be a gradual shift due to the influence of desktop power users.

LM: What are your future plans for blendOS?

RS: I'm planning to add some major new features and editions for the next major release of blendOS (rolled out as an update to version 3) as well. For now, all I'd like to state is that we're still accepting feedback from the community and will continue to do so. We'll continually strive to make blendOS even better in the coming releases.

Conclusion

In case anyone doubts the ambition behind the project, BlendOS's web page advertises itself as "The only operating system you'll ever need." That is possibly an exaggeration, especially when you consider the amount of time that might be needed to keep multiple distributions private and secure. Still, blendOS remains an impressive effort, especially in its innovative use of existing technology. There is no doubt that the interest exists; currently, blendOS is at number 32 on the DistroWatch's list of top page hits, despite its brief history. Without a doubt, blendOS is a distribution to watch. ■■■


```
[thuxley@blend ~]$ sudo system track

We trust you have received the usual lecture from the local System
Administrator. It usually boils down to these three things:

    #1) Respect the privacy of others.
    #2) Think before you type.
    #3) With great power comes great responsibility.

[sudo] password for thuxley:
i: attempting to acquire system lock

Available build profiles:

0. xfce
1. gnome
2. cinnamon
3. deepin
4. mate
5. plasma
6. lxqt

Select a profile [example: 0 or xfce for xfce]: 4

Switching to mate.

The track switch will commence within a minute, and the logs will be visible here. Do not turn off your computer.
```

Figure 6: blendOS's tool for switching desktops without logging out.

Info

[1] blendOS: *https://blendOS.co/*

[2] blendOS documentation: *https://docs.blendOS.co/guides/installation-guide/*

DEVELOPERWEEK™

2024
2024
2024

Feb 21-23 SF Bay Area **Feb 27-29** Live Online

8,000+ Developers at the World's Largest Developer Expo Conference Series

Use code **MP408** for **$100** off all **DeveloperWeek 2024** tickets

Register at **developerweek.com**

250+ Speakers across 15+ technical tracks including:

JavaScript · AI · Containers & Kubernetes · APIs
Microservices · Python · Databases · Dev Tools Dev
Management/Leadership · Blockchain
Product Teams · ML · Serverless · DevOps

OUR NETWORK OF EVENTS

5 events. **4,000+** companies. **15,000+** annual attendees.

DEVELOPERWEEK
February 21–29, 2024

DEVELOPERWEEK EUROPE
April, 2024

DEVELOPERWEEK LATIN AMERICA
June, 2024

DEVELOPERWEEK MANAGEMENT
May, 2024

CLOUDX™
August, 2024

DEVELOPERWEEK ENTERPRISE
November, 2024

Using a Raspberry Pi to put a legacy printer on the network

New Life

Niche hardware from the olden days does not always embrace the network. Attaching a Raspberry Pi or other single board computer can add lots of new functionality. *By Christopher Dock*

I inherited a DYMO 400 label printer that has a simple USB connection, but it has recently lost the Windows support lottery. My goal was to convert this old standalone label printer (Figure 1) into a network label printer that all the computers on my network can access. Linux users often chafe at the need to replace hardware that is still functioning perfectly well, so keeping an old printer going is a problem that many of us have faced in the past. In this case, it actually appears that the old version of the printer might function *better* than the newer model (see the box entitled "DRM Woes"), so I'd prefer to keep using it.

The Linux ecosystem is a rich and mature environment with all sorts of

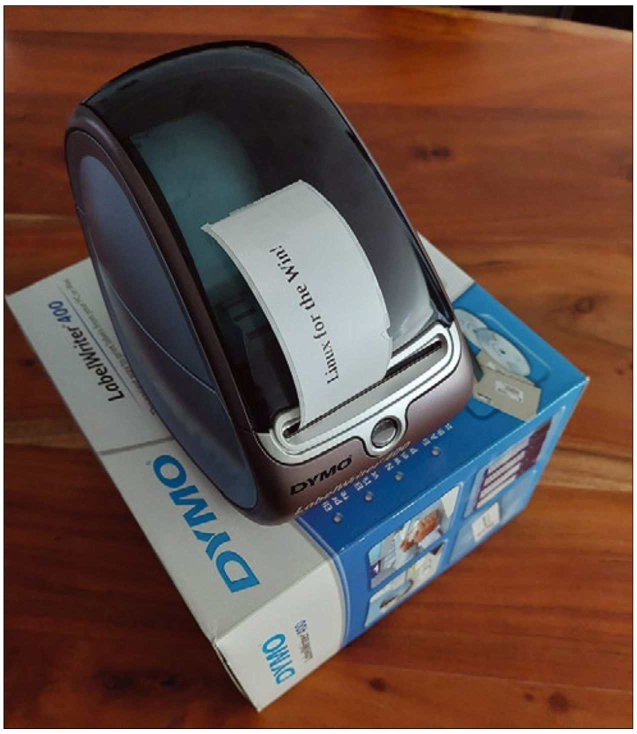


Figure 1: **The DYMO 400 label printer.**

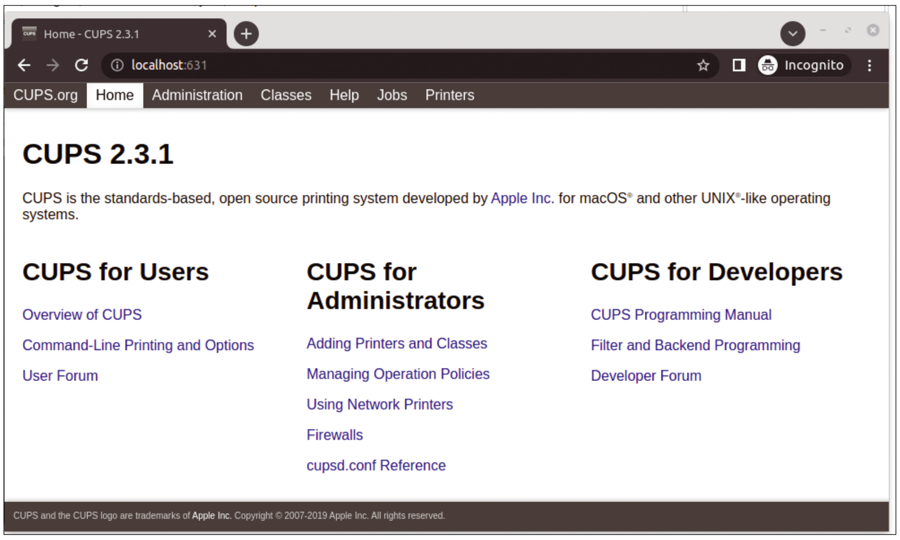
DRM Woes

The DYMO company is still in business and sells a newer version of this printer, but the new version is less of a bargain than their older model. The older DYMO 400 and 450 can accept both DYMO official labels as well as generic labels. The newer 550 model has been engineered with digital rights management, preventing users from printing generic labels. This is apparently by design, according to their documentation [1]:

"The LabelWriter 550 series of printers work only with Authentic DYMO LW labels. Non-authentic DYMO labels will not be recognized by the printers."

This seems to be a rather shabby trick to play on their customers, but electronics is not magic, and solutions to most problems do exist if you are willing to invest enough time. One such solution exists on GitHub [2] to allow generic labels to be used with the DYMO 550; it was developed within a few months of the DYMO 550 being launched.

This solution is a I2C device to deal with the DRM recognition. Although this might be a great option for the technical do-it-yourselfer, most business people will not purchase a microcontroller, compile the code, flash the microcontroller, and solder it to their printer. Keep in mind, also, that hacking your way around DRM is illegal in some jurisdictions. Better to keep the old hardware going if you have the option.



CUPS 2.3.1

CUPS is the standards-based, open source printing system developed by Apple Inc. for macOS® and other UNIX®-like operating systems.

CUPS for Users: Overview of CUPS, Command-Line Printing and Options, User Forum

CUPS for Administrators: Adding Printers and Classes, Managing Operation Policies, Using Network Printers, Firewalls, cupsd.conf Reference

CUPS for Developers: CUPS Programming Manual, Filter and Backend Programming, Developer Forum

CUPS and the CUPS logo are trademarks of Apple Inc. Copyright © 2007-2019 Apple Inc. All rights reserved.

Figure 2: Access CUPS through a web interface.

programs and utilities that cover almost any situation, but back in the 1990s, printing support in Linux was fairly limited due to the large number of standards and protocols and the lack of Linux-specific drivers. In 1999, Easy Software Products saw this defect/opportunity and created the Common UNIX Printing System (CUPS). The CUPS solution basically turns the computer running it into a print server.

More importantly, CUPS provides a single standard that can handle many printers and protocols. CUPS was released in 1999 and was adopted by many Linux distributions. The CUPS printer service was quite successful for its time, but it could only support printers that had CUPS drivers, as well as the few printers with native support for Postscript.

In 2002, Apple was working on OS X, and, like every other operating system, it needed printing support. Apple decided to buy rather than build. Apple first licensed CUPS from Easy Software Products in 2002, but, after a few short years, they purchased the CUPS source code [3] and hired the lead developer Michael Sweet. CUPS is still used by Apple to support printing on all of their operating systems.

Apple owning CUPS might seem like a problem for the open source community, but that was not the case. Early Linux printing was difficult due to the number of different standards and protocols, but more importantly, Linux did not have a very large install base. Apple computers provided too large of a market segment to ignore, so printer companies started to provide printer drivers for Mac OS X. These drivers did not directly support Linux, but there was a great workaround. The workaround was to download the Mac driver and extract the PPD file and then test it to see if it worked for your specific printer. In many cases, this approach seemed to work. Since then, it has become much easier to find the PPD files for many popular printers [4].

Listing 1: lpinfo

```
$ lpinfo -v
direct usb://DYMO/LabelWriter%20400?serial=07000619355405
file cups-brf:/
network beh
network ipp
direct vnc:/
network socket
network ipps
direct hp
network http
network lpd
network https
direct hpfax
network dnssd://HP%20Color%20LaserJet%20Pro%20M478f-9f%20
                %5BD7422C%5D._ipp._tcp.local/?uuid=b69de7ee-
                8055-5cc9-c1d0-526e9f44a5a5
network ipps://HP%20Color%20LaserJet%20Pro%20M478f-9f%20
               %5BD7422C%5D._ipps._tcp.local/
network socket://192.168.178.43:9100
```

No relationship is perfect, and the hundreds of improvements that Apple has committed to Git have slowed down to a trickle since 2020 [5]. CUPS was no longer progressing. The CUPS code was eventually forked and is now maintained by the Linux Foundation's OpenPrinting project, which serves as the upstream source for Apple's CUPS [6] .

Raspberry Pi Networking

One of the really forward-thinking design decisions built into CUPS is that it uses a web browser for the user interface (Figure 2). A browser interface makes it easier for CUPS to run on many different platforms big or small, and that means the Raspberry Pi is just as capable as a full-sized server.

Coupling a DYMO label printer with a Raspberry Pi will make it a true network printer. I will also be able to add a queue to buffer if too many labels are printed at once. CUPS does all the hard work; it is just a matter of installing and configuring everything. Setup only takes a few minutes.

The first step might not even be necessary, depending on which Linux distribution you are using. It was not necessary to install CUPS on either Mint or on Raspberry Pi OS, but for a Debian-based distribution the command is:

```
sudo apt-get update && ⮐
  sudo apt-get install cups ⮐
  cups-client
```

In systemd

If CUPS was not previously installed, you will probably want to configure CUPS to start when the computer is started. Raspberry Pi OS uses systemd, so the `systemctl` command is used to manage processes.

To enable CUPS at startup:

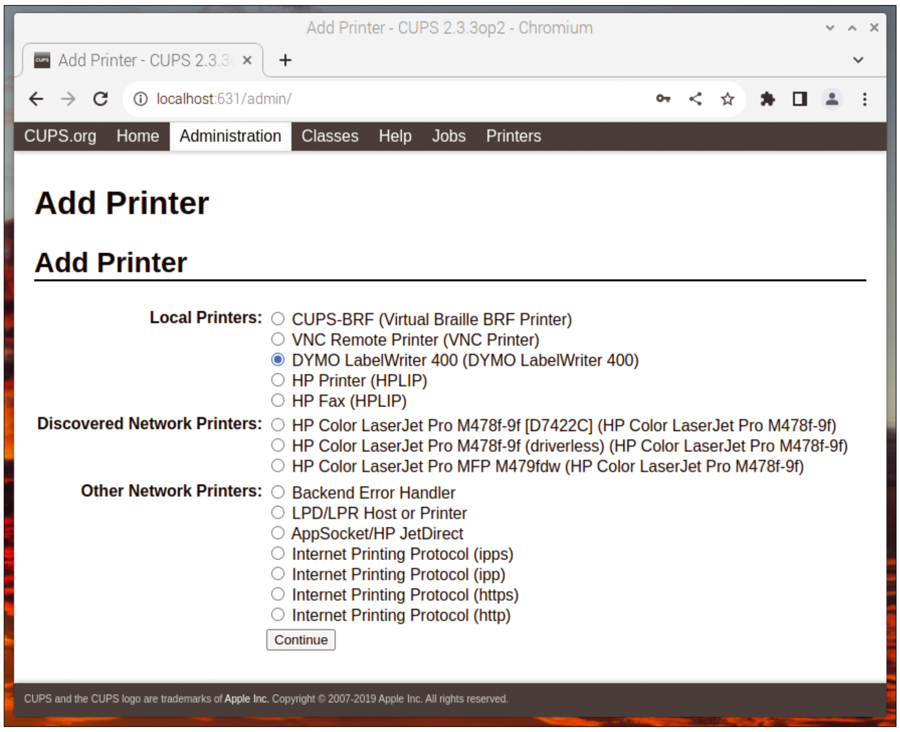
```
sudo systemctl enable cups
```

To restart CUPS:

```
systemctl restart cups.service
```

To stop or start CUPS:

```
systemctl stop cups.service
systemctl start cups.service
```


Add Printer - CUPS 2.3.3op2 - Chromium

localhost:631/admin/

CUPS.org Home Administration Classes Help Jobs Printers

Add Printer

Add Printer

Local Printers:
- ○ CUPS-BRF (Virtual Braille BRF Printer)
- ○ VNC Remote Printer (VNC Printer)
- [✓ DYMO LabelWriter 400 (DYMO LabelWriter 400)]
- ○ HP Printer (HPLIP)
- ○ HP Fax (HPLIP)

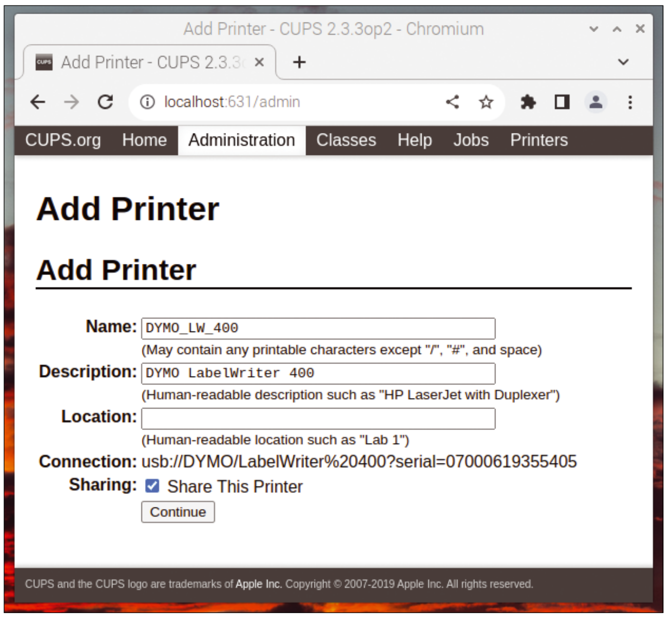
Discovered Network Printers:
- ○ HP Color LaserJet Pro M478f-9f [D7422C] (HP Color LaserJet Pro M478f-9f)
- ○ HP Color LaserJet Pro M478f-9f (driverless) (HP Color LaserJet Pro M478f-9f)
- ○ HP Color LaserJet Pro MFP M479fdw (HP Color LaserJet Pro M478f-9f)

Other Network Printers:
- ○ Backend Error Handler
- ○ LPD/LPR Host or Printer
- ○ AppSocket/HP JetDirect
- ○ Internet Printing Protocol (ipps)
- ○ Internet Printing Protocol (ipp)
- ○ Internet Printing Protocol (https)
- ○ Internet Printing Protocol (http)

Continue

CUPS and the CUPS logo are trademarks of Apple Inc. Copyright © 2007-2019 Apple Inc. All rights reserved.

Figure 3: List of local and discovered printers.


Add Printer - CUPS 2.3.3op2 - Chromium

localhost:631/admin

CUPS.org Home Administration Classes Help Jobs Printers

Add Printer

Add Printer

Name: DYMO_LW_400
(May contain any printable characters except "/", "#", and space)
Description: DYMO LabelWriter 400
(Human-readable description such as "HP LaserJet with Duplexer")
Location:
(Human-readable location such as "Lab 1")
Connection: usb://DYMO/LabelWriter%20400?serial=07000619355405
Sharing: [✓ Share This Printer]

Continue

CUPS and the CUPS logo are trademarks of Apple Inc. Copyright © 2007-2019 Apple Inc. All rights reserved.

Figure 4: Naming the queue and describing the printer.

If CUPS was not installed previously, you will also probably want to set up CUPS to start automatically when the system starts (see the sidebar entitled "In systemd"). Once CUPS is installed and the DYMO label printer is attached, you should verify that the operating system recognizes it. You can use the `lsusb` command to verify, or you can skip directly to the CUPS `lpinfo` command to display what devices are available. In my case, the `lpinfo` command displays both the DYMO LabelWriter 400 and my HP printer (Listing 1).

You can install the DYMO printer while installing the CUPS software. But if you are feeling cautious, you can do each step separately.

```
sudo apt-get install ↩
printer-driver-dymo
```

This DYMO package does not actually add any drivers or programs but instead supplies a lot of defined page formats, which correspond to the labels that you can feed to the printer.

Configure the Printer

Setting up a DYMO printer from the CUPS menu (as shown in Figure 3) doesn't require very much user interaction. From the menu, click on the *Administration* menu and then click on the *Add Printer* button. This button will display a list of local and discovered printers; simply select the DYMO from the list.

Clicking on the *Continue* button will bring up a dialog (Figure 4), allowing you to enter the name of the printer. Initially the printer queue name was `DYMO_LabelWriter_400`, which I shortened to `DYMO_LW_400`. This is just the local name that you can use to print to from the command prompt. During the setup, selecting the *Share This Printer* checkbox did not quite have the effect I was looking for (more on this later). Clicking on the *Continue* button will bring up another dialog (Figure 5), which allows you to select your printer model. These three dialogs are actually associating the printer that was found by CUPS with a user-provided name and model number.

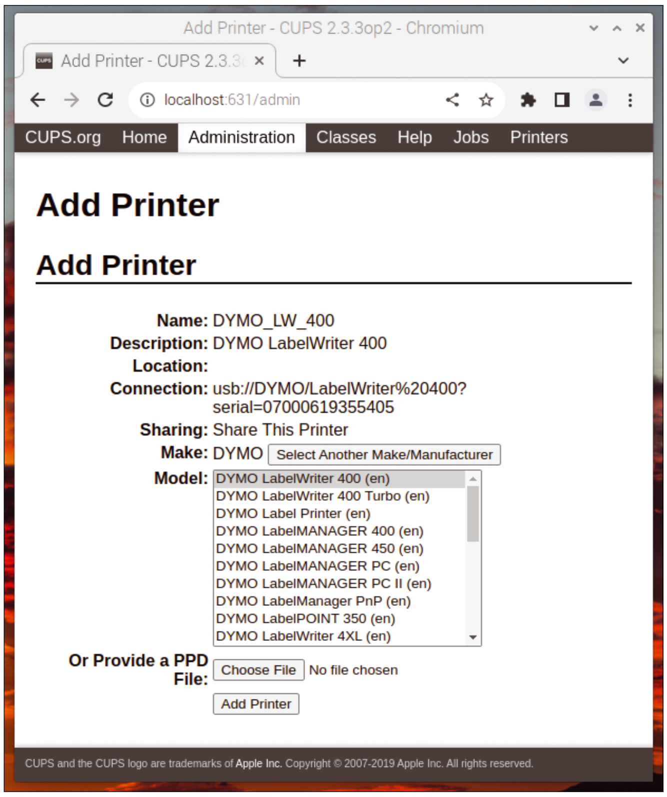
Click on the *Add Printer* button to create the printer configuration and bring up a final dialog. This final dialog (Figure 6) defines a few settings, with the most important being the label size. My DYMO printer has an address label #99012, which is 89x36mm. This size seems a bit odd for a metric measurement but the size doesn't really look much better when converted to Imperial measurements: 3.5x1.4 inches. Pressing *Set Default Options* brings you back to the menu.

If you wish to verify that the printer was set up correctly, you can click on the *Printers* tab of the menu, but a more gratifying test would be to try to print a label. Use the `lpstat` command to see all available print queues (Listing 2).

Initially, when I set up this printer, I was hoping that I could print out

Listing 2: Checking Available Print Queues

```
pi@smartframe:~ $ lpstat -v
device for DYMO_LW_400: usb://DYMO/LabelWriter%20400?serial=07000619355405
device for HP_Color_LaserJet_Pro_M478f_9f_D7422C_: implicitclass://HP_Color_LaserJet_Pro_M478f_9f_D7422C_/
```


Add Printer - CUPS 2.3.3op2 - Chromium

localhost:631/admin

CUPS.org Home Administration Classes Help Jobs Printers

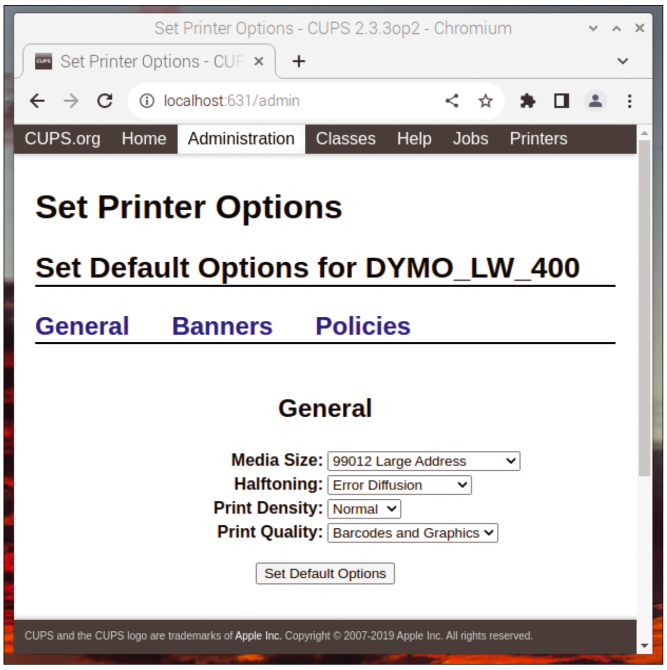
Add Printer

Add Printer

Name: DYMO_LW_400
Description: DYMO LabelWriter 400
Location:
Connection: usb://DYMO/LabelWriter%20400?serial=07000619355405
Sharing: Share This Printer
Make: DYMO [Select Another Make/Manufacturer]
Model: DYMO LabelWriter 400 (en)
DYMO LabelWriter 400 Turbo (en)
DYMO Label Printer (en)
DYMO LabelMANAGER 400 (en)
DYMO LabelMANAGER 450 (en)
DYMO LabelMANAGER PC (en)
DYMO LabelMANAGER PC II (en)
DYMO LabelManager PnP (en)
DYMO LabelPOINT 350 (en)
DYMO LabelWriter 4XL (en)
Or Provide a PPD File: [Choose File] No file chosen
[Add Printer]

CUPS and the CUPS logo are trademarks of Apple Inc. Copyright © 2007-2019 Apple Inc. All rights reserved.

Figure 5: Assigning the printer model of the previously named new printer.


Set Printer Options - CUPS 2.3.3op2 - Chromium

localhost:631/admin

CUPS.org Home Administration Classes Help Jobs Printers

Set Printer Options

Set Default Options for DYMO_LW_400

General Banners Policies

General

Media Size: 99012 Large Address
Halftoning: Error Diffusion
Print Density: Normal
Print Quality: Barcodes and Graphics

[Set Default Options]

CUPS and the CUPS logo are trademarks of Apple Inc. Copyright © 2007-2019 Apple Inc. All rights reserved.

Figure 6: Setting printer defaults.

Internet postage from the German Post Office. It is possible to get a sample label from the German Post Office website [7], which does support the DYMO 99012 label as a PDF. I created a few test labels, which you can download from the Linux Magazine server [8].

The lp command lets me print a label on the newly defined DYMO_LW_400 printer queue from the command prompt:

```
lp -d DYMO_LW_400 testlabel.pdf
```

A more convenient testing method might be to create a document in LibreOffice (Figure 7). The first time I tried to print from LibreOffice, the page size was incorrect. To fix this problem, I selected *Use Printer Properties* and checked *Use only paper size from printer preferences*.

Listing 3: Configuration File

```
# Restrict access to the server to
  local network
<Location />
  Order allow,deny
  Allow from @LOCAL
</Location>
```

If the task was just setting up the printer locally, the process is complete. However, with just a few more small changes, you can make the printer available to the entire network. All of the configuration is in the `cupsd.conf` file, which is usually located in the `/etc/cups` directory.

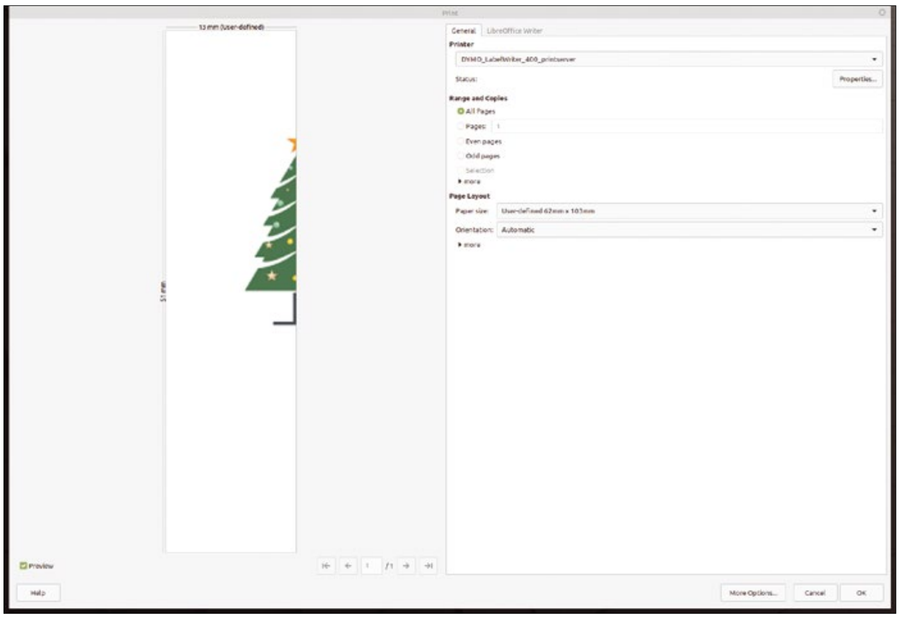
CUPS listens only to the localhost on a port for print jobs but can be configured to accept jobs from the entire network. The configuration file has a line that defines a port number for listening on the network:

```
Listen localhost:631
```

This new configuration will listen on port 631 and essentially accept any IP address that can reach the printer server.

In addition to enabling network printing, it is possible to enable remote configuration and remote admin support. Each of these tasks has a block in the configuration file (Listing 3).

There are situations where allowing full access to the printer is inadvisable. In those cases, it is possible to limit access to a specific subnet. Adding a


Print

General LibreOffice Writer

Printer
DYMO_LabelWriter_400_printserver
Status: [Properties...]

Range and Copies
All Pages
Pages: 1
Even pages
Odd pages
Selection
more

Page Layout
Paper size: User-defined 62mm x 103mm
Orientation: Automatic
more

[Help] [More Options...] [Cancel] [OK]

Figure 7: Printing from LibreOffice.

Listing 4: Restricting Access

```
# Restrict access to the server...
<Location />
  Order allow,deny
  Allow from 192.168.178.*
</Location>
```

partial subnet will limit which computers can access the printer (Listing 4).

I planned to administer the printer remotely, so I added the "Allow from" statement to both the `/admin` and `/admin/conf` location blocks. Once I added these small changes, I had to restart the CUPS daemon. If there are any errors in the configuration file, the restart will not work and might display a cryptic message.

Once everything is correctly set up for your network and CUPS has restarted, you can use the `lpstat -v` command to see if your printer is visible to CUPS (Listing 5).

Direct Printing

When I first decided to use this label printer, it was with the goal of printing Internet postage from the German post office, but I also considered using it for creating and printing out address labels. I thought about writing a small program to print the labels but actually hoped someone might have solved this problem before me. Despite the number of DYMO projects on GitHub, I could not find one that solved my exact problem, even though there were quite a few DYMO label printer projects.

One interesting project [9] was short and contained a fairly concise example of printing complex labels. The label it produces was tailored by its author for a very specific case, but the code did help to illustrate the general Java programming techniques to output to a printer while using different fonts. The effort to create a Java program to print 5 or 6 line labels seemed a bit excessive, considering that I could easily use LibreOffice to create and print labels.

Summary

The initial goal was to make the label printer available on the network to other computers, and this worked out great. It was possible to share the DYMO printer from a Raspberry Pi, making it a true network printer. Raspberry Pis are both difficult to purchase and expensive at the moment, so I ended up attaching the printer to my PC.

Another of my hopes was that the solution would be flexible enough to allow a Windows PC to print to this label printer. Windows did indeed allow me to add the printer, but there were difficulties in defining the label size. In addition, all my labels generated from the Windows machine were stuck in a printer queue.

There is a possibility of sharing Linux printers to Windows using both Samba and CUPS together, but that seemed like too much effort. Should I ever need to print from Windows, I might have to revisit that problem. ■■■

Info

[1] DYMO 550 User Guide: *https://download.dymo.com/dymo/user-guides/LabelWriter/LW550Series/LW550_UserGuide.en.pdf*

[2] free-dmo: *https://github.com/free-dmo/free-dmo-stm32*

[3] CUPS acquisition: *https://www.macrumors.com/2007/07/12/apple-acquired-cups/*

[4] PPD Files: *https://www.openprinting.org/driver/Postscript*

[5] "Has Apple Abandoned CUPS?" by Tim Anderson, *The Register*, October 15, 2020: *https://www.theregister.com/2020/10/15/apple_cups_develoment/*

[6] OpenPrinting: *https://openprinting.github.io/achievements/*

[7] Deutsche Post: *https://www.deutschepost.de/en/home.html*

[8] Source code for this article: *https://linuxnewmedia.thegood.cloud/s/5Rzx9tQW2FJ6N3Z*

[9] dymolabelwriter: *https://github.com/simsam7/dymolabelwriter*

Author

Christopher Dock is a senior consultant for site services at T-Systems. When he is not working on integration projects, he likes to experiment with Arduino and Raspberry Pi. He is the author of *Getting Started with Arduino and Raspberry Pi*.

Listing 5: lpstat -v

```
chris@chrismint20:~$ lpstat -v
device for DYMO_LabelWriter_400_smartframe: implicitclass://DYMO_LabelWriter_400_smartframe/
device for HP-HP-Color-LaserJet-Pro-M478f-9f-new: hp:/net/HP_Color_LaserJet_Pro_M478f-9f?hostname=HPBCE92FD7422C.local
device for HP_Color_LaserJet_Pro_M478f_9f_D7422C_: implicitclass://HP_Color_LaserJet_Pro_M478f_9f_D7422C_/
chris@chrismint20:~$
```

@phptek

@tek@phparch.social

https://tek.phparch.com

CONNECT WITH THE COMMUNITY, LEVEL UP YOUR SKILLS, LEARN THE LATEST TRENDS, AND MAYBE EVEN FIND YOUR NEXT JOB.

RESERVE YOUR SPOT TODAY

PHP[TEK] 2024
16TH ANNUAL PHP[TEK] CONFERENCE FOCUSED ON PHP AND WEB DEVELOPMENT.

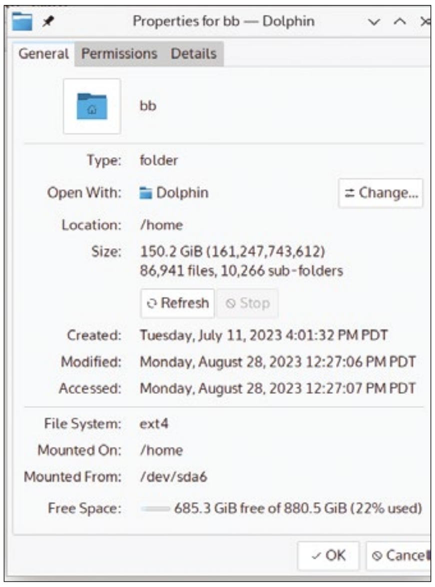
CFP IS CURRENTLY OPEN. IF YOU WOULD LIKE TO HELP SHAPE THE FUTURE OF PHP DEVELOPMENT AND WEB DEVELOPMENT, SUBMIT YOUR TALKS TODAY.
HTTPS://TEK.PHPARCH.COM/SPEAKERS

Editing and displaying advanced file attributes

Metadata Management

The chattr and lsattr commands offer users a convenient way to modify and display advanced file attributes. *By Bruce Byfield*

Most users are familiar with the usual file attributes. Desktop file managers often list them as properties: the name, path, size, and the dates that the file was created, last accessed, and last modified (Figure 1). Many, too, are familiar with the separate read, write, and execution permissions for a file's owner, group, and other users. However, over the decades, new filesystems, as well as the needs of version control for developers, have developed additional optional attributes. These additional attributes are edited by `chattr` [1], the equivalent for `chown` and `chmod` for permissions, and can be viewed by `lsattr` [2]. Both commands are included in the *e2fsprogs* [3] package and are installed by default in most distributions. While the creation of `chattr` and `lsattr` was driven by the needs of developers, many of these additional attributes are practical for everyday use as well.



Properties for bb — Dolphin

General | Permissions | Details

bb

Type:	folder
Open With:	Dolphin
Location:	/home
Size:	150.2 GiB (161,247,743,612) 86,941 files, 10,266 sub-folders
Created:	Tuesday, July 11, 2023 4:01:32 PM PDT
Modified:	Monday, August 28, 2023 12:27:06 PM PDT
Accessed:	Monday, August 28, 2023 12:27:07 PM PDT
File System:	ext4
Mounted On:	/home
Mounted From:	/dev/sda6
Free Space:	685.3 GiB free of 880.5 GiB (22% used)

Figure 1: The standard file attributes are displayed in Plasma's Dolphin file manager. Also note the Permissions tab.

Table 1: Operators

+	Adds the attributes that follow
-	Removes the attributes that follow
=	Makes the attributes that follow the file's only attributes

chattr and lsattr Options

The commands for working with file attributes have limited options. All options use a single letter rather than the full word like GNU style. In `chattr`, the `-f` option will suppress all except critical error messages. However, users are more likely to want `-R` (recursive) to change the attributes of an entire directory or `-V` (verbose) to receive instant feedback. Developers may want to use `-v VERSION` together with version control, or `-p PROJECT` to associate files with a particular project. For commands, `chattr` uses a standard structure:

```
chattr [OPTIONSP [-v VERSION] ➘
  [-p PROJECT] [OPERATOR][ATTRIBUTES ] ➘
  [FILES]
```

Version (`-v VERSION`) and project (`-p project`) are optional. They correspond to directory names and may be placed anywhere after options. One of

Photo by Viktor Talashuk on Unsplash

the three operators shown in Table 1 must be given.

By default, `chattr` does not return any output, but using the verbose (`-V`) option returns the same information as `lsattr` (Figure 2).

Like `chattr`, `lsattr` uses the same options, as well as others. However, its main options set how output is displayed. With `-a`, the output includes hidden files, while `-d` lists directories without their contents. For those who want readability without reference to the man page, `-l` prints the long names of attributes, rather than their one-character abbreviations. In addition, `-p` lists the project, and `-v` the file's version. Remember that normal permissions determine which files each user can view. Unlike `chattr`, `lsattr` is edited with the structure you might expect, with the basic command followed by options, followed by the target files (Figure 3).

Selected Attributes

Each attribute is assigned a single uppercase or lower-class letter. Each attribute is given a paragraph in the man page, but the mnemonic abbreviation in the documentation (`aAcCdDeFijmPsStTux`) is not much help, especially because letters sometimes seem to be assigned to attributes randomly. To make more sense of the attributes, here are some of the most useful attributes, arranged by category.

As might be expected, the largest category of attributes affect how files are edited. With the `C` attribute, copy-on-write (the resource management tool that only creates duplicates of files when one is modified) is suppressed. If applied to a directory that uses copy-on-write, existing files are unchanged, but new files take on the attribute. By contrast, `D` writes two files or directories recursively and at the same time, ensuring that they are identical, just like the `dirsync` command [4]. The same function is had for file attributes using `S`. When editing a file, `a` only allows content to be added at the end of the file, while `i` prevents a file from being edited, renamed, or deleted. Alternatively, `V` uses the kernel's fs-verity feature [5] to have all operations that use the file checked against a cryptographic hash – a so-called Merkle tree [6] – and also makes the file undeletable. Conversely, when a file is deleted, `J` saves all data on an ext4 system to `journal`, while `u` saves a file so that `chattr` can undelete it. For security, a file can also be overwritten with zeros to keep it from being retrieved.

Other normal file operations can also be affected by attributes. For instance, `E` encrypts, and `d` excludes a file in a backup done with `dump`. With `c`, a file is compressed, while `m` prevents a file from being compressed. Developers using `chattr` might want to use `P` to enforce a project's directory hierarchy. Given the tendency of Linux users to write entirely in lowercase, many users might also want to use `I`, which allows a file's path to be entered without uppercase characters.

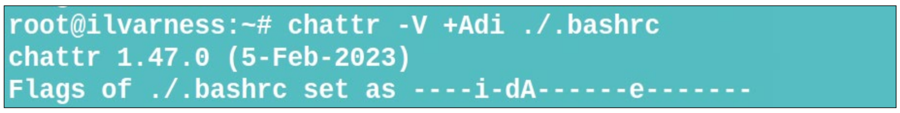
Due to the variety of modern filesystems, all attributes may not work on all filesystems. For instance, applying `t` to ext2, ext3, or ext4 filesystems is pointless, because it suppresses tail merging, which these filesystems do not support. Some filesystems, too, may have limitations on certain attributes. As well, `chattr` cannot delete all attributes, such as `N`, which stores data in the inode.

Check, too, for limitations on other attributes. Because of these limitations, I recommend checking the man page for `chattr` before assigning an attribute. Usually, an unremovable attribute can still be read by `lsattr`.

The Future of Attributes

Because `chattr` and `lsattr` include legacy support, not all attributes may be relevant today. For example, one reason for the `A` attribute, which does not write `atime` (the last time a file was accessed) was to reduce the wear on SSD drives when they were originally introduced and had shorter life spans than mechanical hard drives and laptops. However, modern SSD drives can outlast hard drives, so except on machines that are a decade or older, applying the `A` option has little point. Similarly, on immutable distributions such as Fedora's Silverblue, the `i` option could become obsolete, at least for system files, because its attribute is automatically used.

However, that still leaves plenty of useful options for `chattr` and `lsattr`. No doubt, too, unique features on one filesystem will become usable on others, and new needs will evolve, making `chattr` and `lsattr` more useful than ever in the Linux toolbox ■■■



```
root@ilvarness:~# chattr -V +Adi ./.bashrc
chattr 1.47.0 (5-Feb-2023)
Flags of ./.bashrc set as ----i-dA------e-------
```

Figure 2: The verbose option in `chattr`, with three attributes added, can simulate `lsattr`'s output.

Info

[1] chattr: *https://en.wikipedia.org/wiki/Chattr*

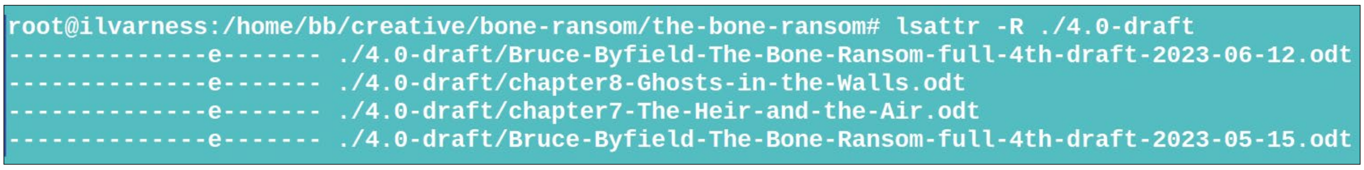
[2] lsattr: *https://man7.org/linux/man-pages/man1/lsattr.1.html*

[3] e2fsprog: *https://en.wikipedia.org/wiki/E2fsprogs*

[4] dirsync: *https://linux.die.net/man/1/dirsync*

[5] fs-verity: *https://www.kernel.org/doc/html/latest/filesystems/fsverity.html*

[6] Merkle tree: *https://en.wikipedia.org/wiki/Merkle_tree*



```
root@ilvarness:/home/bb/creative/bone-ransom/the-bone-ransom# lsattr -R ./4.0-draft
-------------e-------- ./4.0-draft/Bruce-Byfield-The-Bone-Ransom-full-4th-draft-2023-06-12.odt
-------------e-------- ./4.0-draft/chapter8-Ghosts-in-the-Walls.odt
-------------e-------- ./4.0-draft/chapter7-The-Heir-and-the-Air.odt
-------------e-------- ./4.0-draft/Bruce-Byfield-The-Bone-Ransom-full-4th-draft-2023-05-15.odt
```

Figure 3: `lsattr` displays the attributes of all the files in a directory. The `e` attribute indicates that the system's files use extents for mapping the blocks on disk. This attribute cannot be removed using `chattr`.

Using a RESTful API to get data from a music database

Catching up on REST

Craft the perfect URL to access metadata in an online music database. *By John Cofield*

Many network-based web services offer APIs that comply with the REpresentational State Transfer (REST) design style. APIs that comply with REST constraints are called RESTful.

RESTful APIs are attractive because they leverage existing HTTP infrastructure, the same infrastructure and protocols that browsers use. As a result, a RESTful API has access to a great variety of resources (applications, libraries, developers, etc.). From an end-user perspective, you can use your existing browser as a client to access server databases. Developers like RESTful APIs because it is easy to integrate server data and resources into applications.

Music services often use RESTful APIs to provide metadata about artists and musical tracks. Offering this data through a REST interface makes it very easy to build a client application that will receive and display the information. This article shows how to query music data through a RESTful API.

Why REST

A RESTful API allows the client to retrieve data from the server without needing to know details about the server back-end implementation. There is no need for the client to know which server or database management software is used. In addition, the API enhances security by restricting who can access data and selectively restricting which data is accessible.

RESTful APIs operate in two modes, request and response. The client sends a

More on REST

RESTful APIs are built around the following design principles:

- User interface – all requests for the same information look the same, regardless of where the request came from.
- Client-server decoupling – client and server are assumed to be totally independent. The only information that passes between them is through the API.
- Statelessness – each request contains all the input necessary for processing. There is no need to establish a "session." The state of the server remains constant.
- Cacheability – data should be cacheable on the client or server to improve performance.
- Layered system architecture – intermediate layers are integrated invisibly into the architecture. The client can't tell if it is connected directly to the end server or through an intermediary. Layered components adhere to REST constraints.
- Code on demand (optional) – integration with client-side components such as Java-applets or JavaScript.

An API isolates the client from the server, allowing communication through a uniform interface. This approach facilitates the development and debug process by giving you better visibility into network components. As a result, implementation of changes on either the client side or the server side can occur without either one affecting the other, as long as the interface protocol is followed [1].

The following is a list of the REST interface constraints:

- Identification of resources
- Manipulation of resources through representations
- Self-descriptive messages
- Hypermedia as the engine of the application state

The first REST constraint, "identification of resources," uses identifiers to identify target resources. In a RESTful API, these identifiers are referred to as representations. By design, a RESTful API never manipulates server resources directly. Instead, it only manipulates their representations, the identifiers. In the example in this article, you will see how these representations are used in communication between client and server.

request to the server for data. The server then responds with a status code, and if appropriate, the requested data.

In a RESTful web service, requests made to a resource's URI elicit a response with a payload usually formatted in HTML, XML, or JSON. The most common data transfer protocol for these requests and responses is HTTP, which provides operations (HTTP methods) such as `GET`, `POST`, `PUT`, `PATCH`, and `DELETE`. See the box entitled "More on REST" for additional background on RESTful APIs.

Music APIs

Several well-known music services provide RESTful API database access to musicians and to partners who want to integrate their content or music into commercial products.

Spotify, for example, provides an API that allows hardware partners to develop applications for home audio systems, music players, headphones, and other internet-enabled devices. According to the Spotify for Developer website, "Spotify Web API endpoints return JSON metadata about music artists, albums, and tracks, directly from the Spotify Data Catalog."

SoundCloud is a music service that allows musicians to share their music with a community of artists and listeners. Musicians can use the API to upload and manage their music for listeners.

MusicBrainz views its service as an open encyclopedia for music metadata. MusicBrainz is modeled after Wikipedia in that it is community-driven [2]. The metadata content is primarily, but not exclusively, targeted at music player and tagger applications. In this article, I will use the open source MusicBrainz API to request and receive music data through HTTP. (See the "HTTP Primer" box for more on accessing resources through HTTP.)

HTTP Primer

In order to understand RESTful APIs, it is essential to understand the role that HTTP plays. The Hypertext Transfer Protocol (HTTP) is a stateless application-level protocol for distributed, collaborative, hypertext information systems [3]. HTTP was designed, from its beginning in 1990, to support the client-server model (Figure 1), with requests passing in the form of messages from a client to a server. A response message and status code then passes from the server to the client.

When HTTP was first introduced, the only method used to send requests was the `GET` method. HTTP has since evolved to include additional methods to allow requests to modify and delete content on the server. Among the methods currently used are: `GET`, `POST`, `PUT`, and `DELETE` (Table 1).

When a request message is sent by a user agent (the client application) to a server, the message typically contains header fields that can include metadata such as: method, target hostname, content type, content length, and other characteristics of the client, host, or message.

Upon receiving a request from a user agent, the server determines whether it can accept the request. If accepted, the server responds by sending a status code and a message back to the client. Response message content might include metadata or the requested target resource.

MusicBrainz API

The MusicBrainz API gives the client access to a wide range of music metadata about artists and their music, including biographical information, release dates, and media formats. Requests are in the form of a URL that is comprised of multiple components, some mandatory and some optional [4].

REST constraints require the API to be stateless. This means that each request sent by the client to the server must contain all the information the server needs to respond appropriately. As a consequence, all the necessary information is contained in the URL. In the case of MusicBrainz, the syntax for the URL is as follows:

```
<api_root><entity><mbid><inc><format>
```

`<api_root>` is the partial URL to the API, in this case:

```
https://musicbrainz.org/ws/2/
```

`<entity>` is one or more information categories, such as artist, recording, release, etc. `<mbid>` is the MusicBrainz identifier that is unique to each target resource in the database. `<inc>` is an optional subquery string. `<format>` is an optional format string that specifies the transfer format, which can be either JSON or XML. (XML is the default format.)

Python urllib Module

This example illustrates one approach for sending an HTTP or HTTPS request and retrieving the response. I will use the `GET` method to retrieve metadata for a single recording and will request the response data in JSON format.

Multiple Python packages are available to send HTTP requests and handle responses. Some have external package dependencies. For the purposes of this exercise, I will use the built-in *urllib* module [5], which does not require installing an external package.

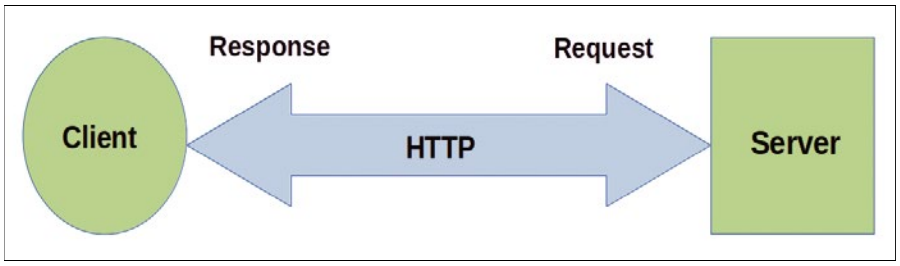


Figure 1: HTTP supports the client-server model.

Table 1: HTTP Methods

`GET`	Asks the server to send the target resource based on an identifier (representation).
`POST`	Asks the server to have the target resource process the enclosed content specified by an identifier.
`PUT`	Asks the server to have the target resource create or replace the enclosed content specified by an identifier.
`DELETE`	Asks the server to have the target resource specified by an identifier to be deleted.

The goal of the exercise is to:

- Send a request from the client (Python script)
- Retrieve the response data from the MusicBrainz server
- Retrieve the response status code

I'll start by importing the required Python modules:

```
import json
import os
import urllib.request
import webbrowser
import pprint
```

I'll use the following Python methods and objects:

```
urllib.request.Request()
json.loads()
request.get_method()
webbrowser.open_new_tab()
request.host
request.status
request.type
```

URL Components

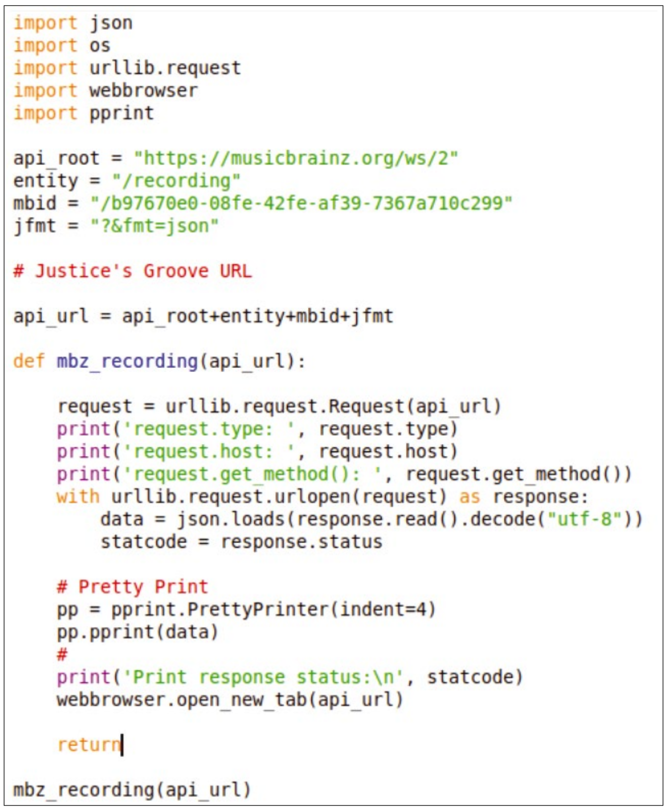
Listing 1 shows the values assigned to the MusicBrainz URL components. Each component is concatenated to form the full URL (`api_url`) required to request the API resource.

In Figure 2, I request a resource using the `urllib.request.Request(api_url)` method. Since the only argument is `api_url`, the request method defaults to the `GET` method. A `POST` request is sent to the server using the same method, specifying `POST` or passing a `data` argument. However, the `POST` method must be explicitly specified.

Listing 1: URL Component Values

```
api_root = "https://musicbrainz.org/ws/2"
entity = "/recording"
mbid = "/b97670e0-08fe-42fe-af39-7367a710c299"
jfmt = "?&fmt=json"

# Full API URL
api_url = api_root+entity+mbid+jfmt
```



```
import json
import os
import urllib.request
import webbrowser
import pprint

api_root = "https://musicbrainz.org/ws/2"
entity = "/recording"
mbid = "/b97670e0-08fe-42fe-af39-7367a710c299"
jfmt = "?&fmt=json"

# Justice's Groove URL

api_url = api_root+entity+mbid+jfmt

def mbz_recording(api_url):

    request = urllib.request.Request(api_url)
    print('request.type: ', request.type)
    print('request.host: ', request.host)
    print('request.get_method(): ', request.get_method())
    with urllib.request.urlopen(request) as response:
        data = json.loads(response.read().decode("utf-8"))
        statcode = response.status

    # Pretty Print
    pp = pprint.PrettyPrinter(indent=4)
    pp.pprint(data)
    #
    print('Print response status:\n', statcode)
    webbrowser.open_new_tab(api_url)

    return

mbz_recording(api_url)
```

Figure 2: Requesting data with a Python script.

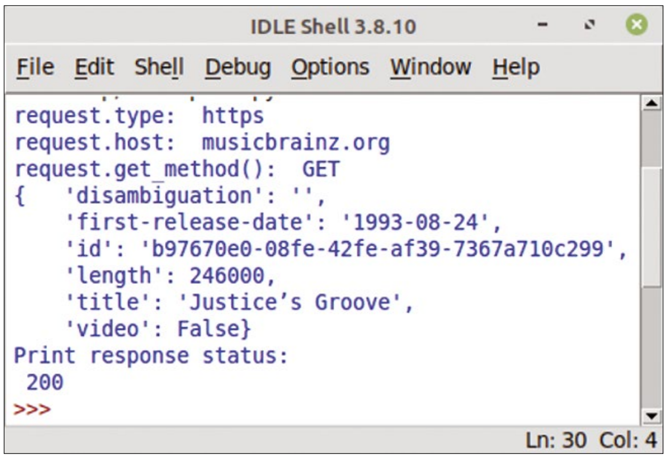
In Figure 3, you can see the response values displayed in the Python IDLE Shell window. Responses normally contain a lot of information, but I have limited the output to display only the `type`, `host`, and `method` to confirm that the request was sent to the MusicBrainz server, that it was sent via secure HTTPS, and that the `GET` method was used.

Note that the JSON data is deserialized by the `json.loads()` method, converted to Python dictionary form, and displayed. Finally, the code value `200` indicates that the request was successfully executed.

If you prefer viewing the data in a browser, one option you can use is the `open_new_tab()` method from the Python built-in module `webbrowser`. The web browser output is shown in Figure 4.

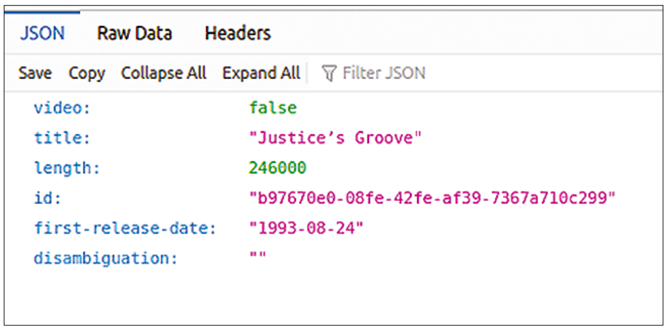
Security

Web APIs typically require some sort of authentication to verify that the client submitting a request is authorized to do so. The most common authentication protocols are HTTP Digest Access Authentication and JSON Web Token (JWT). HTTP Digest Access Authentication uses 256- or 512-bit hash-encrypted username and password to authorize access. JWT tokens are often generated by a third-party authorization server that you authorize in



```
request.type:  https
request.host:  musicbrainz.org
request.get_method():  GET
{   'disambiguation': '',
    'first-release-date': '1993-08-24',
    'id': 'b97670e0-08fe-42fe-af39-7367a710c299',
    'length': 246000,
    'title': 'Justice's Groove',
    'video': False}
Print response status:
 200
>>>
```

Figure 3: Server response: JSON data text output in the Python IDLE Shell window.



```
video:                false
title:                "Justice's Groove"
length:               246000
id:                   "b97670e0-08fe-42fe-af39-7367a710c299"
first-release-date:   "1993-08-24"
disambiguation:       ""
```

Figure 4: Server response: JSON data to the browser.

advance to give you access to one or more applications. The token approach gives you a single sign-on (SSO) source so you don't have to give your username and password to the web service.

MusicBrainz `POST` requests, which allow you to alter data, require authentication. In order to submit a `POST` request, a client application must register using either a username/password or JWT (OAuth 2.0) token. An authorization request must be submitted prior to submitting a `POST` request. See the MusicBrainz website for more information on MusicBrainz authentication [6].

Summary

A RESTful API is an easy way to access music data for a custom script or client application. The REST architecture enables you to develop client applications that can reach flexible and secure RESTful APIs. REST lets you update client and server components independently, which makes RESTful components easier to maintain than tightly integrated client/server systems, and the ability to manipulate representations rather than resources adds additional security to the design. ■■■

Author

John Cofield is a retired software marketing manager in Northern California. His training is in electrical engineering, and he has worked at multiple Silicon Valley semiconductor and software companies. His nontechnical interests include Jazz music ranging from Modal to Fusion.

Info

[1] Fielding, R. *Architectural Styles and the Design of Network-based Software Architectures*, PhD Dissertation, University of California, Irvine, September 2000: *https://roy.gbiv.com/pubs/dissertation/top.htm*

[2] MusicBrainz: *https://musicbrainz.org/*

[3] HTTP Semantics, RFC 9110: *https://httpwg.org/specs/rfc9110.html*

[4] MusicBrainz FAQ: *https://musicbrainz.org/doc/MusicBrainz_API#General_FAQ*

[5] urllib: *https://docs.python.org/3.10/library/urllib.html*

[6] MusicBrainz authentication: *https://musicbrainz.org/doc/Development/OAuth2*

Anonymity with ProxyChains

Secret Chain

If you want to stay anonymous on the web, you don't need the Tor browser or a Tor-based distro like Tails. ProxyChains obscures your presence through proxies – with or without Tor on the back end. *By Chris Binnie*

A number of scenarios lead to a need for online anonymity. You might wish to remain anonymous while working as a journalist in a war-torn country, for example, or when acting as a whistleblower. Perhaps you are working as a security researcher? Or maybe you just want some privacy from commercial Internet businesses. This article shows how to disguise Internet traffic using a tool called ProxyChains [1]. I came across ProxyChains while taking part in some challenges at the TryHackMe site [2].

If there are multiple links in a chain of proxies, it is very hard to perform digital forensics on a visiting IP address. Using a number of proxies to route traffic through before the traffic makes it back to your computer is a highly effective way of remaining anonymous online. If you think about it for a moment, though, not all protocols work in this scenario. For instance, traffic routed in this way typically requires the TCP transfer protocol. DNS also poses some challenges. DNS lookups use UDP instead of TCP. A common weakness with VPNs is that they leak DNS lookups to systems outside of the VPN. The trail of DNS lookups makes it possible for external systems to harvest traffic and determine what sites a user has been visiting.

ProxyChains manages to cleverly resolve DNS through proxies. You can use ProxyChains with many popular TCP-based client applications – including network scanners, mail clients, and web browsers. The versatile ProxyChains is capable of supporting SOCKS4, SOCKS5, and HTTP proxy servers and can even mix different proxy types in the same chain.

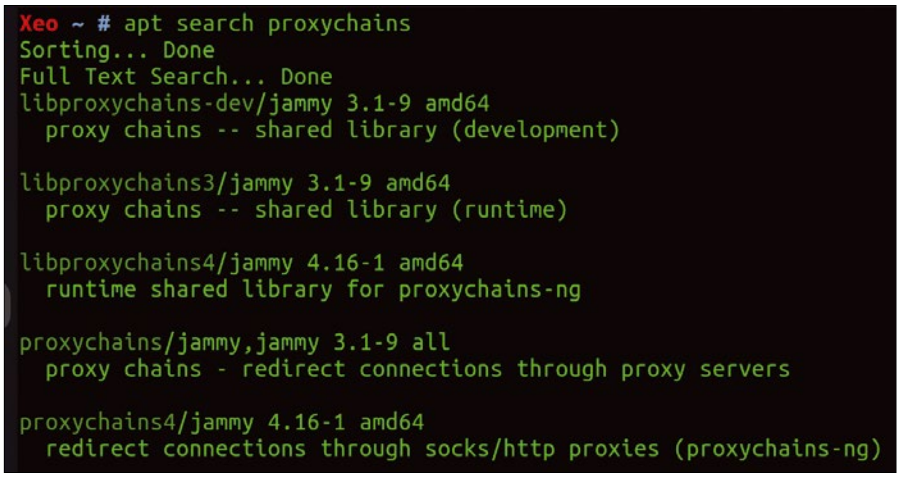
You might be surprised that such a potentially controversial piece of software is readily available in the Ubuntu Linux package repositories. The current version available on GitHub is *Proxychains-4.3.0*, with the last pull request merged when a patch was added around a year ago. And, according to the README page in the repository, if you can't find it using your favorite Linux distribution's package manager, all is not lost. ProxyChains is widely supported on other platforms too and is "…available with pkgsrc to everyone using it on Linux, NetBSD, FreeBSD, OpenBSD, DragonFlyBSD or Mac OS X."

The Ubuntu package repositories that are visible for ProxyChains appear in Figure 1. I used the following command to display the package repositories list:

```
$ apt search proxychains
```

Although the GitHub version was *Proxychains-4.3.0*, Ubuntu is currently offering `proxychains4/jammy 4.16-1` for Ubuntu 22.04.2 LTS. The Ubuntu version is a respectable release that is supported for easy installation, but if you want to push for the very latest features, use the GitHub version and refer to the README for the common `make install` build instructions.

It should go without saying that the tools and information described in this article are purely for educational purposes and legal activities. If you work in the security field and defend systems against attacks, it can only help you in your defensive efforts to learn more about techniques used by attackers.



```
Xeo ~ # apt search proxychains
Sorting... Done
Full Text Search... Done
libproxychains-dev/jammy 3.1-9 amd64
  proxy chains -- shared library (development)

libproxychains3/jammy 3.1-9 amd64
  proxy chains -- shared library (runtime)

libproxychains4/jammy 4.16-1 amd64
  runtime shared library for proxychains-ng

proxychains/jammy,jammy 3.1-9 all
  proxy chains - redirect connections through proxy servers

proxychains4/jammy 4.16-1 amd64
  redirect connections through socks/http proxies (proxychains-ng)
```

Figure 1: **ProxyChains is available in the Ubuntu repositories.**

Routing with a Proxy List

The default behavior of ProxyChains is to route traffic through the Tor anonymity network. However, many users prefer to route directly through a predefined set of proxies on the conventional Internet. Defining the proxies directly has some benefits. For instance, you won't have to install the Tor service on your computer. Also, Tor can be problem as many online services detect and then actively deny traffic from the Tor network, as they are suspicious of a user's intent to route traffic through it.

The first step is to install the ProxyChains software and configure the `.conf` file to use a list of proxies. I have a couple of lists of free proxies to refer to and will make use of those. To install the package as the root user, use the following command, only a minuscule 145KB of disk space will be used up as a result:

```
$ apt install proxychains4 -y
```

This command will install the *libproxychains4* and *proxychains4* packages. You'll need 42.1KB for the archives and an additional 145KB of additional disk space.

The location of the configuration file is `/etc/proxychains4.conf`. In this case, I will use the dynamic chain mode, which is described in the config file comments as follows:

- Each connection will be done via chained proxies
- All proxies are chained in the order as they appear in the list
- At least one proxy must be online to play in the chain
- Dead proxies are skipped

I will uncomment the `dynamic_chain` line by removing the first character and then add a hash or pound sign to the line starting `strict_chain` in order to disable it. The comments explain that strict mode means that all proxies will be chained in exactly the order that they're listed and that all proxies must be online if you want to use the chain. You can also use the settings of the config file to affect the length of the chain, turn on quiet mode, enable a random chain, and change how ProxyChains handles DNS lookups.

ProxyChains offers some alternative approaches for how to perform anonymous DNS lookups. The `proxy_dns` setting is described as the fastest and easiest method to use: A thread is spawned that serves DNS requests and hands down an IP address assigned from an internal list (via `remote_dns_subnet`).
Look online for a really nice explanation of how DNS and ProxyChains work [3]. The post is about three years old, and the exact process might have evolved a little over time, but as you'll see in a moment, ProxyChains is able to cleverly use IP address ranges (such as the 224.x.x.x and 225.x.x.x ranges) in order to keep things private.

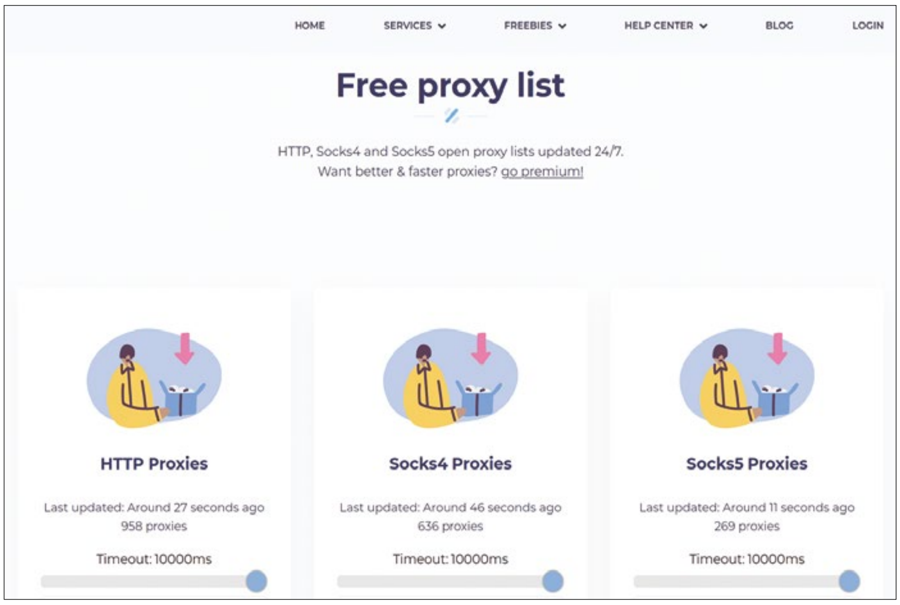
The proxy list section of the config file is shown in Listing 1. As you can see in Listing 1, the end of the file is where to add your list of proxies.

To create a list of proxies that I can test with, I'll visit the ProxyScrape website [4]. You can also find several GitHub repositories that update frequently to automate your proxy list [5]. Another source for proxies is the Free Proxy List site [6].

Websites like ProxyScrape offer free proxies to get you started. If you scroll down a little on their website, you'll find different types of proxies, including HTTP, SOCKS4, and SOCKS5. There's also a table of the countries, IP addresses, network ports, and latency. SOCKS5 supports authentication and other protocols. (See the Security

Listing 1: **ProxyList Format**

```
# ProxyList format
#       type  ip  port [user pass]
#       (values separated by 'tab' or 'blank')
#
#       only numeric ipv4 addresses are valid
#
#
#        Examples:
#
#            socks5  192.168.67.78   1080   lamer   secret
#            http    192.168.89.3    8080   justu   hidden
#            socks4  192.168.1.49    1080
#            http    192.168.39.93   8080
#
#
#       proxy types: http, socks4, socks5, raw
#         * raw: The traffic is simply forwarded to the proxy without modification.
#        ( auth types supported: "basic"-http  "user/pass"-socks )
#
[ProxyList]
# add proxy here ...
```



Free proxy list

HTTP, Socks4 and Socks5 open proxy lists updated 24/7.
Want better & faster proxies? go premium!

HTTP Proxies

Last updated: Around 27 seconds ago
958 proxies

Timeout: 10000ms

Socks4 Proxies

Last updated: Around 46 seconds ago
636 proxies

Timeout: 10000ms

Socks5 Proxies

Last updated: Around 11 seconds ago
269 proxies

Timeout: 10000ms

Figure 2: Use the link on the right side of the window to access the SOCKS5 proxy list in ProxyScrape.

Intelligence website for more on SOCKS5 [7].)

If you click one of the links above the list of proxies, you'll see a choice of options (Figure 2). I'll choose *Socks5 Proxies*. You can see the proxy lists were updated a few seconds ago, which is reassuring.

Clicking on *Socks5 Proxies* reveals a long list of IP addresses and port numbers. I counted 271 proxies, which is both generous and impressive for a free service, even with the assumption that some of them will be offline when I try to use them.

As you can see in Listing 1, I can choose which type of proxies I will use: HTTP, SOCKS4, SOCKS5, and raw. I'll reduce it to the first three entries and add `SOCKS5` to the front of line, so the format changes from: `XXX.XXX.XXX.XXX:PORT` (where the Xs are the octets in the IP address) to the following format:

```
SOCKS5 XXX.XXX.XXX.XXX PORT
```

Many of the proxies in the list have odd-looking port numbers, but their presence in the list assures that they are SOCKS5 proxies. Before editing the (now very long) configuration file, I make a backup copy to save any future headaches. Then I reduce the massive proxy list to just three proxies for testing. To match the format for a SOCK5 proxy, I add `SOCKS5` to the start of each line in the proxy list and then replace the colon before the port number with a space.

Next I will comment out the following lines (by adding a hash symbol at the start of the line), which would have caused the traffic to route through the Tor network directly and not the list of proxies:

```
# defaults set to "tor"
# socks4        127.0.0.1 9050
```

I will run a cursory port scan test against the hostname `scanme.nmap.org`, a site that allows port scans if you stick with their acceptable usage [8].

Figure 3 shows what happens when you use the Telnet program via three proxies in a chain using the ProxyChains configuration. I've redacted the IP addresses. The command I used in Figure 3 was:

```
$ proxychains telnet ↩
scanme.nmap.org 80
```

In Figure 3, you can see three redacted IP addresses before the target hostname `scanme.nmap.org`. I have used the time-honored Telnet command `GET /` to request a web page. As you can see, the top of the abbreviated web page is then dutifully delivered to the command line.

This test shows that I can route traffic through multiple, chained proxies and receive traffic back successfully. ProxyChains is working beautifully.

Peeling Back The Onion

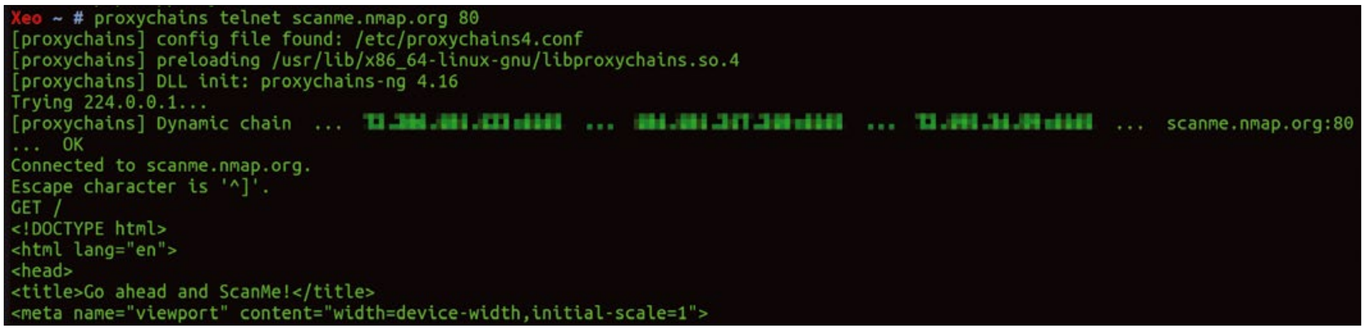
Tor is a remarkable, collaborative effort by the Internet community. The Tor network [9] routes traffic via user-supplied broadband connections, of which there are thousands globally, offering high levels of anonymity. I won't do it justice skimming over its features here, but you can read more about it online if you're new to Tor.

It's worth saying that you don't need ProxyChains to use the Tor network, but they play very nicely together. ProxyChains does not use the Tor browser – it simply uses the Tor service on the back end for establishing the chain of anonymous proxies. You'll find Tor in the Ubuntu repositories. Install it with:

```
$ apt-get install tor
```

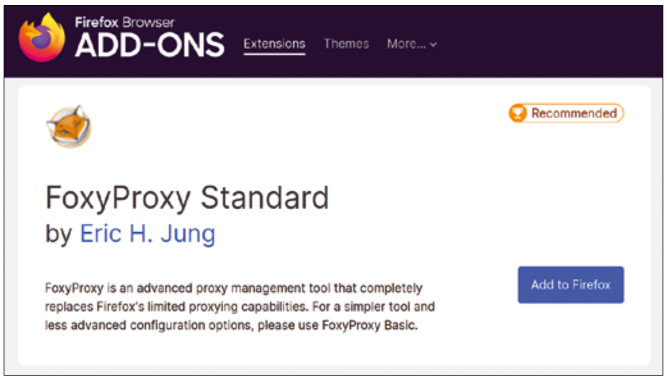
In my case, the package manager installed version 0.4.6.10-1, and it comes in a little under 3MB. Start the `tor.service` unit file with:

```
$ systemctl start tor
```



```
Xeo ~ # proxychains telnet scanme.nmap.org 80
[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
Trying 224.0.0.1...
[proxychains] Dynamic chain  ...  [redacted]  ...  [redacted]  ...  [redacted]  ...  scanme.nmap.org:80
...  OK
Connected to scanme.nmap.org.
Escape character is '^]'.
GET /
<!DOCTYPE html>
<html lang="en">
<head>
<title>Go ahead and ScanMe!</title>
<meta name="viewport" content="width=device-width,initial-scale=1">
```

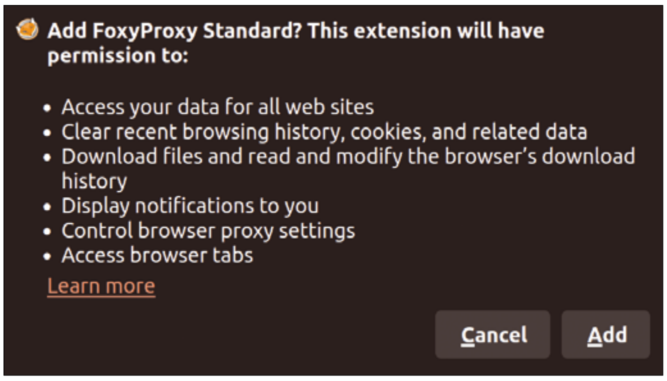
Figure 3: The three redacted SOCKS5 proxies in this chain worked like a treat.



FoxyProxy Standard
by Eric H. Jung

FoxyProxy is an advanced proxy management tool that completely replaces Firefox's limited proxying capabilities. For a simpler tool and less advanced configuration options, please use FoxyProxy Basic.

Figure 4: It's a FoxyProxy extension for Firefox.


Add FoxyProxy Standard? This extension will have permission to:

- Access your data for all web sites
- Clear recent browsing history, cookies, and related data
- Download files and read and modify the browser's download history
- Display notifications to you
- Control browser proxy settings
- Access browser tabs

Figure 5: Permission to come aboard.

If you plan to use Tor, you'll need to uncomment the lines commented out previously in the `/etc/proxychains.conf` file:

```
defaults set to "tor"
socks4  127.0.0.1 9050
```

In the previous test, I used Telnet on the command line. What about using an application with a user interface? Consider the venerable Firefox browser from Mozilla as an example. Before you try to use Firefox with ProxyChains and the Tor network, be sure your Firefox version is up to date:

```
$ snap refresh firefox
firefox 113.0.1-1 from Mozilla? ↩
refreshed
```

Next, I need to ensure that Firefox uses the specified ProxyChains proxy (which is the Tor network in this case). I'll use a popular extension for Firefox called FoxyProxy [10]. You might have used FoxyProxy to redirect traffic through Burp Suite [11] during Capture The Flag (CTF) exercises or while practicing ethical hacking.

To install FoxyProxy, start Firefox and select *Add-ons and themes* in the application menu. Figure 4 shows the blue *Add to Firefox* button on the add-ons page.

You are then confronted with a permissions pop-up dialog box (Figure 5). Click the *Add* button in order to continue.

Once FoxyProxy is installed, look for the icon in the top-right corner of the browser to access the FoxyProxy settings. If you don't see the icon, you might have to rummage through the Firefox menus (*Settings* | *Extensions & Themes*) in order to open up the FoxyProxy settings. For future reference, if the icon is green, traffic is being routed via one of your preconfigured proxies. Figure 6 shows what happens when you click the little FoxyProxy icon.

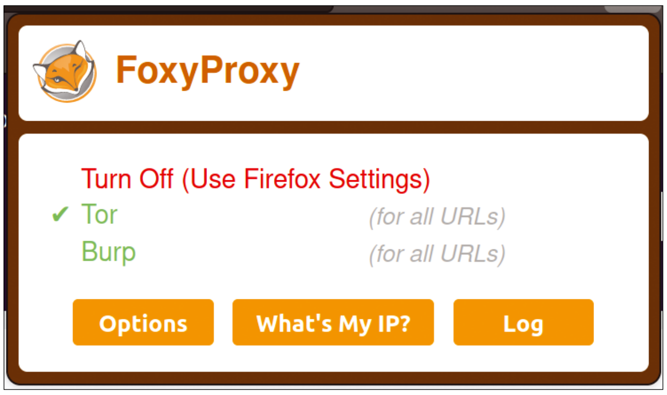
If you haven't already set up Tor, click *Options* and you'll be presented with a screen listing choices for your settings on the right and then a number of menu options on the left. Click the *Add* link at the top-left. Figure 7 shows what to fill in.

The options are a name for the proxy, the IP address (127.0.0.1), and the port (TCP port 9050 in this instance). I also selected SOCKS5 in the pull-down menu. Note the *Send DNS through SOCKS5 proxy* setting is also enabled in Figure 7.

Once you have saved the settings, you should be able to tick the *Tor* line, shown in the Figure 6.

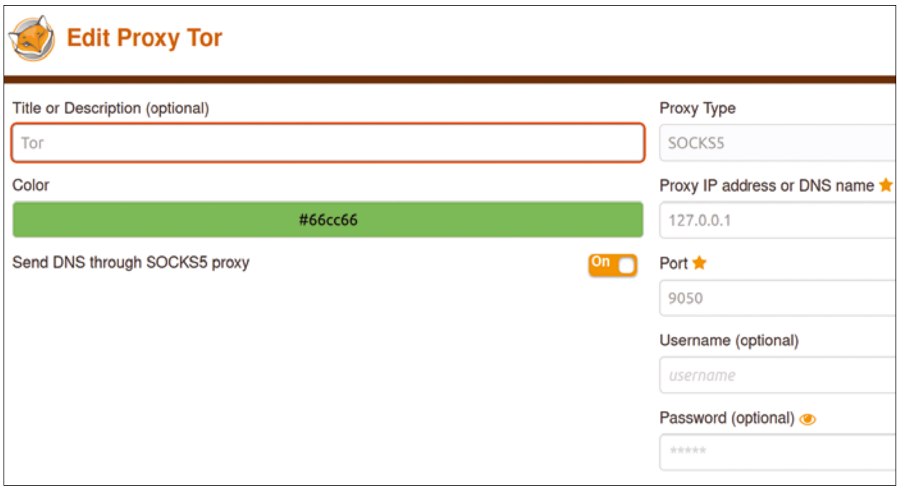
Interestingly, it looks like the Tor network configuration is accepting both SOCKS4 and SOCKS5 as the proxy type. To prove that all is well, I need to close all instances of Firefox and then open the application up again. In Figure 8, I can see what the website *Whats My IP* [12] shows when I visit.

In Figure 8, note the Netherlands IP Address (redacted to protect the innocent); that's not where I live! This test



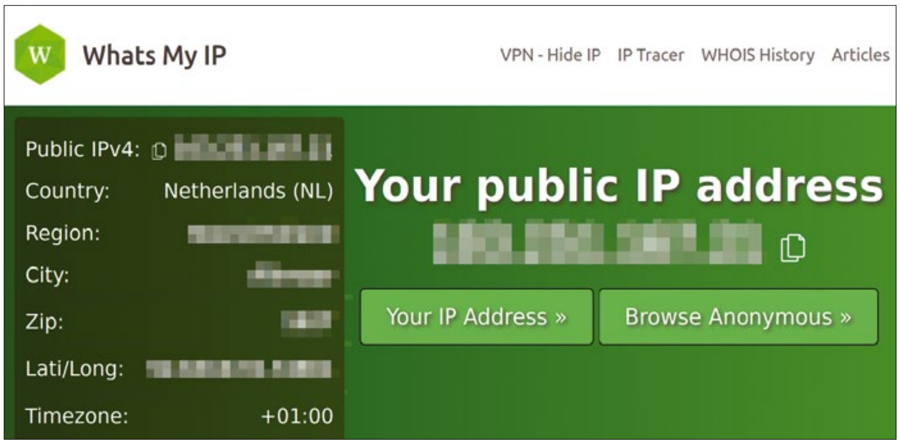
Turn Off (Use Firefox Settings)
✓ Tor (for all URLs)
Burp (for all URLs)

Figure 6: What you'll see when connected to ProxyChains and Tor.



Title or Description (optional): Tor
Color: #66cc66
Send DNS through SOCKS5 proxy: On
Proxy Type: SOCKS5
Proxy IP address or DNS name ★: 127.0.0.1
Port ★: 9050
Username (optional): username
Password (optional): *****

Figure 7: Where to add your Tor proxy settings in FoxyProxy.



Public IPv4: [redacted]
Country: Netherlands (NL)
Region: [redacted]
City: [redacted]
Zip: [redacted]
Lati/Long: [redacted]
Timezone: +01:00

Your public IP address
[redacted]

Figure 8: That's definitely not my IP Address.

shows that network traffic is successfully routing via the Tor network.

ProxyChains with Nmap

Your Tor/ProxyChains network isn't limited to just forwarding browser traffic. ProxyChains works with many other popular applications. For instance, you can use ProxyChains with the mighty network mapper Nmap [13]. Install Nmap on Debian derivatives with:

```
$ apt install -y nmap
```

Try the following command with your ProxyChains and Tor configuration:

```
$ proxychains nmap -v -Pn -sT 
-p22,80,443 scanme.nmap.org
```

Loosely speaking, the `-sT` switch means run a TCP scan. See the Nmap website for more on the *TCP Connect Scan* option [14]. Abbreviated output of the command appears in Listing 2.

The output in Listing 2 demonstrates that I have asked Nmap to scan three network ports on the target system. The `denied` response for TCP port 443 is not actually because Tor is rejecting the routing or failing to route the traffic but is because the destination port is closed to traffic and there's nothing to report back. The other two ports are open, and you can see an OK entry for each.

In addition to Nmap, you could run the Telnet command through ProxyChains to query ports and not just the web page, as shown in Listing 3.

Or if you just want to SSH to another host via the Tor network or a chain of proxies, you can use the command in Listing 4.

Leaking Everywhere

To make sure that I am not leaking any DNS lookups to third parties, I'll use a clever site called *DNSleaktest.com* [15].

Visit the site in Firefox (with proxying enabled), and then run the Standard test (Figure 9). If you are connected to a VPN and some servers displayed aren't associated with that VPN service, then you have a DNS leak.

Conclusion

ProxyChains is an anonymity tool that can operate with or without the Tor network. If you get stuck, a top troubleshooting tip is to make sure your laptop's system clock is precisely set. The simplest way to set the system clock is to just install the `ntpdate` package and manually run the update command. The following example shows a global NTP server pool, but you can adjust the

Listing 2: Routing Nmap with Tor/ProxyChains

```
Initiating Parallel DNS resolution of 1 host. at 14:07
Completed Parallel DNS resolution of 1 host. at 14:07, 0.04s elapsed
Initiating Connect Scan at 14:07
Scanning scanme.nmap.org (224.0.0.1) [3 ports]
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  scanme.nmap.org:80  ...  OK
Discovered open port 80/tcp on 224.0.0.1
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  scanme.nmap.org:443
<--denied
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  scanme.nmap.org:22  ...  OK
Discovered open port 22/tcp on 224.0.0.1
Completed Connect Scan at 14:07, 1.19s elapsed (3 total ports)
Nmap scan report for scanme.nmap.org (224.0.0.1)
Host is up (0.36s latency).
rDNS record for 224.0.0.1: all-systems.mcast.net

PORT    STATE  SERVICE
22/tcp  open   ssh
80/tcp  open   http
443/tcp closed https

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 1.27 seconds
```

Listing 3: Telnet Via ProxyChains

```
$ proxychains telnet scanme.nmap.org 22

[proxychains] config file found: /etc/proxychains4.conf
[proxychains] preloading /usr/lib/x86_64-linux-gnu/libproxychains.so.4
[proxychains] DLL init: proxychains-ng 4.16
Trying 224.0.0.1...
[proxychains] Dynamic chain  ...  127.0.0.1:9050  ...  scanme.nmap.org:22  ...  OK
Connected to scanme.nmap.org.
Escape character is '^]'.
SSH-2.0-OpenSSH_6.6.1p1 Ubuntu-2ubuntu2.13
```

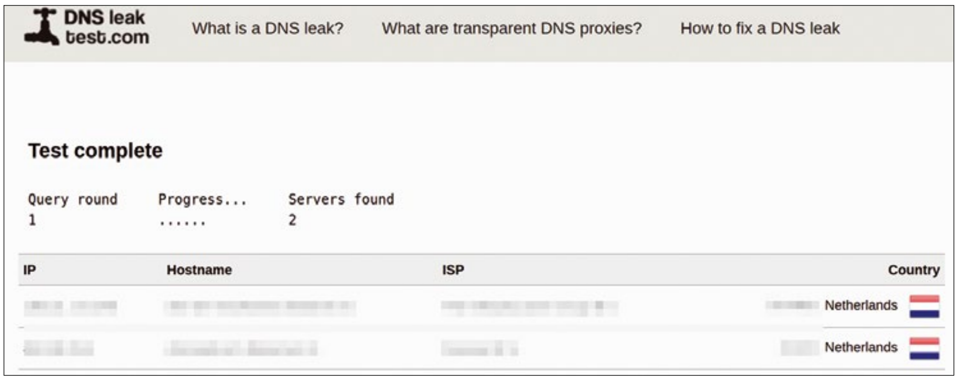
command to your region if needed:

```
$ apt install -y ntpdate
$ ntpdate -v pool.ntp.org
```

I hope that you have been suitably impressed with the undeniably sophisticated ProxyChains and the remarkable Tor network. Setting up ProxyChains is simple and painless, with a useful outcome that other software struggles to get close to.

There are many applications that you can push through a proxy chain using TCP-based traffic or via the Tor network. I'd suggest practicing with a variety of packages.

As a reminder, you should use your newly found knowledge ethically and wisely. ■■■



Test complete

```
Query round    Progress...    Servers found
1              ......         2
```

IP	Hostname	ISP	Country
			Netherlands
			Netherlands

Figure 9: Wow, a clear connection – without DNS leakage.

Listing 4: SSH Through Proxies

```
$ ssh -v -D 127.0.0.1:9050 scanme.nmap.org -p22
[...snip?]
debug1: /etc/ssh/ssh_config line 21: Applying options for *
debug1: Connecting to scanme.nmap.org [45.33.32.156] port 22.
debug1: Connection established.
[...snip?]
root@scanme.nmap.org's password:
```

Author
Chris Binnie is a Cloud Native Security consultant and coauthor of the book Cloud Native Security: *https://www.amazon.com/Cloud-Native-Security-Chris-Binnie/dp/1119782236*

Info

[1] ProxyChains: *https://github.com/haad/proxychains*
[2] TryHackMe: *https://tryhackme.com*
[3] "How Does ProxyChains Avoid DNS Leaks?": *https://security.stackexchange.com/questions/224394/how-does-proxychains-avoid-dns-leaks*
[4] ProxyScrape: *https://www.proxyscrape.com/free-proxy-list*
[5] PROXY-List on GitHub: *https://github.com/TheSpeedX/PROXY-List*
[6] Free Proxy List: *https://free-proxy-list.net*
[7] "What is SOCKS5 and Why Should I Use It": *https://securityintelligence.com/posts/socks-proxy-primer-what-is-socks5-and-why-should-you-use-it*
[8] Scanme at nmap.org: *http://scanme.nmap.org*
[9] The Tor Project: *https://www.torproject.org*
[10] FoxyProxy: *https://addons.mozilla.org/en-US/firefox/addon/foxyproxy-standard*
[11] Burp Suite: *https://portswigger.net/burp*
[12] What's My IP: *https://whatsmyip.com*
[13] Nmap network scanner: *https://nmap.org*
[14] Nmap scanning options: *https://nmap.org/book/port-scanning-options.html*
[15] DNS Leak Test: *https://www.dnsleaktest.com*

Tracking down hard disk space consumption with Go

Housecleaning

This month, Mike Schilli writes a quick terminal UI in Go to help track down space hogs on his hard disk. *By Mike Schilli*

When installing a new hard disk, users think they are in seventh heaven – it seems almost impossible to ever fill up the drive. Given today's huge capacities, though, virtually everyone becomes a thoughtless space waster sooner or later. As the disk fills up, space needs to be freed up to keep things going. Most of the time, the culprit is huge files no longer in use that are still hanging around in forgotten directories and causing overcrowding. The most effective strategy is often a simple one of tracking down the biggest space wasters and then deleting the unneeded files.

The `spacetree` Go program helps with this housecleaning by pointing to the most storage-intensive paths starting from a root directory and provides their disk space usage in bytes. As Figure 1 shows, the Go binary, which you call with the directory to be analyzed, switches the terminal to graphics mode and displays the storage paths as an interactive tree. You can expand the green elements in the tree with a mouse click or by pressing the Enter key to determine which folders contain the fattest files.

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.

Biggest Fish

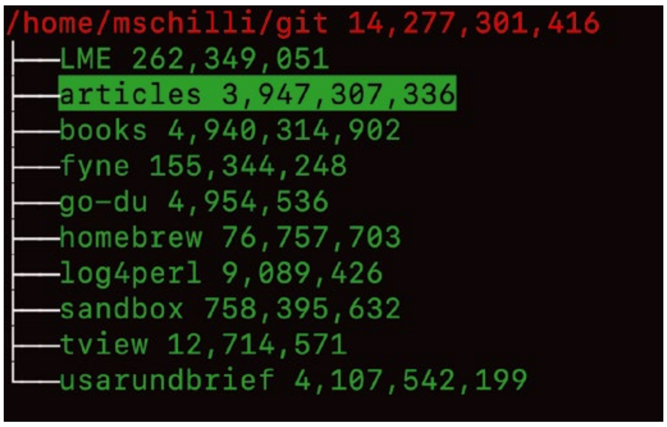
In Figure 1, the Go `spacetree` binary takes a look at the `~/git/` directory and discovers some 14GB of data there. Using the arrow keys, their vi equivalents *J* and *K*, or even the mouse if you fancy, you can navigate to the directories highlighted in green and expand them by pressing the Enter key. Figure 2 shows that, for example, the `articles/` directory, which contains the articles from 26 years of this column, occupies a total of almost 4GB.

Most of it, some 850MB, resides in the internal Git directory, `.git`, which I can't touch without disrupting its inner workings. However, for the biggest disposable space waster, check out the subdirectory with the `wifi` network checker from the October 2023 issue: This directory still has the screencast video in it! By deleting it, I've saved another 850MB. On top of that, the March 2019 `progressbar` article contains the unused `foo` and `foo.bak` files that really need to be removed.

Smarter with Heap

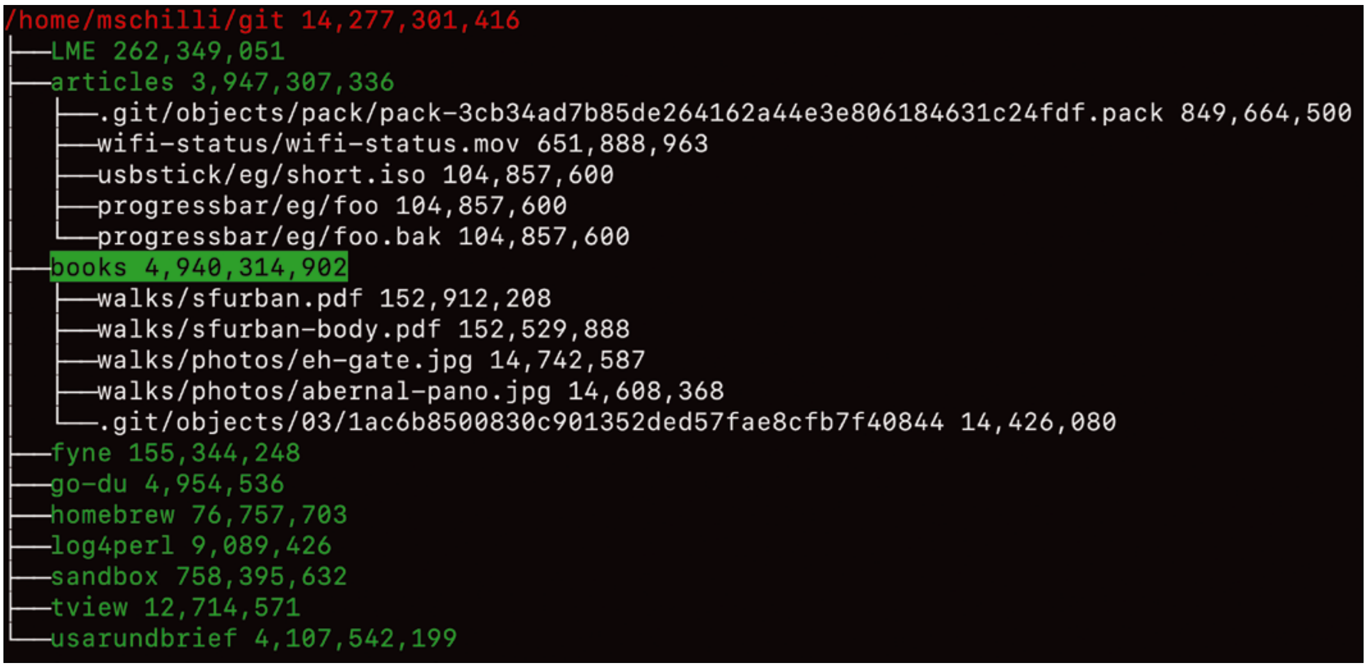
But how does the Go program traverse the deeply nested file structure to count the drive bytes used by each file and print a short list of the largest files? It finds the 10 biggest files from a huge number of files, which can easily be millons on a heavily used system. The simple sounding solution of dropping all the entries into an array, sorting the array in descending order by file size, and then outputting the first 10 entries is not an option. Even a highly effective sorting algorithm would take quite some time to process a million entries, not to mention the RAM wasted by such a huge array. After all, only 10 entries are of interest at the end of the day.

The so-called "min heap" proves to be an efficient data structure for problems like this. The nodes of this binary tree always satisfy the rule that the entries below a given node always have larger values



```
/home/mschilli/git 14,277,301,416
├──LME 262,349,051
├──articles 3,947,307,336
├──books 4,940,314,902
├──fyne 155,344,248
├──go-du 4,954,536
├──homebrew 76,757,703
├──log4perl 9,089,426
├──sandbox 758,395,632
├──tview 12,714,571
└──usarundbrief 4,107,542,199
```

Figure 1: The Go terminal user interface (UI) with storage-intensive Git directories.



```
/home/mschilli/git 14,277,301,416
├──LME 262,349,051
├──articles 3,947,307,336
│  ├──.git/objects/pack/pack-3cb34ad7b85de264162a44e3e806184631c24fdf.pack 849,664,500
│  ├──wifi-status/wifi-status.mov 651,888,963
│  ├──usbstick/eg/short.iso 104,857,600
│  ├──progressbar/eg/foo 104,857,600
│  └──progressbar/eg/foo.bak 104,857,600
├──books 4,940,314,902
│  ├──walks/sfurban.pdf 152,912,208
│  ├──walks/sfurban-body.pdf 152,529,888
│  ├──walks/photos/eh-gate.jpg 14,742,587
│  ├──walks/photos/abernal-pano.jpg 14,608,368
│  └──.git/objects/03/1ac6b8500830c901352ded57fae8cfb7f40844 14,426,080
├──fyne 155,344,248
├──go-du 4,954,536
├──homebrew 76,757,703
├──log4perl 9,089,426
├──sandbox 758,395,632
├──tview 12,714,571
└──usarundbrief 4,107,542,199
```

Figure 2: Expanding the directories shows you the biggest fish.

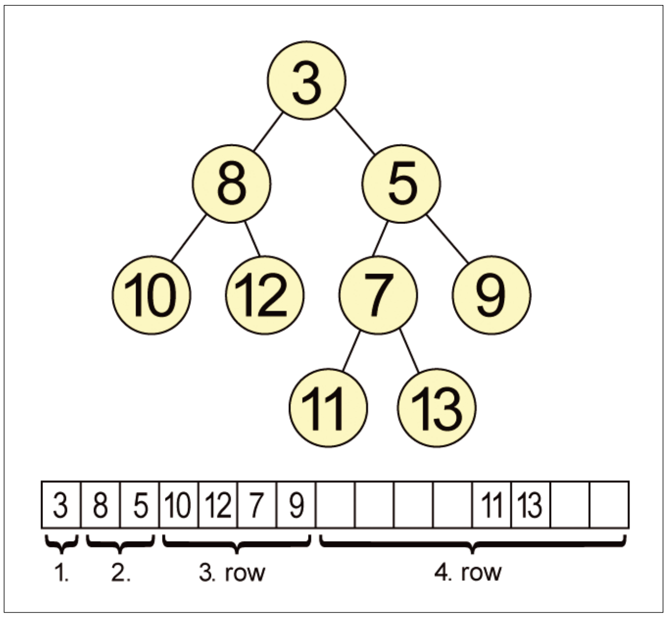
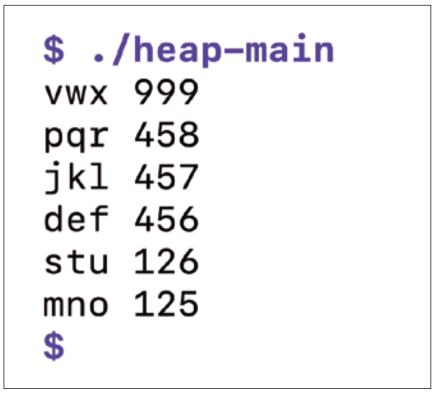


Figure 3: The min heap data structure and its array representation.



```
$ ./heap-main
vwx 999
pqr 458
jkl 457
def 456
stu 126
mno 125
$
```

Figure 4: Output of the test program for the min heap.

than the node itself. In Figure 3, the node with the value *3* is at the top of the tree root and is smaller than all other nodes in the tree. Note, though, that the rule applies only to directly or indirectly connected nodes. In Figure 3, for example, it is perfectly okay for the first element of the second row from the top, which belongs to the left subtree, to contain a value of *8* while the third row of the right subtree contains a value of *7*. They're not in a direct or indirect parent/child relationship, so the rule doesn't apply.

Minimal Heap

Software programs represent the tree as a simple array which, as shown in Figure 3, creates subsets of one, two, four, eight, and so on that border on each other seamlessly, each subset representing one horizontal row within the tree. It is now relatively easy to find and remove the smallest entry in the tree, because it is always at the top, in the root element of the tree. Removing the node leaves an empty spot and disrupts the tree order, but the standardized heap algorithm can reorder the tree so that another node takes the place of the removed one. This causes subsequent nodes to bubble up, until the tree again satisfies the heap rules.

This makes the data structure ideal for an actively maintained list of the *N* largest files found so far. If the tree initially contains less than *N* nodes, all newly found files are moved into it without checking. On the other hand, if the tree grows to *N + 1* nodes when a new file is inserted, it needs to be shrunk back to *N* nodes. This works best if the algorithm removes the root node with the smallest file and lets the remaining nodes move up by repairing the tree.

Figure 4 shows the output of the test program in Listing 1. It first fills the heap with the paths to eight files including their sizes in bytes. However, it limits the maximum size of the heap to six entries. The result is the six largest files from the entire collection, sorted in descending order by file size. As expected, the `abc` and `ghi` entries with byte counts of 123 and 124 are dropped as the smallest files.

Listing 1 uses `NewTopN(6)` in line 17 to create a min heap with a maximum of six nodes. The `for` loop starting in line 18 uses `Add()` in each round to place the next entry from the `entries` array slice onto the min heap. Finally, `Iterate()` works its way through the winning entries in the data structure starting in line 22, from the file with the highest byte count to the smallest of the top six.

Chart Topper Tree

Listing 2 shows the implementation of the top *N* list using a min heap from the Go `container/heap` standard library. Its nodes are variables of the `FsEntry` type structure starting in line 7, which hold the access path to a file along with its size in bytes. The `topN` structure starting in line 12 contains an entire array slice of `FsEntry` structures in the `entries` field and the maximum size of the heap used in the `n` field.

The `NewTopN` constructor starting in line 17 creates the instance of the structure, typical of object-oriented Go programming, and then calls the `heap.Init()` heap initializer from the standard library. `NewTopN` then returns a pointer to the structure to the caller, and the caller uses the pointer as a reference to the object instance for future calls.

Listing 1: heap-main.go

```
01 package main
02 import (
03   "fmt"
04 )
05 func main() {
06   entries := []FsEntry{
07     {path: "abc", used: 123},
08     {path: "def", used: 456},
09     {path: "ghi", used: 124},
10     {path: "jkl", used: 457},
11     {path: "mno", used: 125},
12     {path: "pqr", used: 458},
13     {path: "stu", used: 126},
14     {path: "vwx", used: 999},
15   }
16
17   h := NewTopN(6)
18   for _, e := range entries {
19     h.Add(e)
20   }
21
22   h.Iterate(func(e FsEntry) {
23     fmt.Printf("%s %d\n", e.path, e.used)
24   })
25 }
```

Listing 2: heap.go

```
01 package main
02
03 import (
04   "container/heap"
05 )
06
07 type FsEntry struct {
08   path string
09   used int64
10 }
11
12 type topN struct {
13   entries []FsEntry
14   n       int
15 }
16
17 func NewTopN(n int) *topN {
18   h := &topN{n: n, entries: []FsEntry{}}
19   heap.Init(h)
20   return h
21 }
22
23 func (h *topN) Add(fse FsEntry) {
24   heap.Push(h, fse)
25   if h.Len() > h.n {
26     heap.Pop(h)
27   }
28 }
29
30 func (h *topN) Iterate(cb func(fse FsEntry)) {
31   save := make([]FsEntry, len(h.entries))
32   copy(save, h.entries)
33   r := make([]FsEntry, h.Len())
34   for i := len(r) - 1; i >= 0; i-- {
35     r[i] = heap.Pop(h).(FsEntry)
36   }
37   for _, e := range r {
38     cb(e)
39   }
40   h.entries = save
41 }
42
43 func (h topN) Len() int {
44   return len(h.entries)
45 }
46
47 func (h topN) Less(i, j int) bool {
48   return h.entries[i].used < h.entries[j].used
49 }
50
51 func (h topN) Swap(i, j int) {
52   h.entries[i], h.entries[j] = h.entries[j], h.entries[i]
53 }
54
55 func (h *topN) Push(x interface{}) {
56   h.entries = append(h.entries, x.(FsEntry))
57 }
58
59 func (h *topN) Pop() interface{} {
60   old := h.entries
61   n := len(old)
62   x := old[n-1]
63   h.entries = old[:n-1]
64   return x
65 }
```

The `Add()` function starting in line 23 takes an `FsEntry` type variable with the path to a file and its size in bytes and dumps it onto the min heap. To do this, it uses the `heap.Push()` function from the standard `container/heap` library, but then `Add()` uses `Len()` to check that this action has not grown the heap beyond its permissible length. If this has happened, `heap.Pop()` in line 26 remedies the situation by removing and discarding the entry at the root of the tree (i.e., the smallest file).

To traverse the entries in the heap, `Iterate()` starting in line 30 dismantles the heap with successive calls to `heap.Pop()` within the `for` loop starting in line 34. `Iterate()` starts with the smallest element at the root of the tree and then works its way down to the next largest. However, I want to reverse the output, starting with the largest file and ending with the smallest. This is why the `for` loop starting at line 34 stores the incoming values in reverse order in a newly created array, named `r`. The `for` loop starting in line 37 then iterates forward through the array and calls the `cb()` callback function supplied by the caller for each node found.

The caller can define what happens to the results. In the case of the test program, the function simply outputs a combination of the path and the storage usage. Because `heap.Pop()` successively dismantles the heap, destroying it in the process, `Iterate()` saves the internal data structure up front to be able to restore it at the end. To do this, `Iterate()` copies the `entries` array slice into another array using Go's internal `copy()` function, before finally restoring the heap object with it. This keeps the heap intact for subsequent calls.

Programmed Magic

Attentive readers may now be wondering how it can be that the user-defined structure `topN` type suddenly appears as a parameter in calls such as `heap.Init()` without Go, with its strict type system, complaining about the intruder. How does the standard `container/heap` library know that the array slice used in the heap is in the `entries` field of the struct, or how to sort the `FsEntry` type elements in descending order by file size?

You can find the solution to this conundrum starting in line 43 of Listing 2: The `topN` type defines the `Len()`, `Push()`, `Pop()`, `Less()`, and `Swap()` functions, satisfying the requirements of the `container/heap` interface by doing so. This means that the standard library can determine the length of the heap, append elements to the end of the internal array, or remove the array's last element – without having to know the actual implementation of the type.

`Less()` starting in line 47 determines the smaller of two given elements, while `Swap()` specifies how the heap maintenance algorithm should swap two elements. Using this toolbox, a function like `heap.Pop()` from the standard library (not to be confused with `Pop()` from Listing 2, line 59) can then later remove the node at the root of the tree, return the node, and get the tree back into shape. Talk about magic happening under the hood!

Deep Dive

Charged with this magic, Listing 3 now proceeds to recursively search the directory specified by the user at the command line. It adds up the used bytes count of the files residing within the directory hierarchy to provide a total sum and creates a min heap with the five biggest storage hogs for each direct subdirectory.

To do this, the standard `Walk()` function from the Go `filepath` library delves into the directory structure of the disk in a depth-first search (Figure 5) and calls the callback defined starting in line 18 for each entry found. The callback ignores any subdirectories and just focuses on the files. Starting in line 27, it determines the relative path to the start directory and its topmost subdirectory (`topPath`) for the currently examined file (or link). For every file found, it maintains an entry under `topPath` in the hash map that stores both the running byte total within the hierarchy and a new min hash to keep track of the biggest files.

Hard Links

By the way, this simple version of the byte counter is not totally accurate. Additional hard links – that is, additional inodes pointing to existing files – are counted twice. However, this can be corrected with a little effort. You need to create a huge table and store the hard disk number and its inode for each file already encountered. If an entry with the same values is then found, it has to be a hard link that should only be counted once.

The call to the min heap `Add()` in line 43 registers the entry that was just found. The entry will be immediately dropped by the heap unless it belongs in the top five due to its size. All told, `duDir()` starting in line 14 does all the nitty gritty work of traversing files and counting their bytes to return a hash map to the main program that then goes ahead and displays the result. In fact, the main program calling `duDir()` receives two results: first, the total number of bytes used under the analyzed directory, and second, the `duTotal` hash table, which isn't a return value of the function, but it acts as one, because the parameter is passed to the function as a pointer and is modified by the function.

For the Main Course

The main program in Listing 4 first checks whether the user has given it a directory to analyze on the command line and complains otherwise. It then calls `duDir()` from Listing 3 with an initially empty `duTotal` map that assigns `duEntry` type counters to each of the top-level directories. The function scans the hard disk and passes the compiled results back. If there are subsequently more than 10 top directories, the min heap filters out the biggest 10 starting in line 23. The subsequent `sort` function then arranges them in alphabetical order starting in line 36, before the call to `ui()` in line 40 passes the whole thing to the terminal UI for display.

Terminal Scribbles

To give users a graphical display of the data collected so far, Listing 5 then proceeds to display the data in the terminal's graphics mode using the `tview`

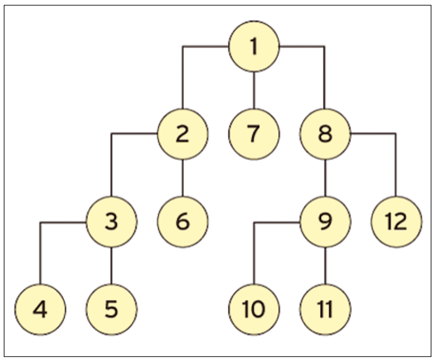


Figure 5: Depth-first tree traversal method.

framework, which I also used in last month's Programming Snapshot column [1]. If it's good enough for the Kubernetes command-line tools, it is certainly good enough for the Programming Snapshot column!

The topmost entry in the tree shown in Figure 1 is the startup directory passed to the program on the command line, along with the total space it occupies in bytes. Line 12 defines the entry and highlights it in red. The tree itself resides in the `tree` variable starting in line 14. The code adds the root node to it first. Then, further down, the `for` loop, starting in line 18, adds the top space-consuming directories that lie below it.

Listing 3: space.go

```
01 package main
02
03 import (
04   "os"
05   "path/filepath"
06   "strings"
07 )
08
09 type duEntry struct {
10   h    *topN
11   used int64
12 }
13
14 func duDir(dir string, duTotal *map[string]duEntry)
   (int64, error) {
15   total := int64(0)
16
17   err := filepath.Walk(dir,
18     func(fpath string, info os.FileInfo, err error) error {
19       if err != nil {
20         return 0, err
21       }
22
23       if info.IsDir() {
24         return nil
25       }
26
27       fpath, _ = filepath.Rel(dir, fpath)
28       parts := strings.Split(fpath, "/")
29       fileName := strings.TrimPrefix(fpath, parts[0]+"/")
30       topPath := parts[0]
31
32       // find or create new path mapping
33       var due duEntry
34       var ok bool
35       if due, ok = (*duTotal)[topPath]; !ok {
36         due = duEntry{}
37         due.h = NewTopN(5)
38       }
39
40       // add file to dir's heap
41       fse := FsEntry{path: fileName, used: info.Size()}
42       fse.used = info.Size()
43       due.h.Add(fse)
44       due.used += info.Size()
45       (*duTotal)[topPath] = due
46       total += info.Size()
47       return nil
48     })
49
50   return total, err
51 }
```

Listing 4: duview.go

```
01 package main
02
03 import (
04   "fmt"
05   "os"
06   "sort"
07   "strings"
08 )
09
10 func main() {
11   if len(os.Args) != 2 {
12     fmt.Printf("usage: %s dir\n", os.Args[0])
13     os.Exit(1)
14   }
15
16   topDir := os.Args[1]
17   duTotal := map[string]duEntry{}
18   bytes, err := duDir(topDir, &duTotal)
19   if err != nil {
20     panic(err)
21   }
22
23   topn := NewTopN(10)
24   for dir, due := range duTotal {
25     if strings.Count(dir, "/") > 1 {
26       continue
27     }
28    topn.Add(FsEntry{path: dir, used: due.used})
29   }
30
31   lines := []FsEntry{}
32   topn.Iterate(func(e FsEntry) {
33     lines = append(lines, e)
34   })
35
36   sort.Slice(lines, func(i, j int) bool {
37     return lines[i].path < lines[j].path
38   })
39
40   ui(topDir, bytes, lines, duTotal)
41 }
```

Listing 5: ui.go

```
01 package main
02
03 import (
04   "fmt"
05   "github.com/gdamore/tcell/v2"
06   "github.com/rivo/tview"
07   "golang.org/x/text/language"
08   "golang.org/x/text/message"
09 )
10
11 func ui(rootDir string, bytes int64, dirs []FsEntry, duTotal map[string]duEntry) {
12   root := tview.NewTreeNode(fmt.Sprintf("%s %s", rootDir, commify(bytes))).
13     SetColor(tcell.ColorRed)
14   tree := tview.NewTreeView().
15     SetRoot(root).
16     SetCurrentNode(root)
17
18   for _, dir := range dirs {
19     node := tview.NewTreeNode(fmt.Sprintf("%s %s", dir.path, commify(dir.used))).
20       SetExpanded(false).
21       SetSelectable(false)
22
23     // new dir
24     root.AddChild(node)
25     node.SetSelectable(true)
26     node.SetColor(tcell.ColorGreen)
27     dut, ok := duTotal[dir.path]
28     if !ok {
29       panic(fmt.Sprintf("Can't find %s", dir.path))
30     }
31
32     // add sub-menu
33     h := dut.h
34     h.Iterate(func(fse FsEntry) {
35       line := fmt.Sprintf("%s %s", fse.path, commify(fse.used))
36       n := tview.NewTreeNode(line).
37         SetExpanded(false).
38         SetSelectable(false)
39       node.AddChild(n)
40     })
41   }
42
43   tree.SetSelectedFunc(func(node *tview.TreeNode) {
44     node.SetExpanded(!node.IsExpanded())
45   })
46
47   err := tview.NewApplication().SetRoot(tree, true).EnableMouse(true).Run()
48   if err != nil {
49     panic(err)
50   }
51 }
52
53 func commify(i int64) string {
54   p := message.NewPrinter(language.English)
55   return p.Sprintf("%d", i)
56 }
```

Listing 6: Creating the Binary

```
$ go mod init duview
$ go mod tidy
$ go build duview.go space.go ui.go
    heap.go
```

`AddChild()`, which is provided by the `tview` GUI, adds each of these entries to the tree as a branch, which is made selectable by `SetSelectable()`. The user can either press the Enter key or click on the branch with the mouse. What happens on this event is defined by the call to the `SetSelectedFunc()` function further down in line 43. Its `SetExpanded()` callback sets the attribute to either true or false, depending on whether the entry has already been expanded. That way the GUI opens closed branches and closes opened ones.

To make the displayed numeric values more readable, the `commify()` function defined starting in line 53 inserts commas into the integer values, turning 123456789, for example, into the more easily readable 123,456,789. This is done using the `NewPrinter()` function from the standard `language` and `message` libraries in `golang.org/x/text/language/`.

Line 47 inserts the previously defined tree object into the terminal window controlled by `tview` and switches the terminal to graphics mode. `Run()` starts the event loop that intercepts user input and refreshes the display to reflect the incoming events. The user can interrupt the merry dance by pressing Ctrl + C, which switches the terminal back to normal text mode and lets the shell take over again.

Clearance Sale

The rule of three from Listing 6 creates the `duview` program from the four source files. If you call the resulting binary now with a directory to be analyzed, the result appears on the screen after a few seconds, depending on the utilization status, size, and performance of the hard disk. Off to the clearance sale – everything that's old has to go! ■■■

Info

[1] "Programming Snapshot – Go Network Diagnostics" by Mike Schilli, *Linux Magazine*, issue 275, October 2023, pp. 58-63

An XML, HTML, and JSON data extraction tool

Easy Extraction

Xidel lets you easily extract and process data from XML, HTML, and JSON documents. *By Marco Fioretti*

There are numerous ways to scrape a web page for data. In fact, the right mix of Python modules and Python logic glue could probably do the trick, but sometimes you just want a convenient tool that lets you extract data from websites. Xidel [1], a multi-platform command-line tool, offers a one-stop alternative to quickly extract, process, and save data from XML, HTML, or JSON documents.

Under the Hood

Xidel wraps XQuery, XPath, and JSON into one convenient front end. XQuery, a W3C Recommendation since 2007, lets you query XML or HTML files as if they were database servers, process the extracted data as desired, and save data to other files. As shown in the XQuery tutorial [2], XQuery-capable software can complete requests like finding all the CDs in an online catalog that cost less than $10, sorted by release date.

Xidel also fully supports the other W3C Recommendations, XPath [3] and the data-interchange language JavaScript Object Notation (JSON) [4]. XPath defines both a syntax for identifying all the elements of an XML document and a library of standard functions that make it easy to navigate through such elements and extract them. JSON data structures represent any kind of data as objects made of unordered sets of name/value pairs (I'll show some examples of this later on in this article).

Installation

You can download Xidel from the website [1] with just a few clicks. Xidel offers the choice between a binary package in DEB format or a ZIP archive that contains just five files: a digital certificate, the changelog, an exhaustive README file that explains in detail how Xidel works, the executable program, and its installer. The installer (Listing 1) should be run with administrator privileges. At 11 lines, the installer could hardly be simpler.

Listing 1 sets as the installation `$PREFIX` the directory passed as the first argument (line 2). On my computer, I chose the root folder (`/`), but you may prefer to use `/opt` or similar locations. Next, the script just uses the `install` program to copy the `xidel` executable and its certificate in `$PREFIX`'s `usr/bin` and, respectively, `usr/share/xidel` subdirectories.

When I tried to launch the program after running the installer, I discovered that Xidel needs the developer versions of *libopenssl* and *libcrypto* (I couldn't find this problem documented at the time of writing). However, both libraries are available as native packages in the standard repositories of most distributions (e.g., *libssl-dev* on Debian derivatives, and *openssl-devel* on Fedora-based systems), so installing them takes a matter of minutes.

Main Features

Xidel can interact with websites if it has the proper data and instructions. It can log into websites on your behalf to perform tasks like updating personal information, submitting forms, or downloading private messages. Among other things, Xidel can reach websites using proxies, manage cookies, and pause between connections to prevent overloading servers and subsequently being banned. However, I do not cover these specific Xidel features for one simple reason: Websites change all the time, so any specific examples would be completely obsolete by the time you read this article. If you want to know how Xidel can, for example, handle your Reddit notifications, I recommend first checking the latest examples on the Xidel website and then if necessary asking for support on the Xidel mailing list (which I did to write this article).

Listing 1: Installation Script

```
#!/bin/bash
PREFIX=$1
sourceprefix=
if [[ -d programs/internet/xidel/ ]]; then sourceprefix=programs/internet/
   xidel/; else sourceprefix=./;  fi
mkdir -p $PREFIX/usr/bin

install -v $sourceprefix/xidel $PREFIX/usr/bin
if [[ -f $sourceprefix/meta/cacert.pem ]]; then
mkdir -p $PREFIX/usr/share/xidel
install -v $sourceprefix/meta/cacert.pem $PREFIX/usr/share/xidel/;
fi
```

Listing 2: follow Output

```
#> xidel https://www.linux-magazine.com --follow //a --extract //title > all-titles
...

Sparkhaus Shop - Linux Magazines & Online-Services
Administration » Linux Magazine
Desktop » Linux Magazine
Web%20Development » Linux Magazine
...
...
```

As far as automatic data processing is concerned, Xidel reads and parses standard input or plain text files in JSON, XML, and HTML formats. After processing their content according to your instructions, Xidel can output the result in the same formats, as well as plain text or, as I will show later, shell variables. In addition, you can define the output separator between multiple items and create custom headers and footers for your data reports.

Xidel's two main modes, `extract` and `follow`, are often used together. In a nutshell, the `extract` mode extracts and processes data from the current document, if you just need to process the data inside one or more local files or web pages. The `follow` mode starts where `extract` leaves off by following all the links found by previous operations in order to download and process the links' content.

Xidel can run multiple `extract` and `follow` actions in the same call, as long as you write them in the right order and never ask to follow data that was not directly passed to Xidel or found by previous `extract` operations.

In `extract` mode, Xidel can recognize and select document elements by their CSS. If you want to process the extracted data, Xidel uses XPath 3.0 expressions. For more complex tasks, you can use the full XQuery standard to make Xidel run Turing-complete scripts, which StackExchange describes as "any algorithm you could think of, no matter how complex" [5].

However, when it's necessary to simultaneously extract multiple pieces of data at once, many times, from specific sections of pages with a fixed structure (e.g, titles and links of the most viewed topics in a forum), I recommend pattern matching, which I will discuss later.

Syntax-wise, as you will see in the examples I provide later, Xidel `extract` commands are one-liners that first pass to Xidel the file it should process and then, with the `--extract=` or `-e` option, a string that contains the actual operations to perform on the given document. When that string becomes so long that it's difficult to edit it on the command line, or you want to save it, you can write it to a file and pass the file to Xidel with the `--extract-file` option.

The option for the `follow` mode is `--follow=` or `-f`. As with `extract`, this option gives Xidel the expression that describes which element or sequence of elements should be followed. There are many other options for the `follow` mode, but with one exception they are almost all mirror versions of the `extract` options (e.g., you can save your commands in a file and pass it to Xidel with `--follow-file`). The exception, `--follow-level`, specifies the maximum recursion level when following pages from other pages. Set this carefully, because its default value is 99,999!

Other Options and Variables

In addition to the options that pass or configure the actual commands, `--silent` suppresses notifications, `--verbose` shows all notifications, and `--debug-arguments` shows how command-line arguments were parsed and your queries were evaluated. For support, add `--help` to get a list of all the available command-line options and `--usage` to read all the same documentation provided in the README file.

Last but not least, I want to mention global variables. Depending on your request, Xidel will provide the full input text available raw inside `$raw` or already parsed in JSON format inside `$json`. If you tell Xidel to download a web page, its full address will be inside `$url`, and its headers, host, and path will be inside `$headers`, `$host`, and `$path` respectively.

Practical Examples

Now I'll put Xidel to work and show you some practical examples. The simplest possible thing you can ask Xidel is for the title of a web page (auxiliary notifications have been omitted for readability in all examples):

```
#> xidel https://www.linux-magazine.com ⮐
  --extract //title
Linux Magazine
```

which yields the expected result. Notice that to indicate an element, in this case the title, you must prefix it with a double slash `//`. You can also give Xidel multiple commands in the same call:

```
#> xidel https://www.linux-magazine.com ⮐
  -e "//title"  https://edition.cnn.com ⮐
  --extract //title
Linux Magazine
Breaking News, Latest News and Videos ⮐
  | CNN
```

In this example, I deliberately used both versions of the `extract` command.

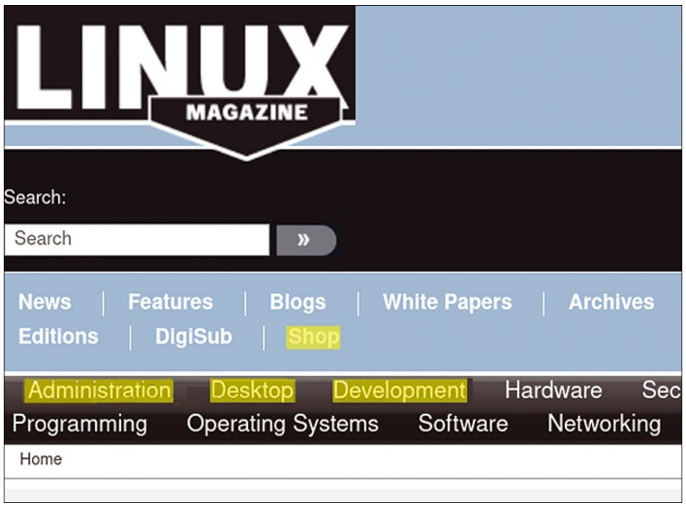

LINUX MAGAZINE
Search:
Search
News | Features | Blogs | White Papers | Archives
Editions | DigiSub | Shop
Administration Desktop Development Hardware Sec
Programming Operating Systems Software Networking
Home

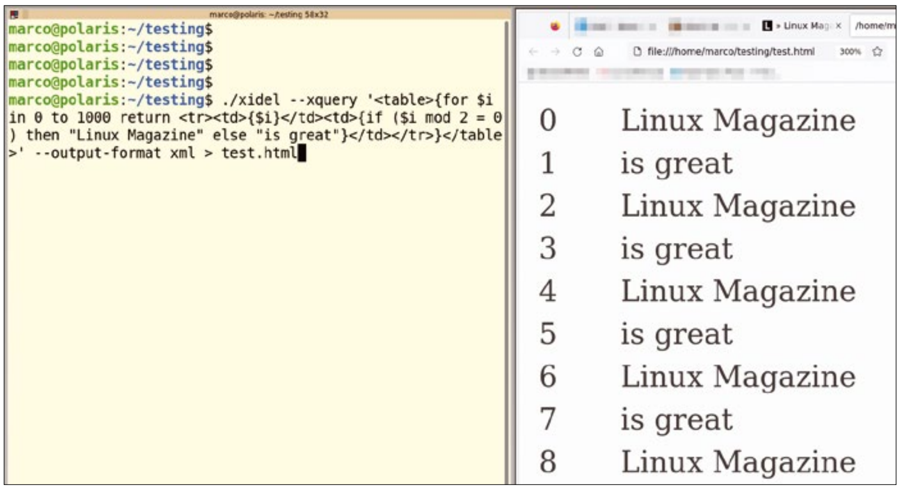
Figure 1: The titles output in Listing 2 are from the linked pages highlighted here on the *Linux Magazine* website.

Listing 2 shows the partial output of a simple `follow` command.

If you compare the four output lines in Listing 2 with the links highlighted in Figure 1, you can see that the command instructed Xidel to download all the pages linked from `www.linux-magazine.com` and then extract and print those pages' titles.

Because XQuery is Turing complete, Xidel can use XQuery to create documents from scratch, without looking anywhere for data, as shown in the following command (also shown on the left side of Figure 2):

```
#> xidel --xquery '<table>↩
  {for $i in 0 to 1000 return <tr><td>↩
  {$i}</td><td>{if ($i mod 2 = 0) ↩
  then "Linux Magazine" else "is great"}</td></tr>}</table>' ↩
  --output-format xml > test.html
```



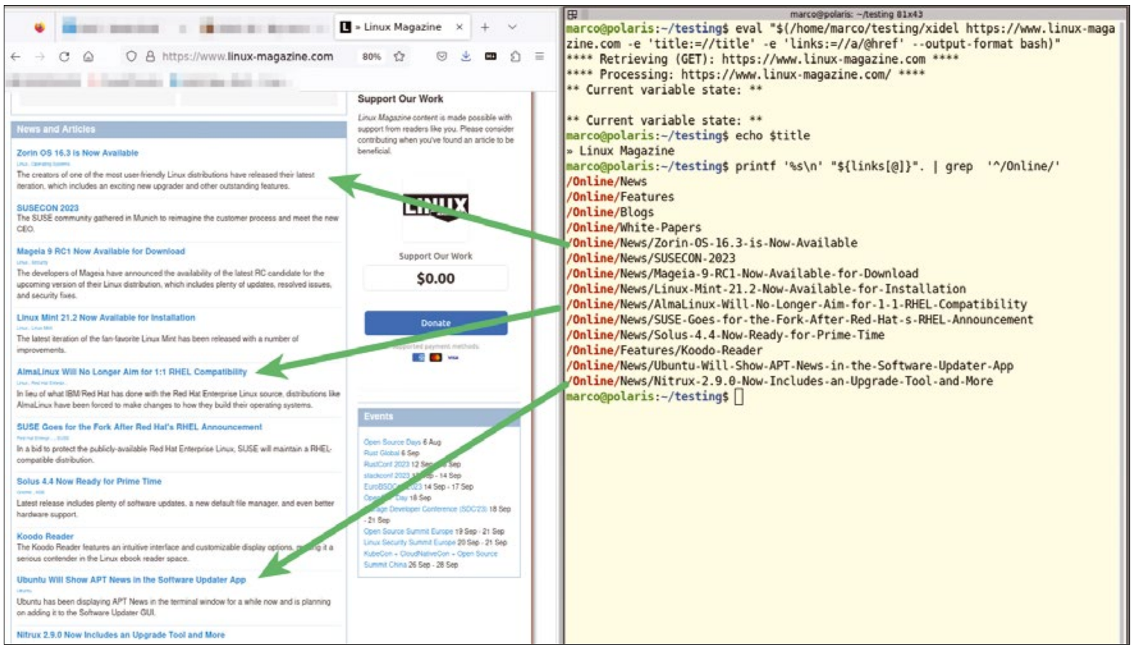
```
marco@polaris:~/testing$
marco@polaris:~/testing$
marco@polaris:~/testing$
marco@polaris:~/testing$
marco@polaris:~/testing$ ./xidel --xquery '<table>{for $i
in 0 to 1000 return <tr><td>{$i}</td><td>{if ($i mod 2 = 0
) then "Linux Magazine" else "is great"}</td></tr>}</table
>' --output-format xml > test.html
```

```
0    Linux Magazine
1    is great
2    Linux Magazine
3    is great
4    Linux Magazine
5    is great
6    Linux Magazine
7    is great
8    Linux Magazine
```

Figure 2: With Xidel, generating HTML tables or other documents is quick and simple.

This command outputs the `test.html` web page, which is the HTML table displayed by Firefox on the right side of Figure 2.

Figure 3 and Listing 3 show the `--output-format` option, which allows Xidel to influence every other Linux system component. Line 1 of Listing 3, whose output, as well as links to its sources, is shown in Figure 3, sets the Bash variable `$title` to the page's title and makes `$links` into an array of all the links on the same page. The `--output-format bash` command tells Xidel to load whatever it finds into Bash variables instead of writing the output to a file, with the specifics declared in the two `-e` options.



```
marco@polaris:~/testing$ eval "$(/home/marco/testing/xidel https://www.linux-maga
zine.com -e 'title:=//title' -e 'links:=//a/@href' --output-format bash)"
**** Retrieving (GET): https://www.linux-magazine.com ****
**** Processing: https://www.linux-magazine.com/ ****
** Current variable state: **

** Current variable state: **
marco@polaris:~/testing$ echo $title
» Linux Magazine
marco@polaris:~/testing$ printf '%s\n' "${links[@]}". | grep  '^/Online/'
/Online/News
/Online/Features
/Online/Blogs
/Online/White-Papers
/Online/News/Zorin-OS-16.3-is-Now-Available
/Online/News/SUSECON-2023
/Online/News/Mageia-9-RC1-Now-Available-for-Download
/Online/News/Linux-Mint-21.2-Now-Available-for-Installation
/Online/News/AlmaLinux-Will-No-Longer-Aim-for-1-1-RHEL-Compatibility
/Online/News/SUSE-Goes-for-the-Fork-After-Red-Hat-s-RHEL-Announcement
/Online/News/Solus-4.4-Now-Ready-for-Prime-Time
/Online/Features/Koodo-Reader
/Online/News/Ubuntu-Will-Show-APT-News-in-the-Software-Updater-App
/Online/News/Nitrux-2.9.0-Now-Includes-an-Upgrade-Tool-and-More
marco@polaris:~/testing$ 
```

Figure 3: After extracting data from a web page, Xidel can send the data everywhere, including a Linux terminal environment.

The first `-e` option (note its `:=` syntax) copies the title of the given web page (`//title`) into a shell variable, which is called `title` for clarity, but as far as the shell is concerned, it could have any other name. The second `-e` option does the same thing, using `//a/@href` to signify that all the `href` values (i.e., the actual URLs) of all the HTML anchor tags (`a`) that define HTML hyperlinks should be copied inside `links`. Because there are many such tags, `links` will automatically become an array instead of a single cell. This is why, in the `printf` statement in line 6, `links` is referenced with curly and square brackets.

Cool, huh? But enough of HTML and XML, I'll turn to some JSON examples. In a previous article in *Linux Magazine* [6], I wrote about the Shaarli bookmark manager (Figure 4), which saves its bookmarks as one JSON array with the structure shown in Listing 4 (heavily edited for clarity).

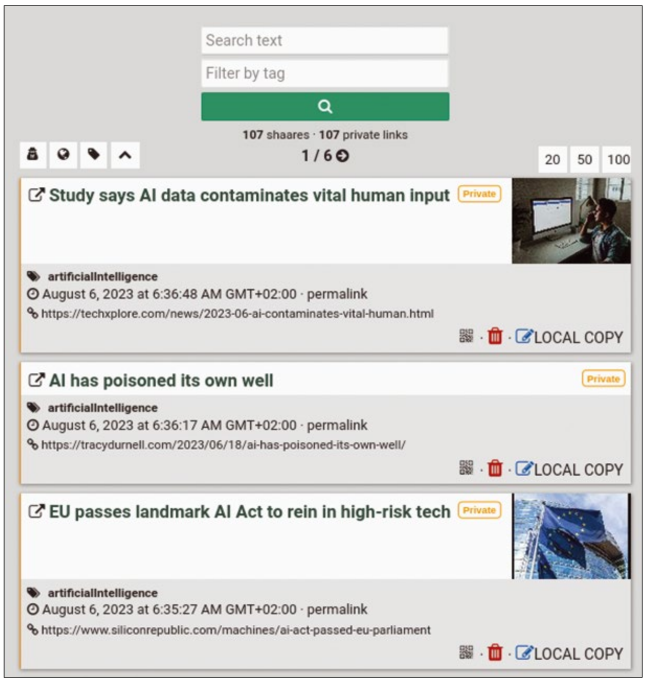
Xidel will read the `shaarli.json` file that stores the JSON array and fetch each name/value pair of every record, as long as I know its position in the array (see Listing 5).

Because XQuery is Turing complete and Xidel automatically loads every JSON file you give it into an array called `$json`, I can use the one-liner loop in Listing 6 to scan the entire array and then filter, with the `egrep` shell command, all my bookmark titles about artificial intelligence (AI).

The `for` command grabs all the `title` values inside `$json` and then loads each of them, one at a time, into the auxiliary variable `$t`. Then, the XQuery `string-join` function attaches the current title to the constant string `TITLE`, using as the connector the other string passed as the last argument.

If you want to extract more than one value per bookmark (e.g., both its ID number and title), you can use the `concat`, which unsurprisingly concatenates all the arguments it gets:

```
#> xidel -s shaarli.json ⤶
  -e '$json()/concat("ARTICLE|",id,"|",title)'
```


Search text
Filter by tag
107 shaares · 107 private links
1 / 6
20 50 100

Study says AI data contaminates vital human input — Private
artificialIntelligence
August 6, 2023 at 6:36:48 AM GMT+02:00 · permalink
https://techxplore.com/news/2023-06-ai-contaminates-vital-human.html
LOCAL COPY

AI has poisoned its own well — Private
artificialIntelligence
August 6, 2023 at 6:36:17 AM GMT+02:00 · permalink
https://tracydurnell.com/2023/06/18/ai-has-poisoned-its-own-well/
LOCAL COPY

EU passes landmark AI Act to rein in high-risk tech — Private
artificialIntelligence
August 6, 2023 at 6:35:27 AM GMT+02:00 · permalink
https://www.siliconrepublic.com/machines/ai-act-passed-eu-parliament
LOCAL COPY

Figure 4: The Shaarli bookmark manager shows some of the bookmarks retrieved by Xidel.

Listing 3: Setting Bash Environment Variables

```
01 #> eval "$(/home/marco/testing/xidel https://www.linux-magazine.com -e
   'title:=//title' -e 'links:=//a/@href' --output-format bash)"
02
03 #>  echo $title
04 Linux Magazine
05
06 #>  printf '%s\n' "${links[@]}". | grep  '^/Online/'
07
08 /Online/News
09 /Online/Features
10 /Online/Blogs
11 /Online/White-Papers
12 /Online/News/Zorin-OS-16.3-is-Now-Available
13 /Online/News/SUSECON-2023
14 /Online/News/Mageia-9-RC1-Now-Available-for-Download
15 /Online/News/Linux-Mint-21.2-Now-Available-for-Installation
16 ...
```

Listing 4: Shaarli Bookmarks in JSON Format

```
#> jq '.' shaarli.json | more
[
  ... other records...
  {
    "id": 180,
    "url": "URL of this bookmark",
    "title": "Why AI Will Save The World",
    other name/value pairs...
    ...
  },
  {
    "id": 178,
    "url": "URL of this other bookmark",
    "title": "Let Them Eat Solar Panels",
    other names/values of this other bookmark...
    ...
  }


  ... many other records...
]
```

Listing 5: Extracting a Specific Value

```
#> xidel shaarli.json -e '$json(4).title' -e '$json(8).title'
Why AI Will Save The World
Open source licenses need to evolve to deal with AI
```

Listing 6: Extracting the Bookmark Titles

```
#> xidel shaarli.json  -e 'for $t in $json/title return
   string-join(("TITLE", $t), " ==>  ")' | egrep 'AI|ntelligence'
TITLE ==>  Why AI Will Save The World
TITLE ==>  AI Is Coming For Your Children
TITLE ==>  AI has poisoned its own well
TITLE ==>  This AI Boom Will Bust
...
```

Listing 7: A Pipe-Separated, Plain Text Excerpt

```
ARTICLE|102|EU passes landmark AI Act to rein in high-risk tech
ARTICLE|134|AI has poisoned its own well
ARTICLE|135|Study says AI data contaminates vital human input
ARTICLE|176|Open source licenses need to evolve to deal with AI
ARTICLE|178|AI Is Coming For Your Children
ARTICLE|180|Why AI Will Save The World
```

Alternatively, you can use the extended string syntax (note the backticks!) as follows:

```
#>xidel -s shaarli.json ⮐
  -e '$json()/`ARTICLE|{id}|{title}`'
```

Both commands will produce the same output, part of which is visible in Listing 7. What is really important in both cases is the `$json()` notation, where the empty parentheses indicate the *entire* array, not just one of its elements. That's what makes Xidel extract and format, as shown in the second part of the command, the ID and title values of *every* bookmark.

Finally, if I want the `id` column in Listing 7 to always be four characters wide (this works because I have fewer than 10,000 bookmarks), I can tell Xidel to always pad the `id` value with enough white spaces – even for ID values smaller than 1,000 – by replacing it with the expression

```
substring(⮐
  "    ",1,4 - string-length(id))||id
```

which replaces the leftmost part of a string made of four spaces with the current value of `id`.

As far as I am concerned, the capability illustrated in these last JSON examples shows Xidel's real power, and my main reason for using it. Formats like the one shown in Listing 7 may be too limited for sophisticated uses, but it would be very hard to find a better compromise between immediate readability, ease of conversion to any other format, and (perhaps most important to me) long-term guaranteed usability, regardless of the software available.

Patterns

Xidel's support for patterns can be very useful if you need to periodically extract dynamic data from web pages with a complex but constant structure, such as an ever-changing Top 10 list on a web forum. Although a thorough presentation on Xidel patterns is outside the scope of this article, I want to briefly describe Xidel, because it may encourage others to try this program.

Listing 8 shows a snippet taken from the Xidel documentation of a pattern file that Xidel can use to fetch titles and links of the "recommended videos" listed on a YouTube page.

The `{$title:=.}` marker in the last line of Listing 8 shows the XPath syntax to tell Xidel that *every* time it finds that particular sequence of CSS elements – a list item (`li`) with CSS class `video-list-item`, followed by an anchor tag (`<a>`), followed by a `span` element whose class is `title` – then the value of that last element is data that should be saved in a variable called `$title`.

Visually, Xidel pattern files look similar to Bash here documents or Perl templates, because in all cases, you have a fixed grid, or "mask" of text, in which the elements of interest occupy a fixed place. The difference is that Bash and Perl use those tools to show where already existing variables should be placed, or written, whereas Xidel patterns do just the opposite. At their core, Xidel patterns are regular expressions, too long to fit in one line, that show where to read the values that should be saved, as well as which variables should store them.

Xidel patterns, however, are more powerful than regular expressions, with many options to control how they process what they find. For instance, a Greasemonkey script [7] can create Xidel patterns by just selecting the text to scrape on the corresponding web page.

Listing 8: Fetching with Patterns

```
< li class="video-list-item">
  <!-- skipped -->
  <span dir="ltr" class="title" title="Idras Sunhawk
     Lyrics">Idras Sunhawk Lyrics</span>
< li class="video-list-item">
  <a>
    <span dir="ltr" class="title">{$title:=.}</span>
```

Conclusion

While there are many other ways to scrape and convert XML, HTML, and JSON documents, Xidel is a multiplatform tool with almost no dependencies. It's fun, relatively easy to learn, and its mailing list is very responsive (I want to specifically thank user Reino for explaining how to process JSON arrays with Xidel).

Above all, Xidel is a simple package that can grab and reorganize data in several standard formats, from very different sources, in a very flexible, efficient way. You can even test Xidel online [8]. I recommend giving Xidel a try. ■■■

Info

[1] Xidel: *www.videlibri.de/xidel.html*

[2] XQuery tutorial: *www.w3schools.com/xml/xquery_intro.asp*

[3] XPath: *www.w3schools.com/xml/xml_XPath.asp*

[4] JSON: *www.json.org*

[5] Turing Complete explained: *https://cs.stackexchange.com/questions/71473/what-does-being-turing-complete-mean*

[6] "Making an Online Archive of All Your Bookmarked Pages" by Marco Fioretti, *Linux Magazine*, issue 232, March 2020, *https://www.linux-magazine.com/Issues/2020/232/Create-a-Personal-Web-Archive/(language)/eng-US*

[7] Greasemonkey script for Xidel patterns: *https://userscripts-mirror.org/scripts/show/144991*

[8] Xidel online demo: *www.videlibri.de/cgi-bin/xidelcgi*

Author

Marco Fioretti (*http://mfioretti.substack.com*) is a freelance author, trainer, and researcher based in Rome, Italy, who has been working with free/open source software since 1995, and on open digital standards since 2005. Marco also is a board member of the Free Knowledge Institute (*http://freeknowledge.eu*).

FOSSLIFE

Open for All

News • Careers • Life in Tech
Skills • Resources

FOSSlife.org

MakerSpace

Creating home automation devices with ESPHome

Automatic Home

With an ESP32 or Raspberry Pi Pico W microcontroller board, you can easily create your own home automation devices. Thanks to ESPHome, you don't even have to be a programmer. *By Koen Vervloesem*

Many home automation devices can be controlled through WiFi, but often these devices have limitations. For example, they might only work through the manufacturer's cloud service, they might be difficult to integrate with your own home automation system if you prefer to do everything local, they might lack advanced functionality, or they might be difficult to update.

Luckily, you can install alternative firmware on many existing or homemade devices. In this article, I introduce you to ESPHome [1], which supports numerous devices with an ESP32, ESP8266, or RP2040 microcontroller (the chip in the popular Raspberry Pi Pico W), although ESPHome support for the RP2040 is still in development. In the examples in this article, I'll use the Raspberry Pi Pico W. However, if you encounter any issues with your own projects, I recommend an ESP32 development board.

With ESPHome, you can create your own home automation devices with a supported microcontroller board that you connect to LEDs, sensors, or switches. What sets ESPHome apart from other solutions like Arduino [2] or MicroPython [3] is that you don't need to program. Instead, you configure which components are connected to which pins on the board. ESPHome then generates the necessary C++ code and compiles it into firmware that you can install on the device (see also the

Replacing Firmware on Commercial Devices

You can replace the existing firmware on commercial devices with ESPHome to gain full control over a device and use it in ways that the manufacturer hasn't anticipated. Two popular brands that have easy-to-flash devices are Shelly [4] and Sonoff [5]. A website [6] hosts more than 300 ESPHome device configuration templates that can help you get the most out of them. Note that often you'll need special hardware to flash your own firmware to these devices, at least the first time – afterward you can update them through WiFi. You'll need a USB-to-TTL adapter and to connect the pins of the adapter to the appropriate GPIO pins on the device. This isn't always a straightforward process (Figure 1).

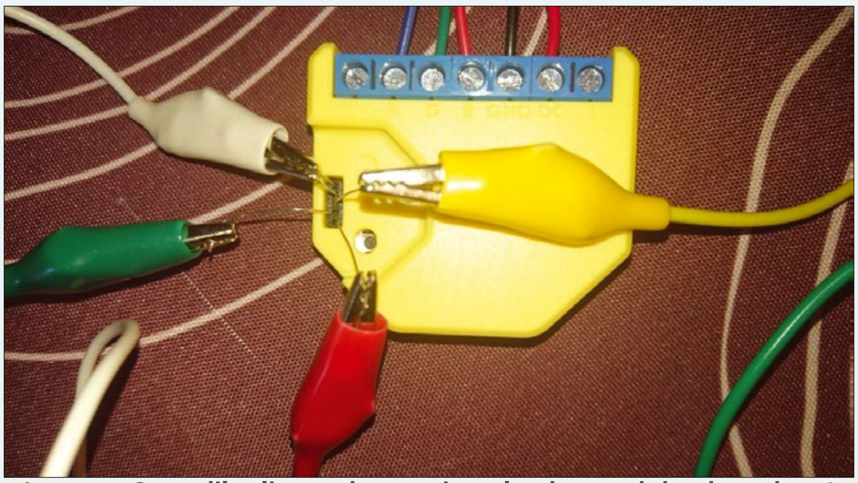
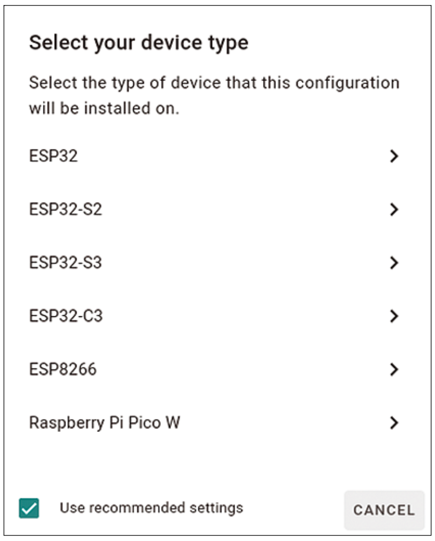


Figure 1: Crocodile clips and cut resistor leads saved the day when I wanted to flash ESPHome to this Shelly RGBW2 WiFi LED controller.



Select your device type

Select the type of device that this configuration will be installed on.

Figure 2: Select the type of device on which to install ESPHome.

"Replacing Firmware on Commercial Devices" box).

Installing ESPHome

ESPHome is a Python program, and most Linux distributions already have Python installed by default. You should first confirm that you have at least version 3.9 installed, by running the command

```
$ python --version
Python 3.9.15
```

If your Python version is older, consider upgrading your distribution, or deploy the ESPHome Docker image [7].

If the Python version looks good, create a virtual environment to contain ESPHome and its dependencies:

```
$ python -m venv esphome_venv
$ source esphome_venv/bin/activate
```

Once you're in the Python virtual environment, install the ESPHome package from PyPI:

```
$ pip install esphome
```

After the installation is complete, enter

```
$ esphome version
Version: 2023.6.5
```

to confirm that ESPHome has been installed successfully.

Creating a Project with the Dashboard

A directory in which you store all of your ESPHome projects is recommended. Suppose you call this directory `config`. To start the ESPHome dashboard and point it to this directory, run:

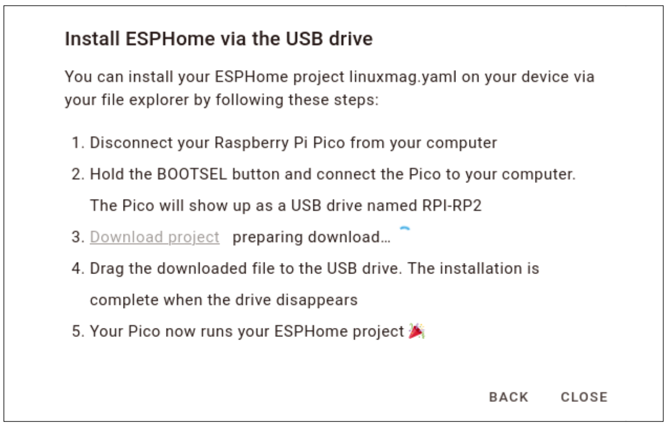
```
$ esphome dashboard config/
```

This command starts a web server on *http://0.0.0.0:6052*, which you should be able to open in your web browser. If you already have ESPHome devices on your network, the dashboard will automatically discover them.

Next, click *New Device* at the bottom right corner, and then *Continue*. Give your device a name and enter the SSID and password for the WiFi network to which you want your device to connect, then click *Next* and select your device type (Figure 2).

In this example, choose *Raspberry Pi Pico W*; for an ESP32 or ESP8266 you also need to select the specific board. The dashboard then creates a minimal configuration and shows an encryption key that you can use to allow the ESPHome device to communicate with Home Assistant [8], a popular open source home automation gateway developed by the same team behind ESPHome. Finally, click *Install*.

You can use several methods to install ESPHome to your device, but not all of them are supported by every device. Because no ESPHome firmware is running on the device yet, the first method (over WiFi) is not possible; the *Plug into the computer running ESPHome Dashboard* choice isn't available either. You can always choose *Manual download*, which has instructions on how to accomplish the installation (Figure 3).



Install ESPHome via the USB drive

You can install your ESPHome project linuxmag.yaml on your device via your file explorer by following these steps:

1. Disconnect your Raspberry Pi Pico from your computer
2. Hold the BOOTSEL button and connect the Pico to your computer. The Pico will show up as a USB drive named RPI-RP2
3. Download project preparing download...
4. Drag the downloaded file to the USB drive. The installation is complete when the drive disappears
5. Your Pico now runs your ESPHome project 🎉

Figure 3: The ESPHome dashboard has excellent instructions for every step of the installation.

Listing 1: Pi Pico W Default Config

```
esphome:
  name: linuxmag
  friendly_name: linuxmag

rp2040:
  board: rpipicow
  framework:
    # Required until https://github.com/platformio/
      platform-raspberrypi/pull/36 is merged
    platform_version: https://github.com/maxgerhardt/
      platform-raspberrypi.git

# Enable logging
logger:

# Enable Home Assistant API
api:
  encryption:
    key: "7wFO19sSMAkGX008l+wX4u53hBz/a1Ha+9bAdouUjo8="

ota:
  password: "20e3778465f1c5b147f8645dc237b146"

wifi:
  ssid: !secret wifi_ssid
  password: !secret wifi_password

  # Enable fallback hotspot in case wifi connection fails
  ap:
    ssid: "Linuxmag Fallback Hotspot"
    password: "DVaDAPFJN5cA"
```

For the Raspberry Pi Pico W, you'll need to disconnect the board from USB, hold the *BOOTSEL* button while reconnecting the board, and then release the button, which causes a USB drive named RPI-RP2 to appear in your file manager. Now, click *Download project* and drag the `.uf2` file to the USB drive. Once the drive disappears, the board runs your ESPHome firmware, and you can click *Close*.

Default ESPHome Configuration

In the ESPHome dashboard, click *Edit* in the box representing your device to open your device configuration in a web editor. The configuration file is written in YAML [9], with various key-value pairs for different options (Listing 1).

As you can see, this configuration file sets the device name and its friendly name, as well as the platform and board. It then enables logging and the Home Assistant API, sets a password for over-the-air (OTA) updates, and configures WiFi credentials and a fallback hotspot in case the WiFi connection fails. If a failure happens, you can connect with your mobile phone to the hotspot of the device to reconfigure the network. The WiFi credentials are stored in a separate file, `secrets.yaml`, which prevents accidental exposure of sensitive information when sharing your device configuration with others.

Note that if you don't use Home Assistant, you should remove the `api` line and the two lines that follow; otherwise, your ESPHome device keeps waiting for a connection from Home Assistant. If no connection is established within 15 minutes, the device will assume that something's wrong and reboot.

Listing 2: Blinking LED

```
output:
  - platform: gpio
    pin:
      number: 32
      mode: output
    id: LED

interval:
  - interval: 1000ms
    then:
      - output.turn_on: LED
      - delay: 500ms
      - output.turn_off: LED
```

Blinking the Built-In LED

Now you can modify this configuration in the web editor or in your favourite desktop or command-line editor. The configuration file is saved in the `config` directory you created. In the next exercise, make the board's built-in LED blink by adding the configuration shown in Listing 2.

Make sure to use the correct indentation because spaces are important in YAML. This configuration adds an `output` component from the `gpio` platform on pin `32`, which corresponds to the built-in LED on the Raspberry Pi Pico W. Additionally, an `interval` component is defined that is triggered every `1000ms`. On each trigger, it turns on the output with `id: LED`, waits `500ms`, and then turns off the same output.

After saving the file (in the web editor at the top right), click *Install* in the dropdown menu of the node and choose your installation method. This time you can choose *Wirelessly*, because your device is already running ESPHome and is connected to your WiFi network. Your device doesn't even need to be connected to your computer's USB port any more. Your YAML configuration is now transformed into C++ code and compiled. If you see the message *INFO Successfully compiled program*, the dashboard will upload the new firmware. Once your device reboots, the LED starts blinking.

Adding Your Device to Home Assistant

If you're running Home Assistant on your home network, your ESPHome device will be recognized automatically. In your Home Assistant dashboard, click *Notifications* in the sidebar and then *Check it out* at the *New devices discovered* message. You will see the name you

Listing 3: Remote LED Control

```
switch:
  - platform: gpio
    pin:
      number: 32
      mode: output
    name: LED
```

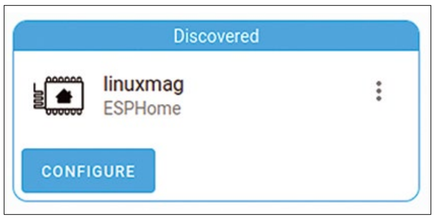


Figure 4: Home Assistant automatically discovers ESPHome devices on your network.

assigned to your ESPHome device (Figure 4). Click the *Configure* button and then *Submit* to add the ESPHome device to Home Assistant.

You will be asked to enter the device's encryption key. Go to the ESPHome dashboard and find the key in the device's YAML code. Alternatively, click on the three dots in the box representing your device, then *Show API Key*. Next, click *Copy* and paste the key in the *Encryption key* field of Home Assistant. After clicking *Submit*, optionally choosing an area, click *Finish* to complete the process. The device is added to Home Assistant.

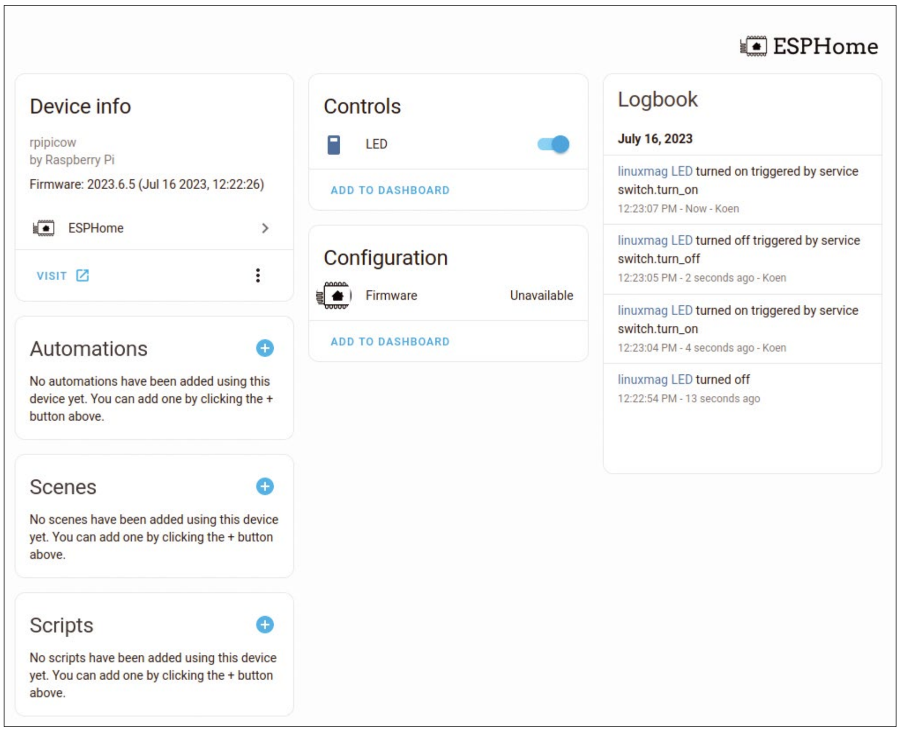
Remotely Control the LED

Now that you have configured your device to blink its LED and added it to Home Assistant, you might want to control it remotely. Instead of having the LED blink automatically, modify the configuration to allow you to control the LED from Home Assistant's dashboard. The required changes are simple: In the YAML configuration file, remove the entire `interval` block, change the `output` key to `switch`, and change the `id` key to `name`. You can find the modified configuration (without the defaults that you leave unchanged) in Listing 3.

After installing the firmware on your device, you can control the built-in LED from Home Assistant's dashboard (Figure 5). Because you have defined the LED as a `switch` component, you can turn it on and off.

Adding Sensors

Controlling a simple LED is relatively easy with alternatives like an Arduino sketch or some MicroPython code. However, things become more complex when you start connecting sensors. This is where ESPHome shines. The ESPHome website provides documentation for



ESPHome

Device info

rpipicow
by Raspberry Pi
Firmware: 2023.6.5 (Jul 16 2023, 12:22:26)

ESPHome

VISIT

Automations

No automations have been added using this device yet. You can add one by clicking the + button above.

Scenes

No scenes have been added using this device yet. You can add one by clicking the + button above.

Scripts

No scripts have been added using this device yet. You can add one by clicking the + button above.

Controls

LED

ADD TO DASHBOARD

Configuration

Firmware Unavailable

ADD TO DASHBOARD

Logbook

July 16, 2023

linuxmag LED turned on triggered by service switch.turn_on
12:23:07 PM - Now - Koen

linuxmag LED turned off triggered by service switch.turn_off
12:23:05 PM - 2 seconds ago - Koen

linuxmag LED turned on triggered by service switch.turn_on
12:23:04 PM - 4 seconds ago - Koen

linuxmag LED turned off
12:22:54 PM - 13 seconds ago

Figure 5: Control the LED on your ESPHome device from within Home Assistant.

various supported sensors [13], each with simple YAML examples.

Take the MH-Z19 CO2 sensor, for example. This sensor is useful to have at home, especially considering the importance of ventilation in the fight against viruses. The worse the ventilation in a home, the higher the concentration of CO2. Not only is CO2 concentration – [CO2] – a good indication of the need for ventilation, it has various health hazards on its own.

Connecting the CO2 Sensor

The ESPHome documentation provides instructions on how to connect and configure the MH-Z19 sensor [14]. First, you disconnect your Raspberry Pi Pico W from power and put it on a breadboard. You will be using four pins on the MH-Z19: VIN, GND, RX, and TX. Their names are listed on the bottom of the sensor board. Additionally, consult the Raspberry Pi Pico W pinout [15] or the pinout of the other microcontroller board you're using.

Connect the VIN pin of the sensor to the VBUS (which receives 5V from the USB power supply) of the Raspberry Pi Pico W, GND to GND, RX to GP4, and TX to GP5 (Figure 6). The configuration for the sensor is shown in Listing 4.

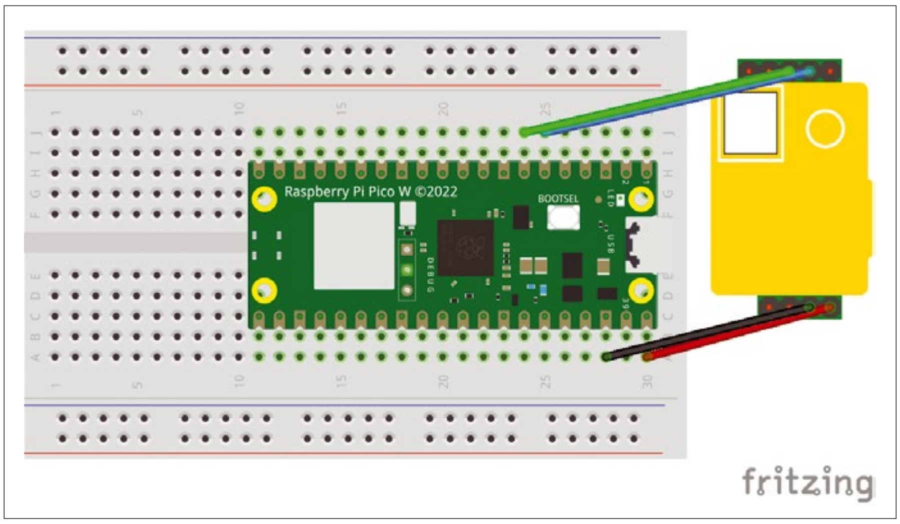


Figure 6: Connect the MH-Z19 CO2 sensor to the Raspberry Pi Pico W.

In this configuration, you define a UART bus on pins GP4 and GP5, operating at 9600 baud. Note that the RX defined here is connected to the TX of the sensor, and vice versa. The configuration also defines the CO2 sensor, which measures both [CO2] and temperature and sets the update interval to once per minute. You can remove the switch for the LED from the configuration because you don't need it here.

After installing this configuration, you'll see the current CO2 concentration in parts per million (ppm) appearing in Home Assistant and in the logs on the ESPHome dashboard. Be sure to read the ESPHome documentation on calibrating the sensor to ensure accurate measurements. Note that the internal temperature sensor of the MH-Z19B isn't that accurate; it's primarily used as a reference for the CO2 sensor.

Automated CO2 Alarm

You can now read the CO2 values in your Home Assistant dashboard (Figure 7), but you might not always be looking at your computer or phone screen. Fortunately, we already know how to control the built-in LED. In principle you can now add an automation in Home Assistant that turns on the LED on your Raspberry Pi Pico W when the CO2 level is too high. However, this detour is not necessary; you can achieve the same result with ESPHome's built-in automation features. The advantage is that these automations continue to work even when your Home Assistant installation or MQTT broker (see the "MQTT Broker" box) is offline or your network connection is down.

To make your Raspberry Pi Pico W board act as a CO2 alarm, turn on the

Listing 4: CO2 Sensor over UART

```
uart:
  rx_pin: 5
  tx_pin: 4
  baud_rate: 9600

sensor:
  - platform: mhz19
    co2:
      name: "MH-Z19 CO2"
    temperature:
      name: "MH-Z19 Temperature"
    update_interval: 60s
```

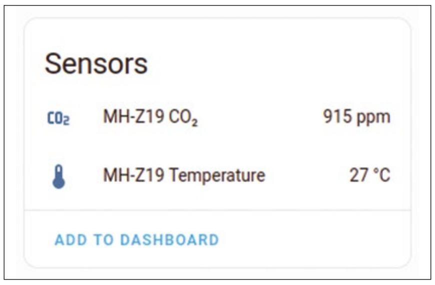


Figure 7: Your CO2 sensor is visible in Home Assistant.

built-in LED when the CO2 concentration exceeds 1,000ppm (Listing 5).

This code first defines the LED as an output, rather than a switch, so it's not controllable through Home Assistant or MQTT. The `id` lets you refer to the LED in the configuration of the CO2 sensor.

Next, it modifies the `co2` section in the CO2 sensor configuration, which configures the sensor to turn on the LED with ID `co2_alarm` when the CO2 value exceeds 1,000ppm and turns it off when the CO2 value drops below 1,000ppm. Your CO2 alarm will now function even without a network connection. When you see the built-in LED turn on, you'll know that it is time to open the windows for ventilation.

Listing 5: CO2 Sensor with LED Alarm

```
output:
  - platform: gpio
    pin: 32
    id: co2_alarm

uart:
  rx_pin: 5
  tx_pin: 4
  baud_rate: 9600

sensor:
  - platform: mhz19
    co2:
      name: "MH-Z19 CO2"
      id: co2_value
      on_value_range:
        - above: 1000
          then:
            - output.turn_on: co2_alarm
        - below: 1000
          then:
            - output.turn_off: co2_alarm
    temperature:
      name: "MH-Z19 Temperature"
    update_interval: 60s
```

Adding a Display

If you want more than just an LED on your device and prefer to see the full sensor value, you can add a display. ESPHome supports various display components [16], and this example uses an SSD1306 [17]. Start by disconnecting power from your Raspberry Pi Pico W and then connecting the display to your breadboard. Connect it as follows: the VCC of the display to 3V3 Out of the Pico W, GND to GND, SDA to GP8, and SCL to GP9 (Figure 8). Listing 6 shows the code you need to add to Listing 5 to show the sensor values on the display.

The SSD1306 display uses the I2C bus (there's also an SPI version), so first define this bus in the ESPHome configuration. If you use other pin numbers on the Raspberry Pi Pico W, make sure to use pins that are designated I2C0 in the pinout, because I2C1 isn't supported yet by ESPHome.

Because you want to show letters and numbers on the display, you also need to define a font. This example uses Roboto font size 18. The display is now defined and uses the previously defined I2C bus, specifying the model and I2C address of the display. The last line is the first real code in this article. This `lambda` is C++ code integrated in your ESPHome configuration. Although most ESPHome configurations can be defined without programming, displays are one of the few components that require code.

In this example, the one-liner calls the `printf` method on the display, specifying the horizontal and vertical coordinates in which to place the text, the `id` of the font, the string template to display, and the state you want to display. The value is obtained by referring to the ID of the

MQTT Broker

If you don't use Home Assistant – or if you prefer not to use the Home Assistant API – ESPHome also supports communication through the MQTT machine-to-machine messaging protocol [10]. This route requires you to set up an MQTT broker, such as Eclipse Mosquitto [11], running in a Docker container or as a Home Assistant add-on. Then you add an MQTT Client Component [12] to your ESPHome device configuration, specifying which MQTT broker to connect to, as well as the username and password for authentication. At this point, you can control the switches defined in your device by sending MQTT messages to your broker, and you can subscribe to MQTT messages from the sensors defined in your device.

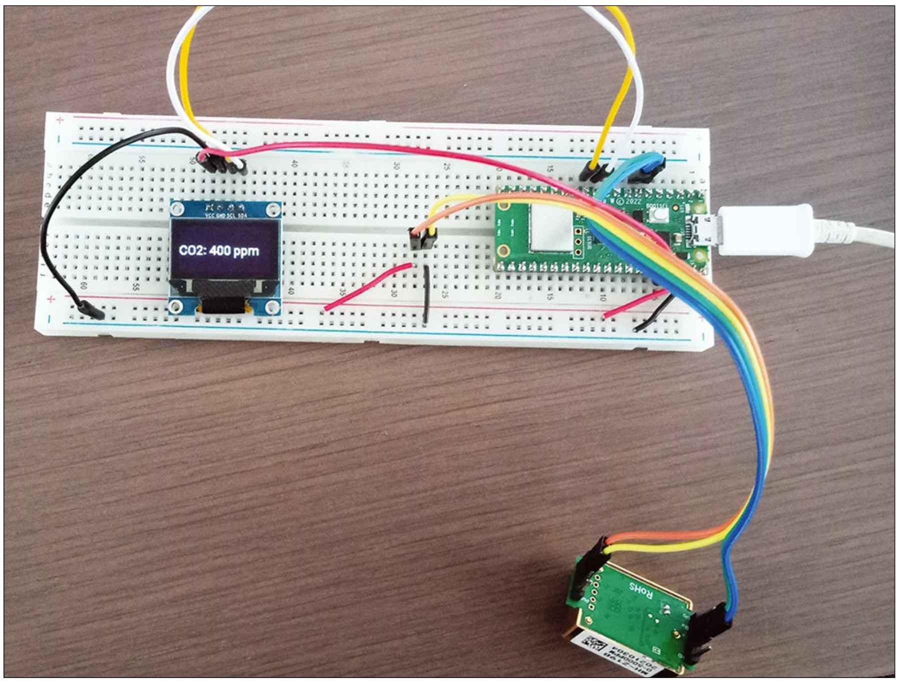


Figure 8: On a real breadboard, a circuit always looks messier than in a diagram.

CO2 sensor and getting its `state` property:

```
id(co2_value).state
```

Because this is a floating-point number (e.g., 1746.00), the `%.0f` pattern shows only the integer part (i.e., 1746).

Complex Devices

The full YAML file of this CO2 sensor device isn't very long, but you can create much more complex configurations. For example, I created an ESPHome air quality monitor [18] that combines a CO2 sensor with a particulate matter (PM) sensor, temperature-humidity-pressure sensor, and display. I've also created an ESPHome configuration for the M5Stack PM2.5 air quality kit [19], as well as an ESPHome heart rate display [20] showing heart rate from a Bluetooth Low Energy (BLE) heart rate sensor on a display.

Finally, if you want to learn more about ESPHome and explore various examples to create your own home automation devices, read my book on the topic [21]. ■■■

Listing 6: Display Showing [CO2]

```
i2c:
  sda: 8
  scl: 9

font:
  - file: "gfonts://Roboto@medium"
    id: font_roboto
    size: 18

display:
  - platform: ssd1306_i2c
    model: "SSD1306 128x64"
    address: 0x3C
    lambda: |-
      it.printf(0, 23, id(font_roboto), "CO2: %.0f ppm", id(co2_value).state);
```

Author

Koen Vervloesem has been writing about Linux and open source, computer security, privacy, programming, artificial intelligence, and the Internet of Things for more than 20 years. You can find more on his website at *koen.vervloesem.eu*.

Info

[1] ESPHome: *https://esphome.io*

[2] Arduino: *https://www.arduino.cc*

[3] MicroPython: *https://micropython.org*

[4] Shelly: *https://shelly.cloud*

[5] Sonoff: *https://sonoff.tech*

[6] ESPHome devices: *https://devices.esphome.io*

[7] ESPHome Docker image: *https://esphome.io/guides/getting_started_command_line.html#installation*

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

[9] YAML: *https://yaml.org*

[10] MQTT: *https://mqtt.org*

[11] Eclipse Mosquitto: *https://mosquitto.org*

[12] MQTT Client Component: *https://esphome.io/components/mqtt.html*

[13] ESPHome sensor components: *https://esphome.io/index.html#sensor-components*

[14] ESPHome MH-Z19 sensor: *https://esphome.io/components/sensor/mhz19.html*

[15] Raspberry Pi Pico W pinout: *https://picow.pinout.xyz*

[16] ESPHome display components: *https://esphome.io/index.html#display-components*

[17] ESPHome SSD1306 display: *https://esphome.io/components/display/ssd1306.html*

[18] ESPHome air quality monitor: *https://github.com/koenvervloesem/ESPHome-Air-Quality-Monitor*

[19] Config for M5Stack PM2.5 air quality kit: *https://github.com/koenvervloesem/M5Stack-Air-Quality-ESPHome*

[20] ESPHome heart rate display: *https://github.com/koenvervloesem/ESPHome-Heart-Rate-Display*

[21] Vervloesem, Koen. *Getting Started with ESPHome*. Elektor International Media B.V., 2021, *https://www.elektor.com/getting-started-with-esphome*

■■■

MakerSpace

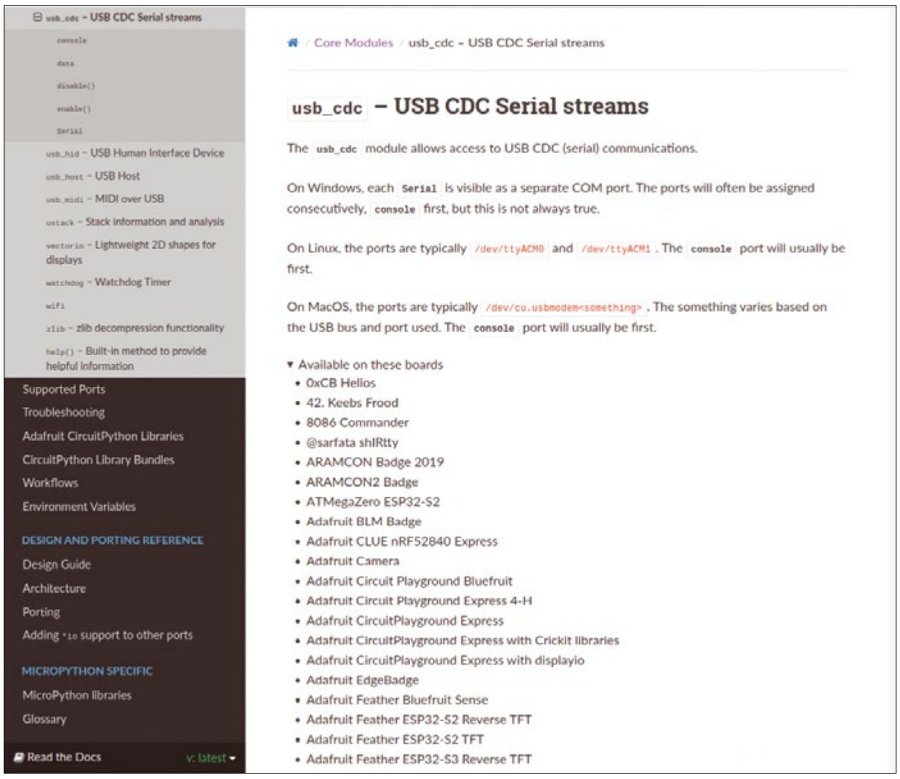
Controlling microcontrollers over USB with the Web Serial API

Under Control

Upgrade your computer with LEDs, buttons, or sensors to control a microcontroller board over USB from your web browser. *By Koen Vervloesem*

Microcontroller boards such as Arduino, Raspberry Pi Pico, and ESP32 can be connected to various LEDs, buttons, and sensors. Many of these boards have a built-in WiFi chip, which allows for remote control. However, sometimes WiFi is not possible, too complicated, or simply unnecessary.

Fortunately, most microcontroller boards for makers are equipped with a universal serial bus (USB) connection, which can often be used to send commands to the microcontroller from your computer or to receive sensor data. In this article, I explore what you need to establish two-way communication over USB, and I guide you through writing the necessary code for both the microcontroller and the computer.



⊟ usb_cdc – USB CDC Serial streams
- console
- data
- disable()
- enable()
- Serial

usb_hid – USB Human Interface Device
usb_host – USB Host
usb_midi – MIDI over USB
ustack – Stack information and analysis
vectorio – Lightweight 2D shapes for displays
watchdog – Watchdog Timer
wifi
zlib – zlib decompression functionality
help() – Built-in method to provide helpful information
Supported Ports
Troubleshooting
Adafruit CircuitPython Libraries
CircuitPython Library Bundles
Workflows
Environment Variables
DESIGN AND PORTING REFERENCE
Design Guide
Architecture
Porting
Adding *io support to other ports
MICROPYTHON SPECIFIC
MicroPython libraries
Glossary
Read the Docs v: latest

/ Core Modules / usb_cdc – USB CDC Serial streams

usb_cdc – USB CDC Serial streams

The `usb_cdc` module allows access to USB CDC (serial) communications.

On Windows, each `Serial` is visible as a separate COM port. The ports will often be assigned consecutively, `console` first, but this is not always true.

On Linux, the ports are typically `/dev/ttyACM0` and `/dev/ttyACM1`. The `console` port will usually be first.

On MacOS, the ports are typically `/dev/cu.usbmodem<something>`. The something varies based on the USB bus and port used. The `console` port will usually be first.

▾ Available on these boards
- 0xCB Helios
- 42. Keebs Frood
- 8086 Commander
- @sarfata shIRtty
- ARAMCON Badge 2019
- ARAMCON2 Badge
- ATMegaZero ESP32-S2
- Adafruit BLM Badge
- Adafruit CLUE nRF52840 Express
- Adafruit Camera
- Adafruit Circuit Playground Bluefruit
- Adafruit Circuit Playground Express 4-H
- Adafruit CircuitPlayground Express
- Adafruit CircuitPlayground Express with Crickit libraries
- Adafruit CircuitPlayground Express with displayio
- Adafruit EdgeBadge
- Adafruit Feather Bluefruit Sense
- Adafruit Feather ESP32-S2 Reverse TFT
- Adafruit Feather ESP32-S2 TFT
- Adafruit Feather ESP32-S3 Reverse TFT

Figure 1: Check the CircuitPython documentation to see whether your microcontroller board supports USB CDC.

USB CDC

To enable communication between the microcontroller and computer, this project uses USB communications device class (CDC). When you connect a USB CDC device to your computer, this interface appears in your Linux system as a device, like `/dev/ttyACM0`. With this device file, software on your computer can communicate with the microcontroller.

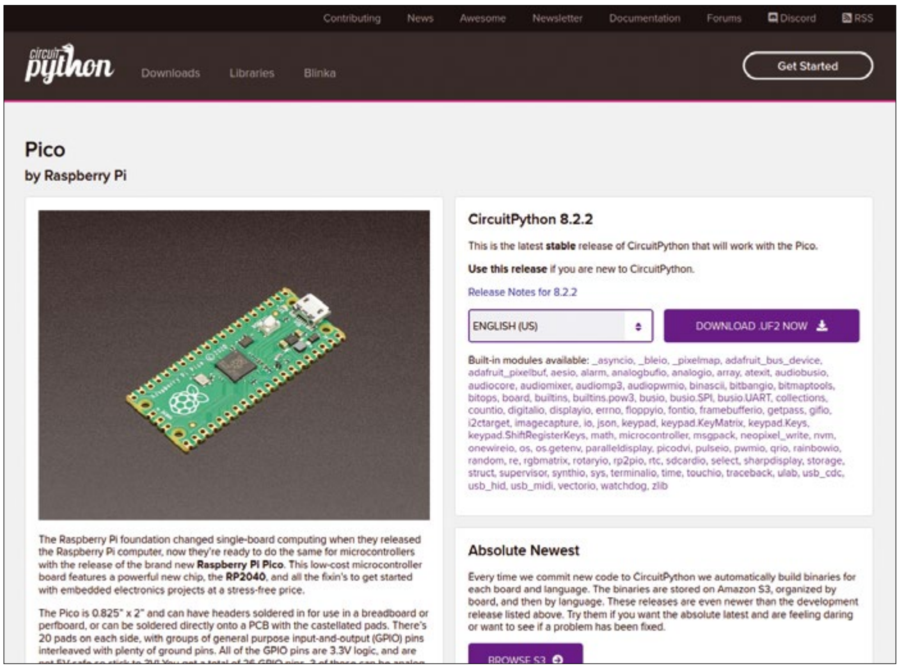
Therefore, you need to establish a serial connection over USB CDC on the microcontroller. In this article, I do this with CircuitPython [1], which supports hundreds of microcontroller boards. Make sure to choose a board with USB CDC support [2] (Figure 1). I have successfully tested this project with the Raspberry Pi Pico (W), Arduino Nano RP2040 Connect, Seeed Studio XIAO SAMD21, and Seeed Studio XIAO nRF52840. For other boards, you might need to modify the CircuitPython code slightly or install the firmware differently.

Installing CircuitPython

Note that you don't need a board with WiFi or Bluetooth Low Energy (BLE) connectivity for serial communication, so you can choose a more affordable board, such as the Raspberry Pi Pico (the version without WiFi), which I use for the examples in this article. To download the appropriate CircuitPython firmware, visit the CircuitPython Downloads page [3] and choose the link for your board.

At the time of writing, the Raspberry Pi Pico [4] version was 8.2.2. You can choose your language, but that doesn't matter much. On the device's download page (Figure 2), you'll see that the list of built-in modules includes `usb_cdc`, which confirms that you can use USB CDC on this board.

For the Raspberry Pi Pico, the downloaded firmware file has the `.uf2` extension. To install the firmware, press and hold the white BOOTSEL button on the Raspberry Pi Pico, connect the board to your computer with a micro-USB cable, and then release the button. The board's internal storage will now appear as a USB drive named RPI-RP2 on your computer. Drag and drop the `.uf2` file onto that drive. After copying, the drive's name will change to CIRCUITPY.



Pico
by Raspberry Pi

CircuitPython 8.2.2

This is the latest **stable** release of CircuitPython that will work with the Pico.

Use this release if you are new to CircuitPython.

Release Notes for 8.2.2

ENGLISH (US)

Built-in modules available: _asyncio, _bleio, _pixelmap, adafruit_bus_device, adafruit_pixelbuf, aesio, alarm, analogbufio, analogio, array, atexit, audiobusio, audiocore, audiomixer, audiomp3, audiopwmio, binascii, bitbangio, bitmaptools, bitops, board, builtins, builtins.pow3, busio, busio.SPI, busio.UART, collections, countio, digitalio, displayio, errno, floppyio, fontio, framebufferio, getpass, gifio, i2ctarget, imagecapture, io, json, keypad, keypad.KeyMatrix, keypad.Keys, keypad.ShiftRegisterKeys, math, microcontroller, msgpack, neopixel_write, nvm, onewireio, os, os.getenv, paralleldisplay, picodvi, pulseio, pwmio, qrio, rainbowio, random, re, rgbmatrix, rotaryio, rp2pio, rtc, sdcardio, select, sharpdisplay, storage, struct, supervisor, synthio, sys, terminalio, time, touchio, traceback, ulab, usb_cdc, usb_hid, usb_midi, vectorio, watchdog, zlib

The Raspberry Pi foundation changed single-board computing when they released the Raspberry Pi computer, now they're ready to do the same for microcontrollers with the release of the brand new **Raspberry Pi Pico**. This low-cost microcontroller board features a powerful new chip, the **RP2040**, and all the fixin's to get started with embedded electronics projects at a stress-free price.

The Pico is 0.825" x 2" and can have headers soldered in for use in a breadboard or perfboard, or can be soldered directly onto a PCB with the castellated pads. There's 20 pads on each side, with groups of general purpose input-and-output (GPIO) pins interleaved with plenty of ground pins. All of the GPIO pins are 3.3V logic, and are

Absolute Newest

Every time we commit new code to CircuitPython we automatically build binaries for each board and language. The binaries are stored on Amazon S3, organized by board, and then by language. These releases are even newer than the development release listed above. Try them if you want the absolute latest and are feeling daring or want to see if a problem has been fixed.

Figure 2: Download the CircuitPython firmware file for Raspberry Pi Pico.

Mu Editor

The easiest way to program your board in CircuitPython is with the code editor Mu [5]. You can download a TAR archive of an AppImage for Linux, currently at version 1.2.0. To untar the AppImage and make the file executable, run the following command:

```
tar xf MuEditor-Linux-1.2.0-x86_64.tar
chmod + x Mu_Editor-1.2.0-x86_64.AppImage
```

On most Linux distributions, you can now simply double-click the AppImage file in your favorite graphical file manager to start Mu. Detailed installation instructions for Mu [6] for various distributions can be found on the website.

After launching Mu, click *Mode* at the top left, choose *CircuitPython* from the list, and click *OK*. If you connect your board to your computer with a USB cable, Mu should detect it, displaying the message *Detected new CircuitPython device* at the bottom.

Now you can type the code you want to run on your board in the large text field. To test the hardware to make sure it is working properly, enter the code in Listing 1, which makes the built-in LED blink.

At the top, click *Save*, select *code.py* (Figure 3), and confirm that you want to overwrite it. You should now see the built-in LED on the Raspberry Pi Pico blinking, confirming that the board is working.

Defining a Protocol

Before programming the Pico, it's important to establish a communication protocol between the microcontroller and the computer. Such a protocol is essentially a set of agreements for communication. What kind of communication does the microcontroller expect, and how does it respond to different commands? For the purpose of this example, I'll keep the protocol simple. Each command sent to the microcontroller over USB consists of a single character that corresponds to an action to be performed on the built-in LED, as listed in Table 1.

After executing each command, the microcontroller should reply to the computer with the current state of the LED: *0* if the LED is off, and *1* if the LED is on.

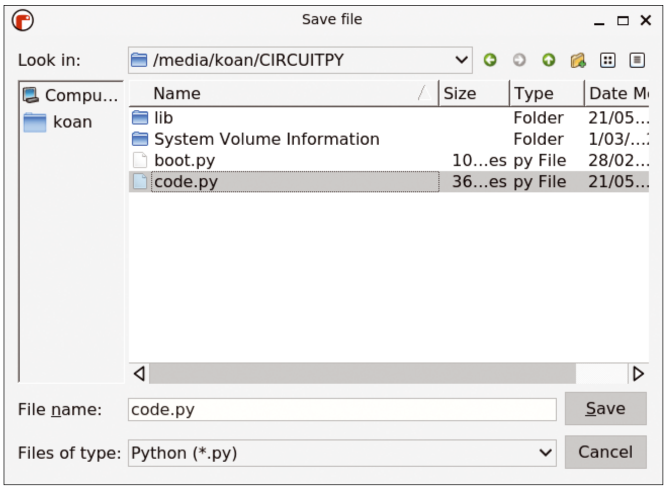
Finally, note that I'm talking about characters here (letters, or in this case

Listing 1: Blink the LED

```
"""Make the built-in LED blink."""
from board import LED
from digitalio import DigitalInOut, Direction
from time import sleep

led = DigitalInOut(LED)
led.direction = Direction.OUTPUT

while True:
    led.value = True
    sleep(0.5)
    led.value = False
    sleep(0.5)
```



Look in: /media/koan/CIRCUITPY

Name	Size	Type	Date M...
lib		Folder	21/05...
System Volume Information		Folder	1/03/...
boot.py	10...es	py File	28/02...
code.py	36...es	py File	21/05...

File name: code.py

Files of type: Python (*.py)

Figure 3: Save your CircuitPython code to the Raspberry Pi Pico with Mu.

Table 1: Serial Communication Protocol

Character	Action
0	Turn off LED
1	Turn on LED
2	Toggle LED
3	Flash LED briefly

numbers), whereas serial communication operates on bytes. Therefore, the characters need to be converted to the corresponding bytes on both the computer and the microcontroller. For example, the character *1* is represented by 0x31 in hexadecimal notation.

Waiting for Commands

Now that the communication protocol between the microcontroller and computer is defined, you can program the CircuitPython code accordingly. As a first step, create a new file in Mu and add:

```
import usb_cdc

# Enable console-over-serial and ↩
  data-over-serial
usb_cdc.enable(↩
  console=True, data=True)
```

This code enables data transfer over USB CDC.

By default, CircuitPython only uses USB CDC for the REPL (read-eval-print loop), which allows you to run Python commands over the USB connection (`console=True`). Adding `data=True` also sets up USB CDC for your own data channel.

If you save this file as `boot.py`, CircuitPython will execute it immediately after booting the microcontroller and before running any other code. Next, replace the existing code in `code.py` with the code in Listing 2.

This code starts by obtaining the serial object for the data, `usb_cdc.data`, and storing it in the variable `serial`. In an infinite loop (`while True`), it checks to see whether any data has come in on the serial interface from the computer. If it has, it reads one byte with `serial.read(1)` (line 15), then compares this byte with the values defined in the protocol.

Because the communication protocol deals with bytes, the program needs to convert the characters to bytes. For example, the code compares the received byte with `b"0"`, which is the byte representation of character *0*. It then turns the LED off or on accordingly.

Lines 23, 25, and 27 toggle the LED. Flashing the LED is achieved by toggling it, sleeping for a short time (200ms), and then toggling it again. Finally, the code writes the state of the LED to the serial interface with `serial.write`, so the computer can read its state. Save this file as `code.py`. The microcontroller is now waiting for commands.

Giving Commands

If you have been paying attention, you might have noticed that connecting your Pico to the computer now results in two serial devices appearing on your Linux system: `/dev/ttyACM0` and `/dev/ttyACM1`. The console port for the REPL usually appears as the first listed, whereas the data port that you enabled in `boot.py` is typically the second listed. You'll use this second port, `/dev/ttyACM1`, to communicate with the Pico from your computer.

To give these commands, simply open a terminal window and run `screen`:

```
screen /dev/ttyACM1
```

As soon as you press one of the keys for the characters 0, 1, 2, or 3, the Pico responds with the corresponding action and returns the state of the LED as a 0 (off) or 1 (on). You do not need to press Enter. If you type anything other than the four characters defined in the protocol, the Pico simply replies with the current state of the LED: 0 or 1. (See the "Expand the Possibilities" box.)

To close the serial connection in the screen session, just press Ctrl + A and then a backslash (\). Confirm that you want to terminate by typing *y*.

From Your Web Browser

Although the `screen` command is useful for testing, it's not the most user-friendly interface. In the remaining part of this article, I demonstrate how to build a web-based interface for communication with your microcontroller and the Web Serial API [7]. (See also the "Alternative Firmware and PC Software" box.) Note

Expand the Possibilities

For simplicity, I have limited the example code to recognize only four commands. However, you can easily define a more complex protocol. For example, you can add a second character to the commands to choose a specific LED number, allowing you to control multiple LEDs, or you can define additional commands to set the color of an LED, enabling control of multiple RGB color LEDs. Try expanding the code in this article to take advantage of the full potential of your microcontroller and computer.

Listing 2: code.py LED Control

```
01 from board import LED
02 from digitalio import DigitalInOut, Direction
03 from time import sleep
04 import usb_cdc
05
06 led = DigitalInOut(LED)
07 led.direction = Direction.OUTPUT
08
09 # Get the USB data feed object
10 serial = usb_cdc.data
11
12 while True:
13     # Check for incoming data
14     if serial.in_waiting > 0:
15         command = serial.read(1)
16
17         # Process command
18         if command == b"0":
19             led.value = False
20         elif command == b"1":
21             led.value = True
22         elif command == b"2":
23             led.value = not led.value
24         elif command == b"3":
25             led.value = not led.value
26             sleep(0.20)
27             led.value = not led.value
28
29         # Return state of LED
30         if led.value:
31             serial.write(b"1")
32         else:
33             serial.write(b"0")
```

Alternative Firmware and PC Software

In this article, I've demonstrated the microcontroller side with CircuitPython. However, you can achieve the same result with various other development platforms, such as Arduino code. For example, the small and cheap Digispark [11] microcontroller board can communicate with your computer and the Arduino library DigiCDC [12]. Although the original board is no longer manufactured, you can still find clones in various web shops. Numerous alternatives for Web Serial also exist for the software on the computer. For example, you can write a program in Python that communicates with the serial port and the *pySerial* library [13]. You can even create a graphical interface with a toolkit such as Tkinter [14]. As long as you ensure that both sides use the same protocol, the alternatives are interchangeable.

that this API is currently only supported in the Chrome or Edge web browsers.

A simple implementation is a website called *serialterminal.com* [8]. After visiting this website from a supported web browser, click *Connect* in the top left corner. Your web browser will then display a list of serial interfaces (Figure 4). Select the appropriate one (*Pico (ttyACM1) – Paired*) from the list and click *Connect*.

Make sure to uncheck the options *send with /r*, *send with /n*, and *echo*. You can then enter commands in the text field at the top. For example, type *1* and click *Send* on the right. The LED on your Pico should now turn on because the CircuitPython code running on the microcontroller recognizes the character 1 and reacts accordingly. Type *2* to toggle the LED, after which it will turn off. You will also see the output from the microcontroller in the large text field at the bottom: *1* if the LED is on and *0* if it is off.

Creating a Web Interface

Now that you know you are able to communicate with the board from a web page, you can create a more convenient interface in your own web applications to control the LED with the same Web Serial technology from the *serialterminal.com* website. First, create a simple HTML page with the code in Listing 3.

This web page has buttons to connect, turn off, turn on, toggle, and flash the LED. It also displays an icon of a light bulb (in the form of an emoji) to represent the state of the LED (Figure 5).

JavaScript Code

The HTML page includes the `usb-led.js` file as a script. This file contains the code in Listing 4 to send commands to the connected microcontroller by the Web Serial API.

The JavaScript code comprises functions to read the state of the LED from the microcontroller and to write commands to control the LED. The `readState` function reads a byte from the serial interface and updates the light bulb icon to be visible if the byte corresponds to the character *1* or to be hidden if it corresponds to the character *0*. The `writeCommand` function (lines 18-24) writes a command to the serial interface and then reads the LED state by calling the `readState` function.

The remaining code is executed after the Document Object Model (DOM) has fully loaded, adding event listeners to all the buttons. When you click *Connect*, the port is chosen by the Web Serial API. Clicking on the other buttons calls the `writeCommand` function with the corresponding command.

To use this web interface, open the HTML page in Chrome, make sure your

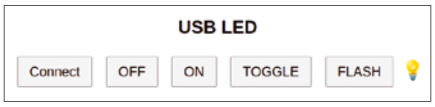
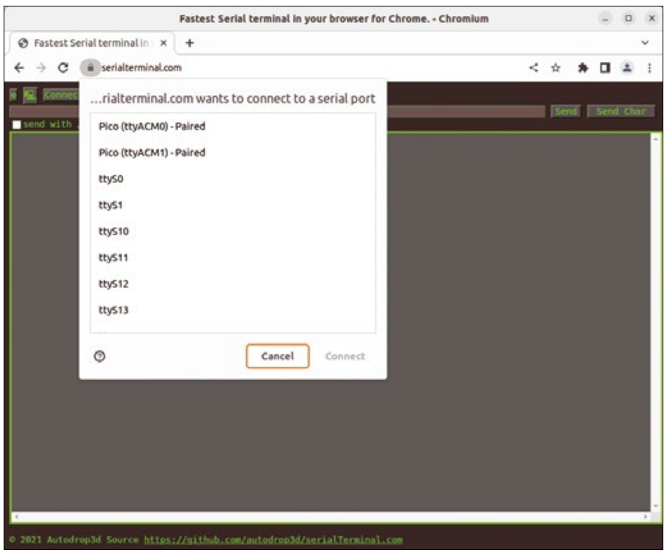


Figure 5: This web interface allows you to control the LED on the Raspberry Pi Pico.


Fastest Serial terminal in your browser for Chrome. - Chromium

serialterminal.com

...rialterminal.com wants to connect to a serial port

Pico (ttyACM0) - Paired
Pico (ttyACM1) - Paired
ttyS0
ttyS1
ttyS10
ttyS11
ttyS12
ttyS13

Cancel Connect

© 2021 Autodrop3d Source https://github.com/autodrop3d/serialTerminal.com

Figure 4: You can test your serial communication on the website *serialterminal.com*.

Listing 3: usb-led.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>USB LED</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            padding: 50px;
        }
        button {
            font-size: 24px;
            padding: 10px 20px;
            margin: 10px;
        }
        #lightbulbIcon {
            font-size: 32px;
        }
    </style>
    <script src="usb-led.js"></script>
</head>
<body>
    <h1>USB LED</h1>
    <button id="connectButton">Connect</button>
    <button id="offButton">OFF</button>
    <button id="onButton">ON</button>
    <button id="toggleButton">TOGGLE</button>
    <button id="flashButton">FLASH</button>
    <span id="lightbulbIcon">&#128161;</span>
</body>
</html>
```

Pico is connected, and click *Connect*. Choose the correct port (Figure 6) and try out the different buttons to control the LED on the Pico. The light bulb will disappear when you turn off the LED and reappear after turning it on.

Temperature Sensor

Similarly, you can connect a temperature sensor to the Raspberry Pi Pico and have it send measurements to your computer over USB. For this example, I chose the popular BME280 temperature sensor made by Bosch – specifically, Adafruit's breakout board with the sensor. Cheaper versions are available from Chinese manufacturers, but make sure it is an I2C version that operates at 3.3V.

Disconnect the USB cable from the Pico and place the board on a breadboard. Connect SDA (SDI on the Adafruit board) to pin 26 (GP20) of the Pico, SCL (SCK on Adafruit) to pin 27 (GP21), VCC (Vin on Adafruit) to 3.3V, and GND to GND (Figure 7). If you are unsure about the correct pins on the microcontroller board, refer to the Raspberry Pi Pico Pinout website [9]. After connecting all the wires, reconnect the Pico to USB.

Next, download Adafruit's CircuitPython Library Bundle [10] for CircuitPython 8.x, extract the ZIP file, go into the `lib` directory, and copy the `adafruit_bme280` and `adafruit_bus_device` directories to the `lib` directory of your

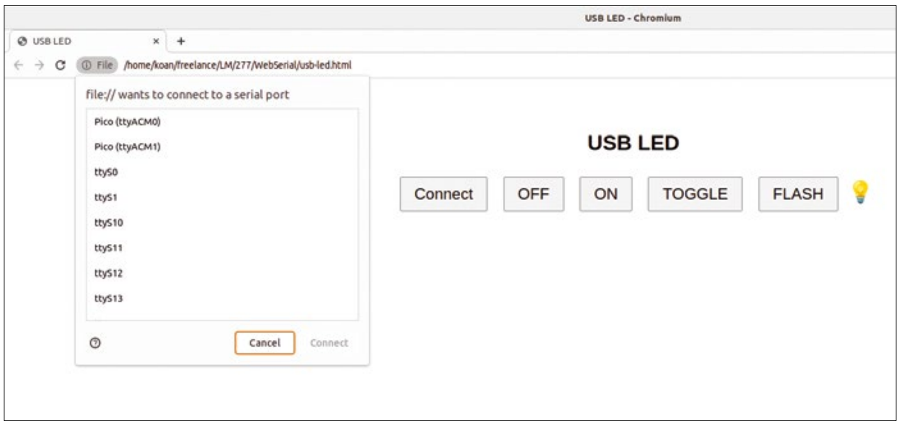


Figure 6: Grant the web page access to your microcontroller's USB port with Web Serial.

Listing 4: usb-led.js

```
let lightbulbIcon;
let port;
let writer;

async function readState() {
    const reader = port.readable.getReader();
    const { value, done } = await reader.read();

    const receivedByte = value[0];
    if (receivedByte === '1'.charCodeAt(0)) {
        lightbulbIcon.style.display = "inline";
    } else if (receivedByte === '0'.charCodeAt(0)) {
        lightbulbIcon.style.display = "none";
    }
    reader.releaseLock();
}

async function writeCommand(command) {
    if (!writer) return;

    const data = new TextEncoder().encode(command);
    await writer.write(data);
    await readState();
}

document.addEventListener("DOMContentLoaded", () => {
    const connectButton = document.getElementById("connectButton");
    const offButton = document.getElementById("offButton");
    const onButton = document.getElementById("onButton");
    const toggleButton = document.getElementById("toggleButton");
    const flashButton = document.getElementById("flashButton");
    lightbulbIcon = document.getElementById("lightbulbIcon");

    connectButton.addEventListener("click", async () => {
        if (!navigator.serial) {
            alert("Web Serial API not supported by this browser.");
            return;
        }

        try {
            port = await navigator.serial.requestPort();
            await port.open({ baudRate: 9600 });
            writer = port.writable.getWriter();

            connectButton.disabled = true;
        } catch (error) {
            console.error("Error:", error);
        }
    });

    offButton.addEventListener("click", async () => {
        await writeCommand("0");
    });

    onButton.addEventListener("click", async () => {
        await writeCommand("1");
    });

    toggleButton.addEventListener("click", async () => {
        await writeCommand("2");
    });

    flashButton.addEventListener("click", async () => {
        await writeCommand("3");
    });

});
```

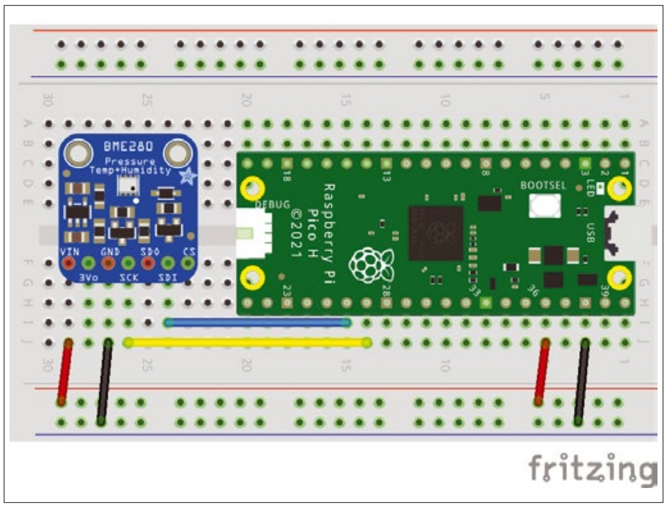


Figure 7: Connect the BME280 temperature sensor to the Raspberry Pi Pico.

Listing 5: code.py Temperature Sensor

```
from time import sleep
import board
import busio
from adafruit_bme280.basic import Adafruit_BME280_I2C
import usb_cdc

i2c = busio.I2C(scl=board.GP21, sda=board.GP20)
bme280 = Adafruit_BME280_I2C(i2c)
serial = usb_cdc.data

while True:
    temperature = round(bme280.temperature, 1)
    serial.write(str(temperature).encode("utf-8"))
    serial.write(b"\n")
    sleep(1)
```

CIRCUITPY drive to install the CircuitPython driver for the BME280.

Sending Temperature Measurements

Next, modify the code in `code.py` in the Mu editor to read the temperature and humidity continuously from the sensor and send them through the serial interface (Listing 5).

First, the code sets up the I2C bus, defining the appropriate GPIO pins for SCL and SDA. Next, it initializes the Adafruit BME280 driver. By default, the library uses the sensor with I2C address 0x77. If your BME280 uses address 0x76, specify this as an argument when initializing the driver:

```
bme280 = Adafruit_BME280_I2C(↩
        i2c, address=0x76)
```

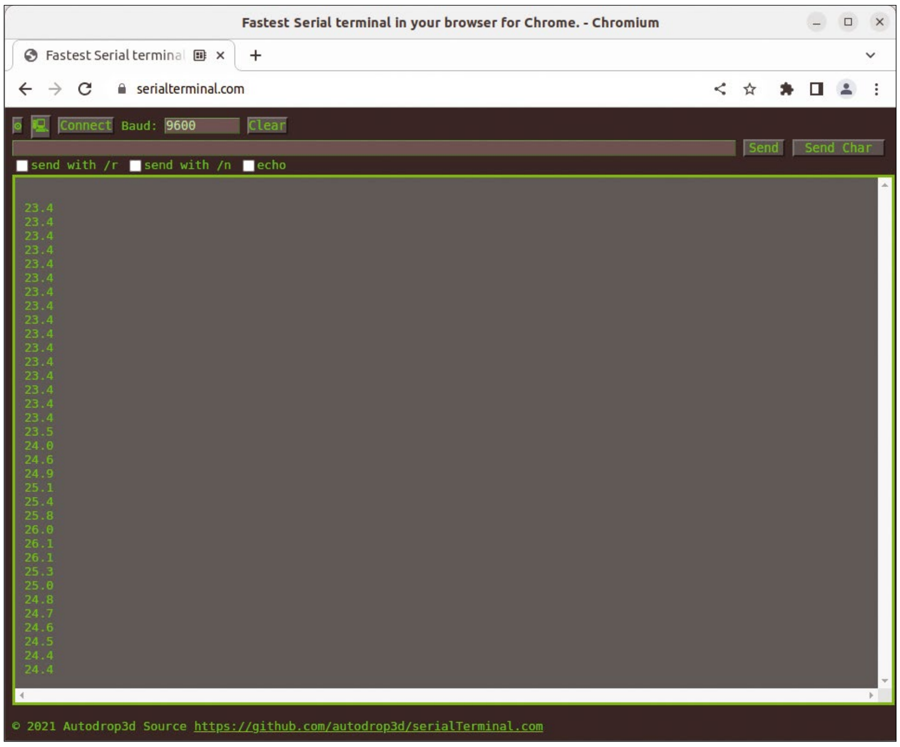
The next line obtains the data connection for the serial interface. Finally, the program starts an infinite loop that reads the temperature from the sensor, rounds it to one decimal place, writes it to the serial interface as a string encoded in UTF-8 followed by a newline character, and sleeps for one second.

Save this code as `code.py` and connect to the Pico's `/dev/ttyACM1` device with `screen` in the terminal or *serialterminal.com* in Chrome. You'll see the temperature measurements scrolling by (Figure 8), each measurement on a new line because of the `"\n"` newline character.

Sensor Measurements in HTML

To display the sensor measurements in a web interface, you can create a simple HTML page, like that shown in Listing 6. This page only contains a button to establish the serial connection and a `span` element to display the temperature. The corresponding JavaScript file shown in Listing 7 is also straightforward.

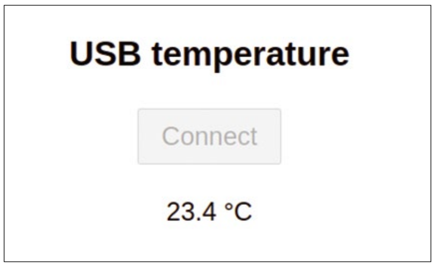
After the DOM has loaded, the JavaScript code adds an event listener to the *Connect* button. Once you have clicked on the button and the serial connection is established, the code continuously reads a text stream. Each time a line of text is received, it updates the `span`



```
23.4
23.4
23.4
23.4
23.4
23.4
23.4
23.4
23.4
23.4
23.4
23.4
23.4
23.4
23.4
23.4
23.5
24.0
24.6
24.9
25.1
25.4
25.8
26.0
26.1
26.1
25.3
25.0
24.8
24.7
24.6
24.5
24.4
24.4
```

© 2021 Autodrop3d Source https://github.com/autodrop3d/serialTerminal.com

Figure 8: See the temperature sensor measurements in your serial terminal.



USB temperature

Connect

23.4 °C

Figure 9: The web page continuously updates with the temperature from the sensor.

Listing 6: usb-bme280.html

```
<!DOCTYPE html>
<html lang="en">
<head>
    <title>USB Temperature</title>
    <style>
        body {
            font-family: Arial, sans-serif;
            text-align: center;
            padding: 50px;
        }
        button {
            font-size: 24px;
            padding: 10px 20px;
            margin: 10px;
        }
        .measurement {
            font-size: 24px;
            padding: 10px 20px;
            margin: 10px;
        }
    </style>
    <script src="usb-bme280.js"></script>
</head>
<body>
    <h1>USB Temperature</h1>
    <button id="connectButton">Connect</button>
    <p class="measurement"><span
       id="temperature">TODO</span> °C</p>
</body>
</html>
```

Listing 7: usb-bme280.js

```
document.addEventListener("DOMContentLoaded", () => {
    const connectButton = document.getElementById("connectButton");
    const temperatureSpan = document.getElementById("temperature");

    connectButton.addEventListener("click", async () => {
        if (!navigator.serial) {
            alert("Web Serial API not supported by this browser.");
            return;
        }

        try {
            port = await navigator.serial.requestPort();
            await port.open({ baudRate: 9600 });
            connectButton.disabled = true;

            const textDecoder = new TextDecoderStream();
            const readableStreamClosed = port.readable.pipeTo(textDecoder.
                                             writable);
            const reader = textDecoder.readable.getReader();

            // Listen to data coming from the serial device.
            while (true) {
                const { value, done } = await reader.read();
                if (done) {
                    // Allow the serial port to be closed later.
                    reader.releaseLock();
                    break;
                }
                // Remove carriage return and newlines from string
                temperature = value.replace(/[\r\n]+/g, "")
                if (temperature.length > 0) {
                    temperatureSpan.textContent = temperature;
                }
            }
        } catch (error) {
            console.error("Error:", error);
        }
    });
});
```

element with the ID `temperature`, updating the temperature continuously on the web page (Figure 9).

Improvements

This project has many areas for improvement. For example, you can convert the temperature sensor measurements from degrees Celsius to degrees Fahrenheit, read the humidity from the sensor and display it in another text box on the web page, and improve the look of the layout with a stylesheet. ■■■

Info

[1] CircuitPython: *https://circuitpython.org*

[2] USB CDC support: *https://docs.circuitpython.org/en/latest/shared-bindings/usb_cdc/*

[3] CircuitPython downloads: *https://circuitpython.org/downloads*

[4] CircuitPython firmware for Raspberry Pi Pico: *https://circuitpython.org/board/raspberry_pi_pico/*

[5] Mu: *https://codewith.mu*

[6] Installation instructions for Mu: *https://codewith.mu/en/howto/1.2/install_linux*

[7] Web Serial API: *https://wicg.github.io/serial/*

[8] serialterminal.com (Chrome or Edge browsers only): *https://www.serialterminal.com*

[9] Raspberry Pi Pico pinout: *https://pico.pinout.xyz*

[10] CircuitPython Library Bundle: *https://circuitpython.org/libraries*

[11] Digispark: *http://digistump.com/products/1*

[12] DigiCDC: *http://digistump.com/wiki/digispark/tutorials/digicdc*

[13] pySerial: *https://pyserial.readthedocs.io*

[14] Tkinter: *https://docs.python.org/3/library/tkinter.html*

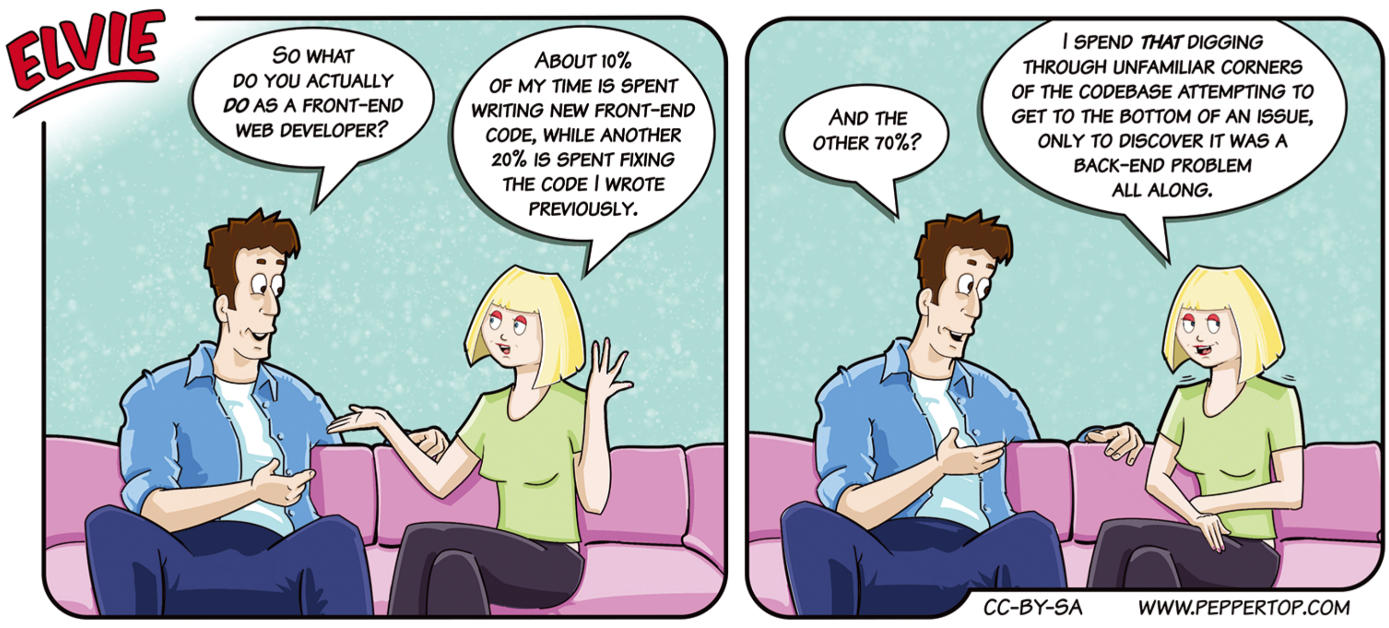
Author

Koen Vervloesem has been writing about Linux and open source, computer security, privacy, programming, artificial intelligence, and the Internet of Things for more than 20 years. You can find more on his website at *koen.vervloesem.eu*.

Everybody knows Blender is one of the coolest open source applications on the planet, but it is also one of the most intimidating. The Blender 3D graphics toolkit is a powerful tool for modeling, rendering, sculpting, animation, and story art, but how do you dip your foot in it without getting thrown in the deep end? This month we show you a modeling project that could be a good way to get started: a partially automated 3D model of a clock. And speaking of graphics, we also take a look at Tube Archivist, a handy app for organizing videos. To round out this edition of Linux Voice: Get started on backing up your data with Duplicati.

Image © Olexandr Moroz, 123RF.com

LINUXVOICE


ELVIE
SO WHAT DO YOU ACTUALLY DO AS A FRONT-END WEB DEVELOPER?
ABOUT 10% OF MY TIME IS SPENT WRITING NEW FRONT-END CODE, WHILE ANOTHER 20% IS SPENT FIXING THE CODE I WROTE PREVIOUSLY.
AND THE OTHER 70%?
I SPEND THAT DIGGING THROUGH UNFAMILIAR CORNERS OF THE CODEBASE ATTEMPTING TO GET TO THE BOTTOM OF AN ISSUE, ONLY TO DISCOVER IT WAS A BACK-END PROBLEM ALL ALONG.
CC-BY-SA
WWW.PEPPERTOP.COM

Looking for your place in open source?

Set up job alerts and get started today!

OpenSource JOB HUB

opensourcejobhub.com/jobs

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

From no licenses to too many, software copyright finally made its way to including today's free software designation. BY JON "MADDOG" HALL

Software Copyright Evolution

Last month I touched briefly on copyright and this month I would like to continue the theme of intellectual property by discussing licenses.

Initially software was not able to have a copyright.

If the sources were exposed and there was no contract, then, basically, the software was in the public domain and people could do whatever they wanted with it.

Copyright applied to software changed all that. In most jurisdictions, the copyright was generated automatically and the sources and binaries of the code were not authorized for copying or even use.

One or more licenses had to be applied to the software (either in source code or binary form) that told the user about what rights they had with that software.

Working for Digital Equipment Corporation (DEC) at that time, my colleagues and I were required to write specific licenses for our software products. We had to learn the specific legal terms for the software we obtained from our sources and realized that in many cases we were a "sub-licensee." The originators of the software still had certain legal rights to it that we had to protect. Some of these rights we passed on to our customers, and some we retained.

For example, in most cases, we were not authorized to pass on the source code to our customers because the supplier still maintained the source code rights to the binaries we supplied. A prime example of this was Adobe's Display PostScript engine that went into our X Window System server that rendered Display PostScript on our workstations. There were only two software engineers at DEC who could see that source code, one for Digital UNIX and one for OpenVMS.

When DEC's customers wanted to get the source code for Ultrix (as an example), which was based on 4.1c BSD Unix, the customer first had to buy a Unix source code license from AT&T, then buy a source code license from DEC, then buy a source code distribution from DEC, and they *still* could not compile and build the entire system. "Pieces" were missing, not because DEC did not want to pass them on, but because DEC could not pass them on.

Another interesting part of licensing was the licenses that universities like University of California, Berkeley; MIT; Carnegie Mellon; and others had in their source code. These universities typically wanted to give their software away to other universities to collaborate with research. However, they did not want to spend thousands of dollars in legal fees for each copy of the software when they were not going to sell it. So they hit on the plan to embed a license into the source code that not only told the users what their rights were, but also worked to indemnify the university, because the software was not warranted for any specific use.

Different universities had different desires, so there were various licenses with certain minor differences.

Over time there were other entities that wanted to get credit for the software, have changes to the software returned to them so they could fix them, or get feedback from the use of the software.

Add in commercial companies with their changes, and soon there was a plethora of licenses out there, with only minor (and usually useless) differences.

Eventually it was recognized that the licenses were too cumbersome, particularly when taking all of these licenses and trying to combine them onto one CD/DVD/ISO. The distribution's lawyers would have to read every license and see if they were compatible.

So a group of people got together and sorted through the licenses, separating out all the similar licenses and putting them into two main groups: "permissive" and "restrictive."

The permissive licenses typically had the least requirements for the developer using the sources to pass on their source changes, either to the next developer or the end user. The restrictive licenses, of which GPL is the most famous, required the user of the sources to pass their changes on to the next developer or the end user.

The collection of all of these licenses were called "open source." However, it is worthwhile to explain the real differences between permissive and restrictive licenses.

Companies and developers love permissive open source. They get to use the sources that others have developed, which allows them to make products faster and with tested code, yet they are not forced to pass on *their* intellectual property to their competitors or to their customers. Their customers have to wait for the developers to produce the changes the customer needs or the bug fixes.

Customers love restrictive open source, better known as free software. With free software the customer can support their own software. They can find other programmers who have the expertise to fix it and not be dependent on the developers that made it. Software freedom. ■■■

Let Tube Archivist organize your YouTube collection

And ... Action!

Tube Archivist indexes videos or entire channels from YouTube in order to download them with the help of the yt-dlp tool.

BY FERDINAND THOMMES

YouTube videos have become an important part of many users' leisure and educational activities in recent years. Many videos now also end up on free video platforms such as PeerTube [1], LBRY [2], Invidious [3], or on NewPipe [4] for Android, but by no means do all of them. This is why, no matter what users think of Google, most will continue to rely on Google's video platform.

To archive videos and preempt deletions by content creators or Google, many YouTube fans download the videos to hardware they control. Downloading videos is prohibited by default in the terms of service for the free version of YouTube, however, content creators can grant permission through a Creative Commons license. Also, downloads for offline viewing are allowed for subscribers to the YouTube Premium edition.

Video collections have a habit of growing uncontrollably over time, to the point where things just run wild. That's why it makes sense to counter the threat of chaos by organizing everything right from the outset. This is where Tube Archivist (TA) comes in [5], a self-hosted media server for YouTube videos.

Functionality

TA lets you subscribe to YouTube channels, download videos, and index your archive so the collection can be browsed and streamed offline from any device on the local network. You can download both manually and automatically. Yt-dlp [6] runs in the background for this purpose. Searches against the collection are handled by the Elasticsearch [7] tool. For Firefox and Chrome there is the TA Companion [8], an extension that displays a download button when you open a YouTube video. In the case of a YouTube channel, a subscribe button also appears in TA.

TA gives you full control over the content you want to store. You can rescan the subscribed channels at any time and decide if you want to archive the new content. This is also a good way to manage the kind of YouTube content your kids are allowed to watch. The stored videos can be played with TA's built-in player, but also – given a little extra effort – output to Plex, Emby, or other media platforms.

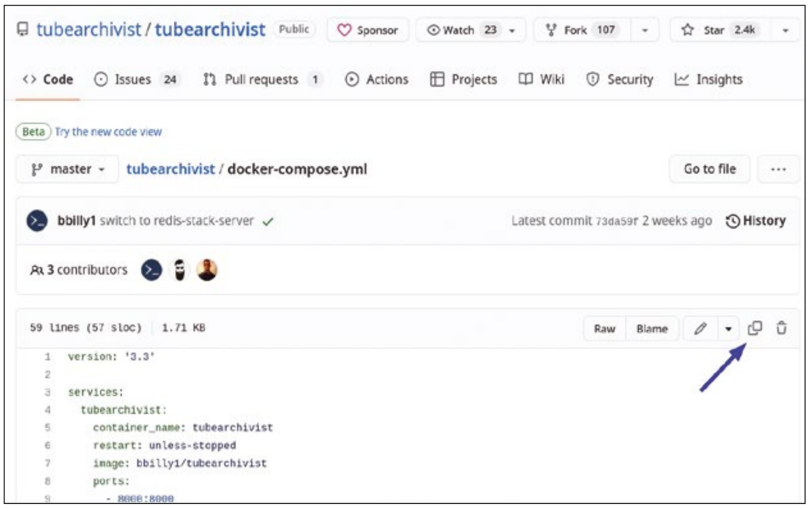
Before I go into more detail about the operation and other functions, I'll briefly look at the installation. The hardware requirements, according to the developers, are a 4-core CPU and 2GB RAM for smaller collections, or 4GB for medium to large collections, but more RAM never hurts, as we all know. The size of the required hard disk is primarily driven by the size of the archive you intend to keep. You can install TA as a Docker container; Docker Compose largely automates the process. The "Installing Docker and Docker Compose" box explains how to set up the current versions of the two components.

For testing, I installed TA on Proxmox. However, any hardware that meets the requirements can be used just as easily. Hardkernel single board computers (SBCs) such as ODROID-H3 or ODROID-H3+ [10] are ideal as a hardware basis for TA and can also host other services. Instructions for installing TA with Podman (via Docker Compose), Unraid, TrueNAS Scale, or on a Synology NAS can be found on GitHub [11]. If using an ARM64-based device, TA is available as a multi-arch container, and the same applies to Redis. For Elasticsearch, simply use the official image with ARM64 support. The software does not support other architectures.

Installing Docker and Docker Compose

You should always install Docker and Docker Compose directly and not through the distribution archives [9]. If you are familiar with Docker Compose: Previously, the command line was `docker-compose`, but newer versions use `docker compose`. The hyphenated variant just provokes an error message.

Figure 1: Copy the contents of `docker-compose.yml` from the GitHub page and save them to a file of the same name in your home directory.


tubearchivist / tubearchivist Public
Sponsor
Watch 23
Fork 107
Star 2.4k
Code
Issues 24
Pull requests 1
Actions
Projects
Wiki
Security
Insights
Beta Try the new code view
master
tubearchivist / docker-compose.yml
Go to file
bbilly1 switch to redis-stack-server
Latest commit 73da59f 2 weeks ago
History
3 contributors
59 lines (57 sloc) 1.71 KB
Raw
Blame

```
version: '3.3'

services:
  tubearchivist:
    container_name: tubearchivist
    restart: unless-stopped
    image: bbilly1/tubearchivist
    ports:
      - 8000:8000
```

Figure 2: TA comes with a clear-cut design. As you can see, I have already subscribed to some channels; some of the videos are partially viewed.

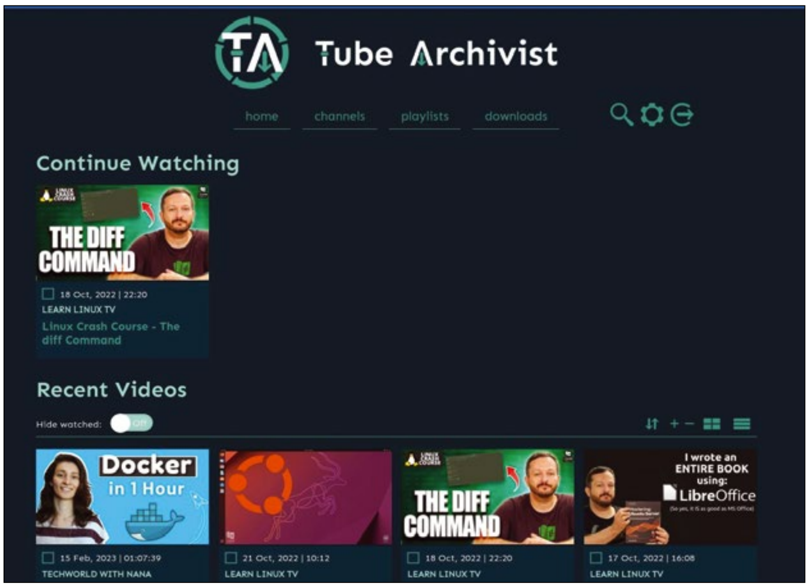
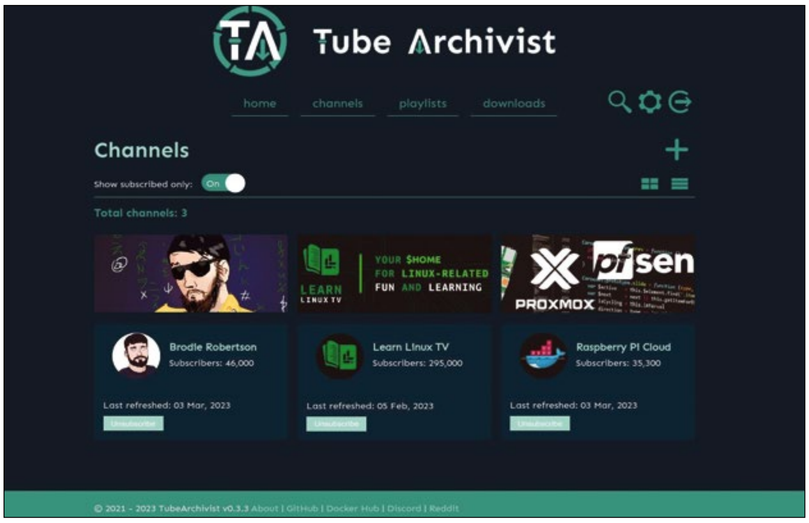


Figure 3: The *Channels* section contains the subscribed channels in a list or grid view. You can use the *Unsubscribe* button to delete subscriptions you no longer need.



Preliminary Work

After installing Docker and Docker Compose, copy the contents of the YAML file for Docker Compose (Figure 1) to a file in your home directory and name it `docker-compose.yml` [12].

Then edit the file and enter the IP address or domain name of your TA system for `TA_Host`. `TA_USERNAME` and `TA_PASSWORD` are where you store the matching access credentials. Also replace the default password in `Elastic_Password` and set the time zone correctly.

Use the `docker ps` command to check that ports 8000 for TA and 9200 for Elasticsearch are available. If not, change the first part of the `8000:8000` port specification to a different port in the YAML file. Changing the port for Elasticsearch is a bit more complex; it is described in the documentation in the wiki.

`HOST_UID` and `GID_UID` are equivalent to the ID of `1000` specified in the YAML file on Ubuntu. For other distributions, check this with the `id` command and edit the details of the YAML file if needed. `volumes:` is where you specify where TA stores the videos and where the cache is located.

After saving the file, set up the containers with the `docker compose up -d` command. Then launch TA by typing *http://<TA-IP>:8000* or *http://<TA-FQDN>:8000* in a browser and logging in with the credentials created previously. If you want to access TA from outside your home network, you will also need to set up a reverse proxy.

Getting Started

After the launch, you are taken to TA's user interface in Dark Mode; you can swap this for a brighter desktop in the settings. The interface is clearly laid out and visually appealing (Figure 2). In the upper part you will see four sections *Home*, *Channels*, *Playlists*, and *Downloads* along with three icons for searching, settings, and logging out. The *Home* [13] section gives you an overview of the videos you already downloaded, which can be played back there by the integrated player. TA shows separately videos that have not been viewed completely to let you continue viewing. The software can also hide viewed videos if so desired.

The *Channels* [14] section lists YouTube channels you subscribe to (Figure 3). To subscribe to a channel, enter either the 25-digit YouTube channel ID or the channel's URL in a field that you open by pressing the plus button. Alternatively, enter the URL of a video from the channel and let TA find the channel's URL. Subscribing to a channel does not download any videos.

If you click on one of the subscribed channels, the list of videos already downloaded from that channel will appear. You can use the

Unsubscribe button to terminate a subscription to a channel. *About* gives you more information and configuration options for the respective channel (Figure 4). These settings override the entries in the global settings dialog. For example, you can specify a video format that differs from the global settings and specify whether and when TA automatically deletes videos you have played back. In addition, you can decide whether you want to see the sponsor block for this channel.

Downloading

After indexing the videos of a channel, you can either ignore or download individual clips from it. The *Start download* button starts downloading the indexed movies of all channels, except the ignored titles. How many videos TA retrieves in a session is determined in the settings. The *Playlists* section [15] lets you subscribe to playlists, in a similar way to subscribing to channels.

The *Downloads* [16] view lets you rescan your subscriptions, start downloading the movies in the queue, or add videos, channels, or playlists to it (Figure 5). Below that, you can use the drop-down box to filter for the subscriptions you want TA to display in the list below. After triggering a queue download, two new elements appear under the button; you can use them to stop the download immediately or after the current video. All four categories support either a list or grid format.

Searching

The powerful search function provided via the automatically integrated Elasticsearch container makes it easy to find videos, channels, and playlists and even offers a full-text search against the indexed subtitles.

By default, fuzzy searching is enabled for all search queries; this also catches typos in search queries. You can narrow down the search using primary keywords and control the fuzziness using secondary keywords. For further details, please refer to the detailed documentation [17]. For example, for a full-text search of the subtitles for the term *supported open source*, type

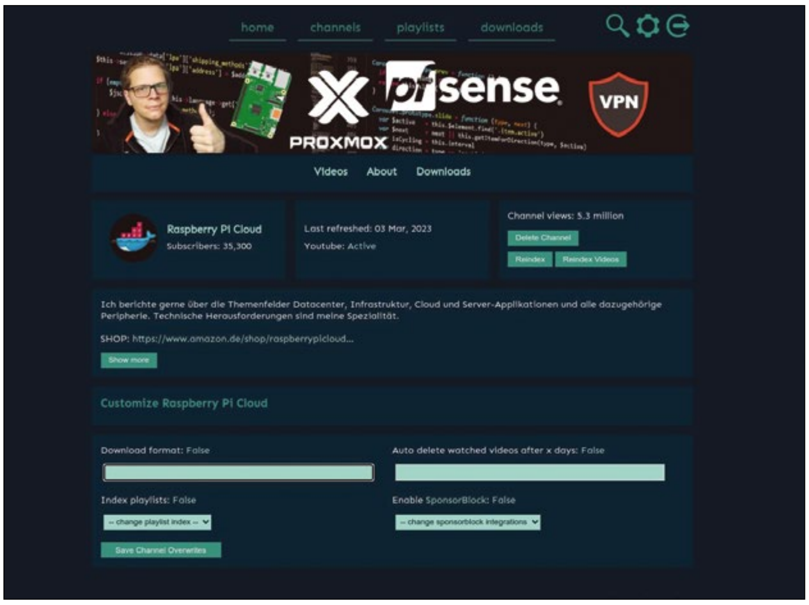
```
full:supported open source lang:en.
```

TA's global settings (Figure 6) turn out to be so extensive that an explanation would go beyond the scope of this article. Again, please refer to the detailed documentation [18].

TA Companion

The TA Companion browser extension for Firefox and Chrome, which I mentioned earlier on, lets you download from and add subscriptions on the YouTube website. After installing the add-on, call it and enter your TA URL, including port number, and, below it, the matching API key (Figure 7). You will find this in the TA settings.

TA Companion now adds a download button to the YouTube pages of individual videos and an additional *Subscribe* button to pages for channels (Figure 8). After that you don't need TA Companion's icon in the browser buttonbar any more and can hide it.



Raspberry Pi Cloud
Subscribers: 35,300
Last refreshed: 03 Mar, 2023
Youtube: Active
Channel views: 5.3 million

Ich berichte gerne über die Themenfelder Datacenter, Infrastruktur, Cloud und Server-Applikationen und alle dazugehörige Peripherie. Technische Herausforderungen sind meine Spezialität.

SHOP: https://www.amazon.de/shop/raspberrypicloud...

Customize Raspberry Pi Cloud

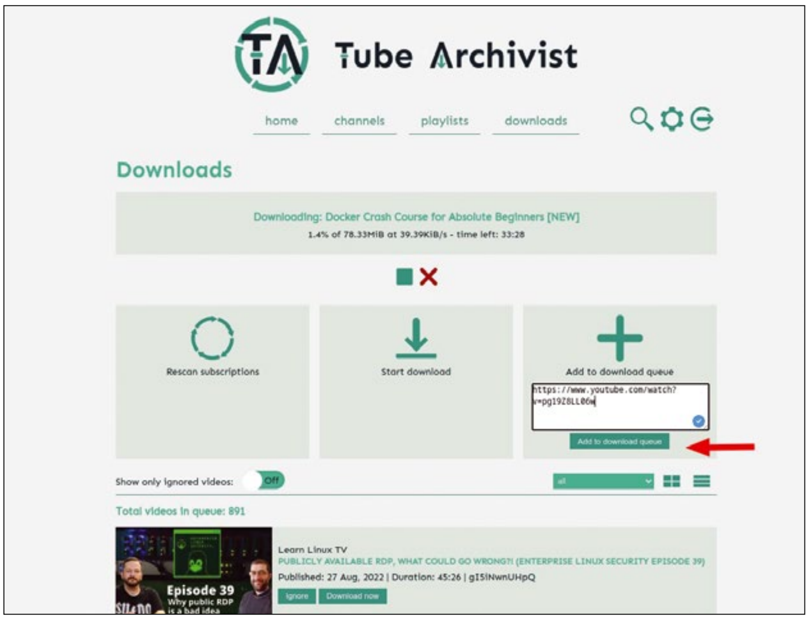
Download format: False
Auto delete watched videos after x days: False
Index playlists: False
-- change playlist index --
Enable SponsorBlock: False
-- change sponsorblock integrations

Figure 4: After clicking on the label of a channel you will see the *Show more* button, among other things. It provides more detailed information and supports individual configuration of the channel.

Conclusions

Tube Archivist is by far the best YouTube media server I know of that not only indexes content, but also plays it. Integration with media players such as Plex is possible, although anything but trivial [19]. During our testing, TA ran without errors, regardless of whether we controlled it

Figure 5: In the *Downloads* section, press the plus button to add individual videos to the queue. *Start download* lets you trigger the queue download; to the left, you can update your subscriptions.



Downloads

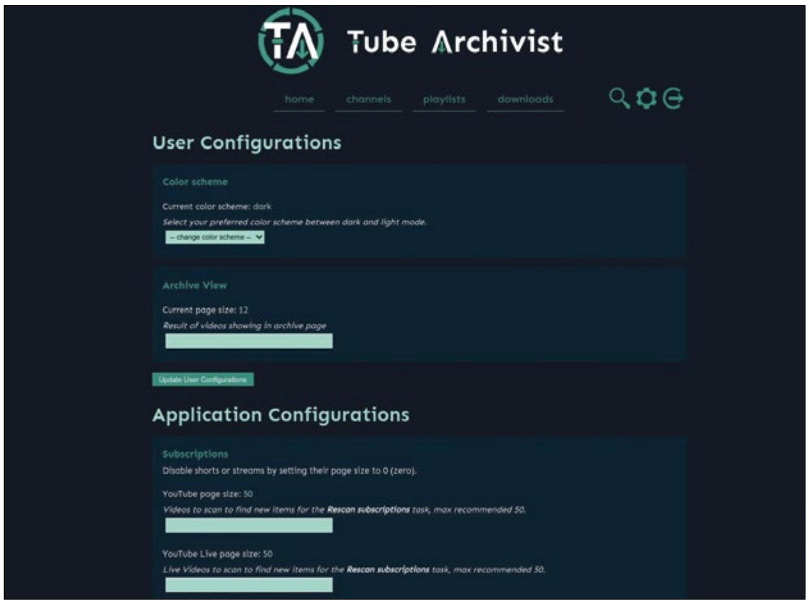
Downloading: Docker Crash Course for Absolute Beginners [NEW]
1.4% of 78.33MiB at 39.39KiB/s - time left: 33:28

https://www.youtube.com/watch?v=pg19Z8LL06w

Show only ignored videos: Off

Total videos in queue: 891

Learn Linux TV
PUBLICLY AVAILABLE RDP, WHAT COULD GO WRONG?! (ENTERPRISE LINUX SECURITY EPISODE 39)
Published: 27 Aug, 2022 | Duration: 45:26 | gI5lNwnUHpQ


Tube Archivist

home channels playlists downloads

User Configurations

Color scheme

Current color scheme: dark

Select your preferred color scheme between dark and light mode.

-- change color scheme --

Archive View

Current page size: 12

Result of videos showing in archive page

Update User Configurations

Application Configurations

Subscriptions

Disable shorts or streams by setting their page size to 0 (zero).

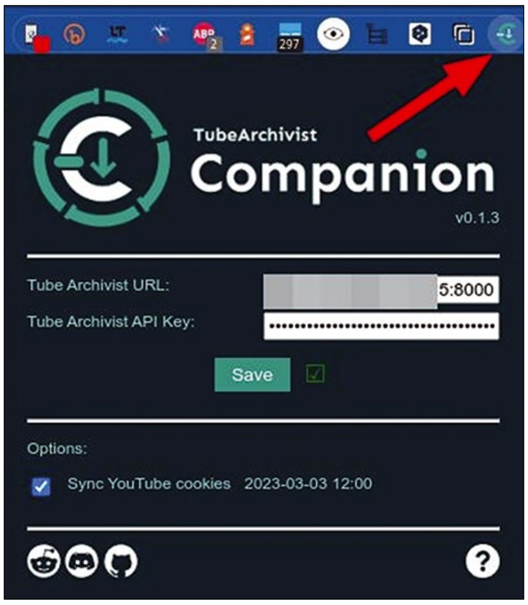
YouTube page size: 50

*Videos to scan to find new items for the **Rescan subscriptions** task, max recommended 50.*

YouTube Live page size: 50

*Live Videos to scan to find new items for the **Rescan subscriptions** task, max recommended 50.*

Figure 6: Despite their considerable scope, TA's global settings are pleasingly clear and described in detail in the documentation.


TubeArchivist Companion

v0.1.3

Tube Archivist URL: 5:8000

Tube Archivist API Key: ••••••••••••••••••••••••••••••••••••

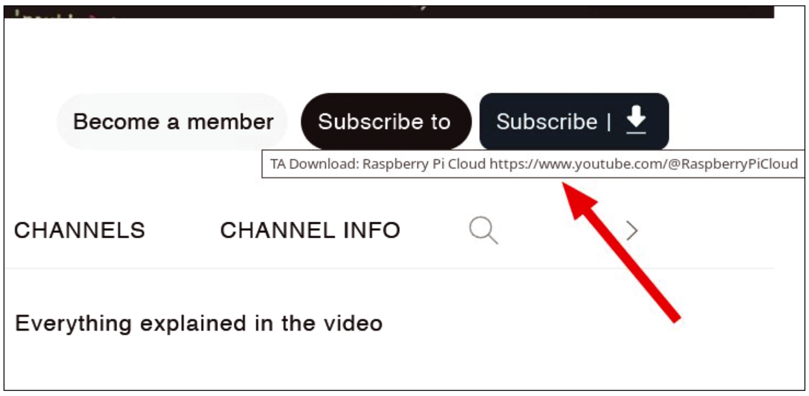
Save

Options:

☑ Sync YouTube cookies 2023-03-03 12:00

Figure 7: To activate TA Companion, you need TA's IP address or the domain name including the port and the API key found in the settings.

Figure 8: After setting up TA Companion in the browser, a new *Subscribe to* button appears on the left for channels on YouTube. The button to the right is for individual videos.


Become a member
Subscribe to
Subscribe |
TA Download: Raspberry Pi Cloud https://www.youtube.com/@RaspberryPiCloud
CHANNELS
CHANNEL INFO
Everything explained in the video

directly in TA or used the TA Companion browser plugin directly on the YouTube page.

The application is under continuous development. Among other things, extended user management, podcast integration, and PyFilesystem2 integration for more flexible storage of the videos are on the roadmap. Notifications and improved statistics will also be added. It makes you curious about how this interesting project will develop. ∎∎∎

The Author

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

Info

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

[2] LBRY: *https://en.wikipedia.org/wiki/LBRY*

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

[4] NewPipe: *https://newpipe.net*

[5] Tube Archivist: *https://www.tubearchivist.com/*

[6] yt-dlp: *https://github.com/yt-dlp/yt-dlp*

[7] Elasticsearch: *https://www.elastic.co/elasticsearch/*

[8] TA Companion: *https://github.com/tubearchivist/browser-extension*

[9] Setting up Docker Compose: *https://docs.docker.com/compose/gettingstarted/*

[10] Odroid H3+: *https://ameridroid.com/products/odroid-h3?_pos=4&_sid=c3c42ef82&_ss=r*

[11] Install TA: *https://github.com/tubearchivist/tubearchivist/blob/master/docs/Installation.md*

[12] YAML: *https://github.com/tubearchivist/tubearchivist/blob/master/docker-compose.yml*

[13] TA project page: *https://github.com/tubearchivist/tubearchivist/wiki#getting-started*

[14] TA Channels: *https://github.com/tubearchivist/tubearchivist/wiki/Channels*

[15] TA Playlists *https://github.com/tubearchivist/tubearchivist/wiki/Playlists*

[16] TA Downloads: *https://github.com/tubearchivist/tubearchivist/wiki/Downloads*

[17] TA Search: *https://github.com/tubearchivist/tubearchivist/wiki/Search*

[18] TA settings: *https://github.com/tubearchivist/tubearchivist/wiki/Settings*

[19] TA integration in Plex: *https://www.reddit.com/r/PleX/comments/pxkx37/comment/heozfcf/?context=3*

Build a clock with Blender and Python

Tick Tock

With a little help from Blender you can create your own 3D models – including animations. This article shows you how to assemble a partially automated virtual watch model with Blender and Python. BY RALF KIRSCHNER

The free Blender program package for modeling, texturing, animation, and video and image editing can be found in the package repositories of most Linux distributions, and there is also a distribution-independent Snap package. Typing the `snap install blender` command at the command line installs the graphics suite. If you need more in the way of installation options, you can download the application directly from the Blender Foundation website [1]. While you are there, you can also access the extensive documentation, tutorials, and examples and grab the versions for Windows, macOS, and others. On top of this, because the Blender Foundation provides the source code, the program can even be adapted to run on less common operating systems. Of course, in this case you will have to compile Blender yourself.

On Linux, either open Blender in a terminal or use the *Open in Terminal* shortcut in the window manager of your choice. If neither succeeds, first launch Blender and then try to discover the path of the Blender installation in the Python Interactive Console by typing `bpy.app.binary_path`. Then enter this as the start parameter for the call in the terminal. This ensures that error messages and output from the Python command `print("Hello world")`, for example, also reach their target (i.e., the terminal window). This is especially important if you don't just want to use the Blender Python Console in Blender's Scripting workspace and individual commands, but also want to call Python programs you saved previously.

Desktop

Blender's user interface is divided into workspaces. Each of them hosts a different collection of editors and windows that appear at specific positions on the screen. The program lists the available workspaces on the right below the menu bar. They include Layout, Modeling, Sculpting, UV Editing, and Animation. Almost all workspaces contain the 3D viewport window and other windows.

Figure 1 shows Blender after starting with the default basic scene. Basically, you can assemble 3D scenes to your heart's content directly in the interface. There are numerous tutorials on the Internet for getting started with Blender; a description of this would go beyond the scope of this article due to the range of functions to cover. Instead, the task at hand is to enable you to write tools for yourself in the form of Python scripts that let you build complex scenes one by one in a scripted way.

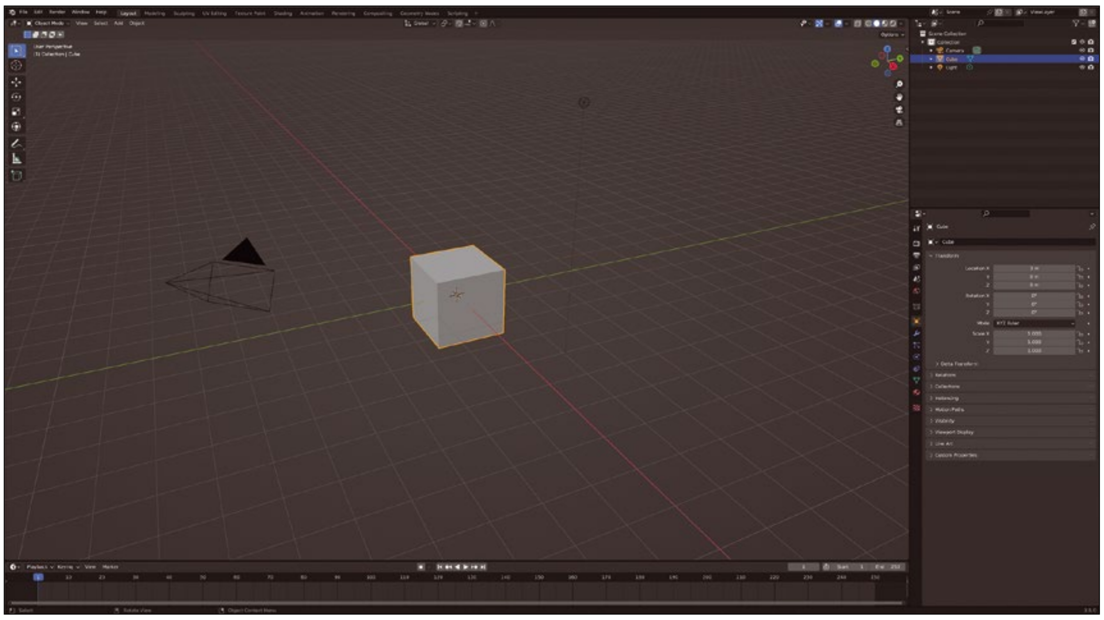


Figure 1: Blender with the default basic scene in a 3D view: A perspective view on a cube, a camera, and a light source.

Clock Face

As an example, let's look at a three-dimensional clock face with raised numerals and hands. What's interesting in this context is the positions of the numerals on the dial.

The only way to position them correctly is to use some trigonometric functions you may still recall from math.

But let's start by partially assembling the dial by hand, and at the same time you'll get to know some Blender Python. To do this, switch to the Scripting workspace. This will take you to a programming interface (Figure 2). On the left, you can see the familiar, somewhat reduced 3D viewport window. Below that are the Blender Python Console and the Info line. Clicking on *Add* | *Mesh* | *Plane* creates a square base for the dial. Parallel to this, Blender logs your manual actions in the Info window bottom left. The window content is very similar to the first line of Listing 1.

One thing up front: Blender forwards keyboard input directly to whichever window you mouse over. This means that you do not need to click on the window beforehand. But, especially in the scripting workspace, you need to make sure that the mouse pointer is in the right position when entering keystrokes. Start by moving the mouse over to the 3D viewport window, press *X*, and confirm the *Delete* prompt. The square base area should disappear again and another line will be logged in the Info window.

Then build the square base area again, but this time with a different size and using command input in the Blender Python Console (Listing 1, line 2). In the 3D viewport window, the program now shows you a larger version of the square base area. Some positive feedback appears in the Blender Python Console (line 3). The Info line below contains the complete log entry including all default values that are not specified.

If you want to familiarize yourself with how variables and loops work as a small exercise before actually building the dial, why not first create a small pyramid? To do this, enter the code from Listing 2 in the Blender Python Console. Note that loops in Python are always formed by indenting the loop block. In the Blender Python Console, end the indentation using a blank line at the end of the loop.

Key Concepts

You can use the Python API to access the data in Blender in the same way as via the user interface. Basically, what you can access by pressing switches and buttons and selecting menu items can also be controlled using Python. All the data of the currently loaded Blender file can be accessed using the `bpy.data` module.

The application organizes the data in Collections, which you access either by index or string. In the Outliner, there are precisely three objects after starting the program with the standard

Listing 1: Blender Action

```
01 bpy.ops.mesh.primitive_plane_add(size=2, enter_editmode=False,
      align='WORLD', location=(0, 0, 0), scale=(1, 1, 1))
02 bpy.ops.mesh.primitive_plane_add(size=5)
03 {'FINISHED'}
```

Listing 2: Pyramid Builder

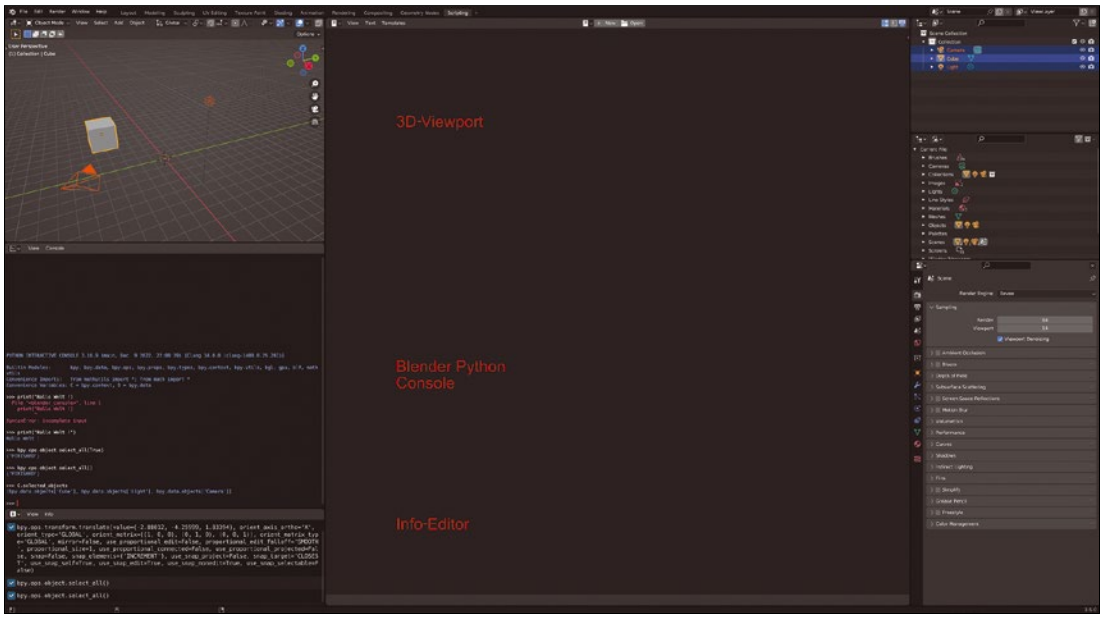
```
i=1
while i < 9:
  bpy.ops.mesh.primitive_cube_add(size=1, enter_editmode=False,
     align='WORLD', location=(0, 0, i-0.5), scale=(9-i, 9-i, 1))
  i+=1
```

Listing 3: Preparations and Cleanup

```
import bpy
import math
import datetime

bpy.ops.object.select_pattern(pattern='text*')
bpy.ops.object.select_pattern(pattern='*hand')
bpy.ops.object.select_pattern(pattern='plane')
bpy.ops.object.delete()
hour = datetime.datetime.now().hour
minute = datetime.datetime.now().minute
hourangle = 360 / 12 * (hour + minute/60)
minuteangle = 360 / 60 * minute
fromCenter = 3
angleInc = 30 * math.pi/180

bpy.ops.mesh.primitive_plane_add(size=9, enter_editmode=False,
   align='WORLD', location=(0, 0, 0), scale=(1, 1, 1))
```



```
>>> bpy.ops.object.select_all(True)
{'FINISHED'}

>>> bpy.ops.object.select_all()
{'FINISHED'}

>>> C.selected_objects
[bpy.data.objects['Cube'], bpy.data.objects['Light'], bpy.data.objects['Camera']]
```

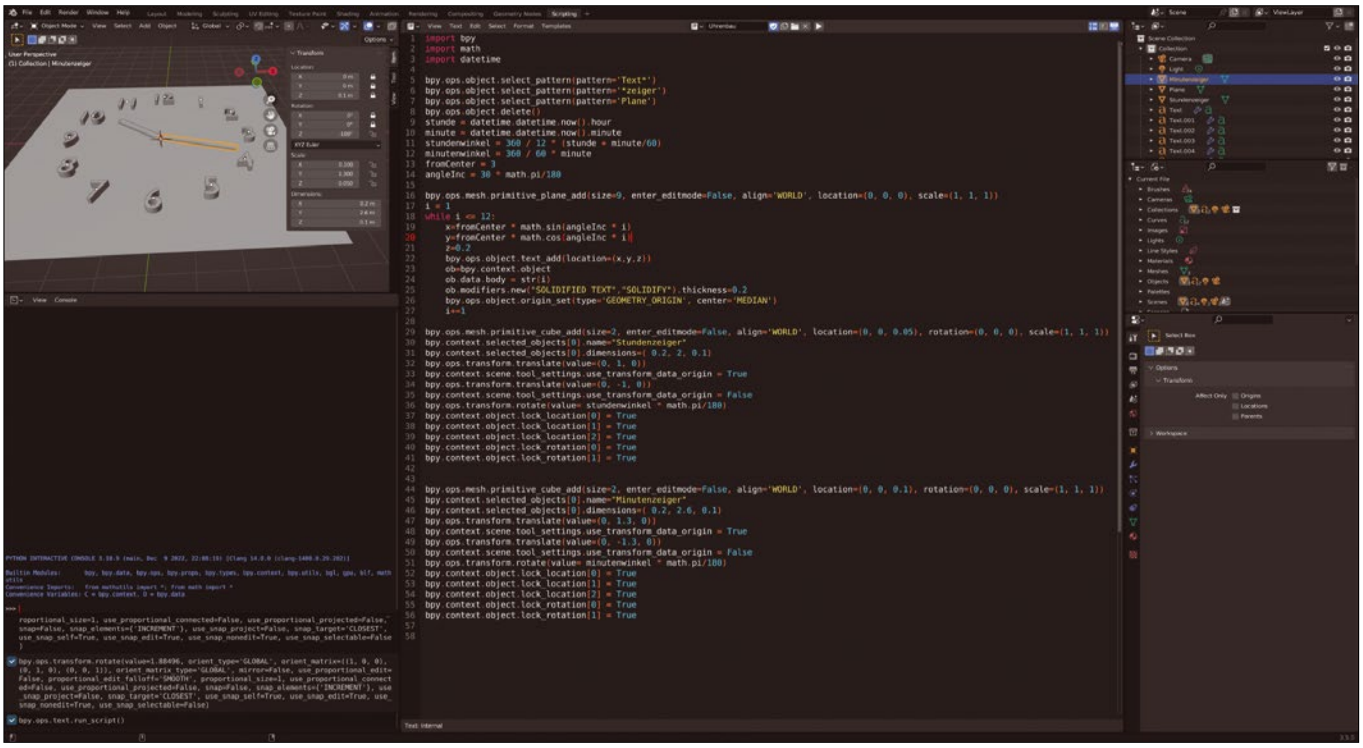
```
bpy.ops.transform.translate(value=(-2.88812, -4.25559, 1.83394), orient_axis_ortho='X', orient_type='GLOBAL', orient_matrix=((1, 0, 0), (0, 1, 0), (0, 0, 1)), orient_matrix_type='GLOBAL', mirror=False, use_proportional_edit=False, proportional_edit_falloff='SMOOTH', proportional_size=1, use_proportional_connected=False, use_proportional_projected=False, snap=False, snap_elements={'INCREMENT'}, use_snap_project=False, snap_target='CLOSEST', use_snap_self=True, use_snap_edit=True, use_snap_nonedit=True, use_snap_selectable=False)
bpy.ops.object.select_all()
bpy.ops.object.select_all()
```

Figure 2: With the Scripting workspace enabled, the application displays the Blender Python Console and an info line next to the 3D viewport window.

scene: `bpy.data.objects['Camera']`, `bpy.data.objects['Cube']`, and `bpy.data.objects['Light']`. Objects can also be retrieved via their index numbers (e.g., the cube is `bpy.data.objects[1]`).

As you are probably familiar with from the dialog-oriented interface, in many cases you do not select the object by typing its name, but use the mouse instead. You can even pick any number of objects at the same time, for example, to move them together. However, only one of them acts as the active object to which all actions refer. In addition to this, all the displayed values come from the active object. Operations that you want to act on multiple objects use this object as a reference, for example, if you assign a different material.

Within Python, you can access the active object using the `bpy.context.object` command and access all the selected objects using `bpy.context.`



```python
import bpy
import math
import datetime

bpy.ops.object.select_pattern(pattern='Text*')
bpy.ops.object.select_pattern(pattern='*zeiger')
bpy.ops.object.select_pattern(pattern='Plane')
bpy.ops.object.delete()
stunde = datetime.datetime.now().hour
minute = datetime.datetime.now().minute
stundenwinkel = 360 / 12 * (stunde + minute/60)
minutenwinkel = 360 / 60 * minute
fromCenter = 3
angleInc = 30 * math.pi/180

bpy.ops.mesh.primitive_plane_add(size=9, enter_editmode=False, align='WORLD', location=(0, 0, 0), scale=(1, 1, 1))
i = 1
while i <= 12:
    x=fromCenter * math.sin(angleInc * i)
    y=fromCenter * math.cos(angleInc * i)
    z=0.2
    bpy.ops.object.text_add(location=(x,y,z))
    ob=bpy.context.object
    ob.data.body = str(i)
    ob.modifiers.new("SOLIDIFIED TEXT","SOLIDIFY").thickness=0.2
    bpy.ops.object.origin_set(type='GEOMETRY_ORIGIN', center='MEDIAN')
    i+=1

bpy.ops.mesh.primitive_cube_add(size=2, enter_editmode=False, align='WORLD', location=(0, 0, 0.05), rotation=(0, 0, 0), scale=(1, 1, 1))
bpy.context.selected_objects[0].name="Stundenzeiger"
bpy.context.selected_objects[0].dimensions=( 0.2, 2, 0.1)
bpy.ops.transform.translate(value=(0, 1, 0))
bpy.context.scene.tool_settings.use_transform_data_origin = True
bpy.ops.transform.translate(value=(0, -1, 0))
bpy.context.scene.tool_settings.use_transform_data_origin = False
bpy.ops.transform.rotate(value= stundenwinkel * math.pi/180)
bpy.context.object.lock_location[0] = True
bpy.context.object.lock_location[1] = True
bpy.context.object.lock_location[2] = True
bpy.context.object.lock_rotation[0] = True
bpy.context.object.lock_rotation[1] = True


bpy.ops.mesh.primitive_cube_add(size=2, enter_editmode=False, align='WORLD', location=(0, 0, 0.1), rotation=(0, 0, 0), scale=(1, 1, 1))
bpy.context.selected_objects[0].name="Minutenzeiger"
bpy.context.selected_objects[0].dimensions=( 0.2, 2.6, 0.1)
bpy.ops.transform.translate(value=(0, 1.3, 0))
bpy.context.scene.tool_settings.use_transform_data_origin = True
bpy.ops.transform.translate(value=(0, -1.3, 0))
bpy.context.scene.tool_settings.use_transform_data_origin = False
bpy.ops.transform.rotate(value= minutenwinkel * math.pi/180)
bpy.context.object.lock_location[0] = True
bpy.context.object.lock_location[1] = True
bpy.context.object.lock_location[2] = True
bpy.context.object.lock_rotation[0] = True
bpy.context.object.lock_rotation[1] = True
```

```
PYTHON INTERACTIVE CONSOLE 3.10.9 (main, Dec  9 2022, 22:08:19) [Clang 14.0.0 (clang-1400.0.29.202)]

Builtin Modules:       bpy, bpy.data, bpy.ops, bpy.props, bpy.types, bpy.context, bpy.utils, bgl, gpu, blf, math
utils
Convenience Imports:   from mathutils import *; from math import *
Convenience Variables: C = bpy.context, D = bpy.data

>>> 
  roportional_size=1, use_proportional_connected=False, use_proportional_projected=False,
  snap=False, snap_elements={'INCREMENT'}, use_snap_project=False, snap_target='CLOSEST',
  use_snap_self=True, use_snap_edit=True, use_snap_nonedit=True, use_snap_selectable=False
  )

bpy.ops.transform.rotate(value=1.88496, orient_type='GLOBAL', orient_matrix=((1, 0, 0),
(0, 1, 0), (0, 0, 1)), orient_matrix_type='GLOBAL', mirror=False, use_proportional_edit=
False, proportional_edit_falloff='SMOOTH', proportional_size=1, use_proportional_connect
ed=False, use_proportional_projected=False, snap=False, snap_elements={'INCREMENT'}, use
_snap_project=False, snap_target='CLOSEST', use_snap_self=True, use_snap_edit=True, use_
snap_nonedit=True, use_snap_selectable=False)

bpy.ops.text.run_script()
```

Figure 3: Blender with the fully constructed clock in the activated Scripting workspace.

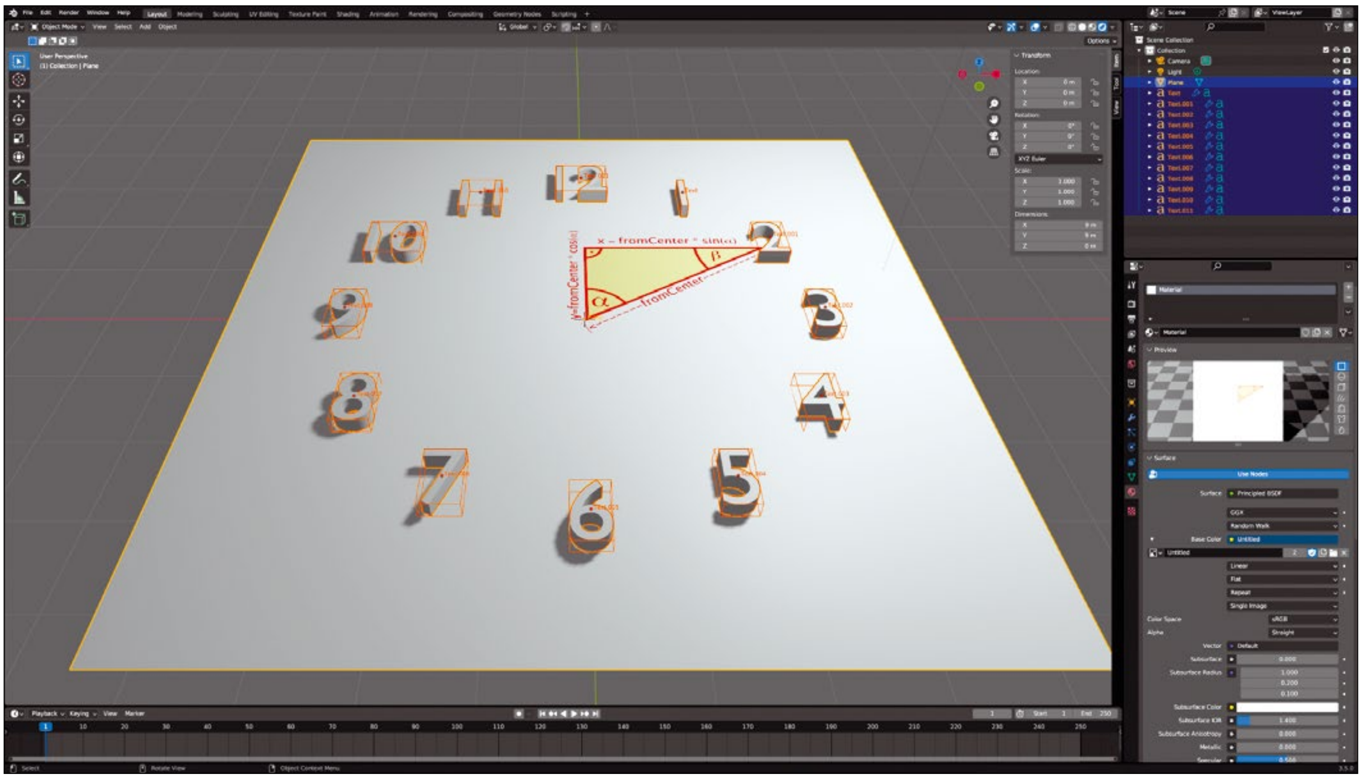


Figure 4: The Blender clock with the formulas as image texture in the Layout workspace.

`selected_objects`. However, access to this Context is read-only. To change the values, you will need to call API functions or use `bpy.data`.

Listing 3 shows the part of the Python script that positions the digits on the dial (Figure 3). Because the three-dimensional numerals have different widths and heights, you need to pay attention to the correct setting of the origin points of the numerals on the clock face. This ensures correct centering of the one- and two-digit numbers. Figure 4 illustrates the relationship of the trigonometric functions from Listing 4. Listing 5 shows the entire script (available online at [2]).

Conclusions

Blender gives you free professional 3D and animation software that is even used in shorts on DVD, in YouTube clips and animation projects, by game developers, and more. In line with the many potential applications, the learning curve is relatively steep when you start working with Blender. As a reward for persevering, the program imposes virtually no limits to your own abilities when you start designing. If you invest sufficient time and effort, the results can be really impressive, as the lead image of this article demonstrates. ■■■

Listing 4: Positioning the Digits

```
i = 1
while i <= 12:
  x=fromCenter * math.sin(angleInc * i)
  y=fromCenter * math.cos(angleInc * i)
  z=0.2
  bpy.ops.object.text_add(location=(x,y,z))
  ob=bpy.context.object
  ob.data.body = str(i)
  ob.modifiers.new("SOLIDIFIED TEXT","SOLIDIFY").thickness=0.2
  bpy.ops.object.origin_set(type='GEOMETRY_ORIGIN', center='MEDIAN')
  i+=1
```

Info

[1] Blender Foundation: *https://www.blender.org/*

[2] Download: *https://linuxnewmedia.thegood.cloud/s/5Rzx9tQW2FJ6N3Z*

The Author

Ralf Kirschner worked as a Visual Basic programmer in a software and systems house. He is a qualified system administrator and offers computer training as a freelancer.

Listing 5: Finished Python Script

```
import bpy
import math
import datetime

bpy.ops.object.select_pattern(pattern='Text*')
bpy.ops.object.select_pattern(pattern='*Pointer')
bpy.ops.object.select_pattern(pattern='plane')
bpy.ops.object.delete()
hour = datetime.datetime.now().hour
minute = datetime.datetime.now().minute
hourangle = 360 / 12 * (hour + minute/60)
minuteangle = 360 / 60 * minute
fromCenter = 3
angleInc = 30 * math.pi/180

bpy.ops.mesh.primitive_plane_add(size=9, enter_editmode=False,
  align='WORLD', location=(0, 0, 0), scale=(1, 1, 1))
i = 1
while i <= 12:
  x=fromCenter * math.sin(angleInc * i)
  y=fromCenter * math.cos(angleInc * i)
  z=0.2
  bpy.ops.object.text_add(location=(x,y,z))
  ob=bpy.context.object
  ob.data.body = str(i)
  ob.modifiers.new("SOLIDIFIED TEXT","SOLIDIFY").thickness=0.2
  bpy.ops.object.origin_set(type='GEOMETRY_ORIGIN',
    center='MEDIAN')
  i+=1

bpy.ops.mesh.primitive_cube_add(size=2, enter_editmode=False,
  align='WORLD', location=(0, 0, 0.05), rotation=(0, 0, 0),
  scale=(1, 1, 1))
bpy.context.selected_objects[0].name="hour hand"
bpy.context.selected_objects[0].dimensions=( 0.2, 2, 0.1)
bpy.ops.transform.translate(value=(0, 1, 0))
bpy.context.scene.tool_settings.use_transform_data_origin = True
bpy.ops.transform.translate(value=(0, -1, 0))
bpy.context.scene.tool_settings.use_transform_data_origin = False
bpy.ops.transform.rotate(value= angleHours * math.pi/180)
bpy.context.object.lock_location[0] = True
bpy.context.object.lock_location[1] = True
bpy.context.object.lock_location[2] = True
bpy.context.object.lock_rotation[0] = True
bpy.context.object.lock_rotation[1] = True

bpy.ops.mesh.primitive_cube_add(size=2, enter_editmode=False,
  align='WORLD', location=(0, 0, 0.1), rotation=(0, 0, 0),
  scale=(1, 1, 1))
bpy.context.selected_objects[0].name="minute hand"
bpy.context.selected_objects[0].dimensions=( 0.2, 2.6, 0.1)
bpy.ops.transform.translate(value=(0, 1.3, 0))
bpy.context.scene.tool_settings.use_transform_data_origin = True
bpy.ops.transform.translate(value=(0, -1.3, 0))
bpy.context.scene.tool_settings.use_transform_data_origin = False
bpy.ops.transform.rotate(value= angleMinutes * math.pi/180)
bpy.context.object.lock_location[0] = True
bpy.context.object.lock_location[1] = True
bpy.context.object.lock_location[2] = True
bpy.context.object.lock_rotation[0] = True
bpy.context.object.lock_rotation[1] = True
```

FOSSPicks

Sparkling gems and new releases from the world of Free and Open Source Software

This month, Graham realized just how important command-line applications have become after it became increasingly difficult to find a new FOSSPick that hasn't been developed for the terminal. BY GRAHAM MORRISON

Vector image editor

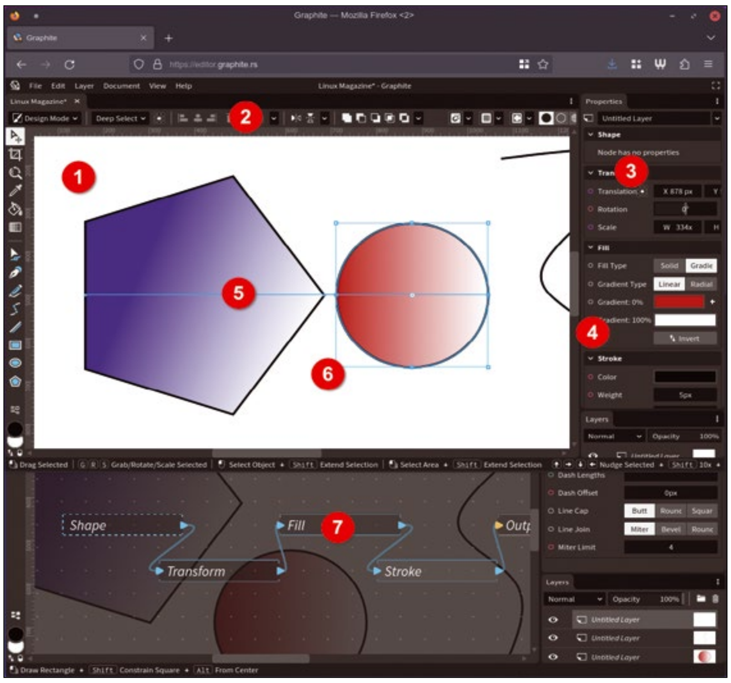
Graphite

Open source art and design tools have transformed what is possible for anyone needing digital illustration. It wasn't so long ago that costly applications such as Photoshop or Illustrator were the only ways to produce professional output, but that has changed. Tools such as Inkscape and Krita can do just as well, and even the humble Gimp is great for multilayer bitmap editing – no subscriptions required. The new software idea in design and illustration tools is non-destructive node graph editing, and outside of Blender, open source options are currently few and far between. Graphite, however, is one such option, and while it's only early in the developmental phase, and currently only available within a web browser, it's an entirely open source vector graphics editor that promises a native Linux desktop client soon.

With non-destructive node graph editing, each distinctive tool or process you use adds a "node" to a graph, with each node linked to both the previous edit and the subsequent edit. A final session will consequently look like a row of various nodes all linked together. Graphite shows this graph beneath the editing canvas, and it's the only part of the UI that comes from other graphics applications. There's a large tool palette on the left, individual properties for tools on the right, and the canvas itself in the center. The palette includes tools for freehand drawing, a path generator, various shapes to draw, advanced typesetting, and flood-fill with lots of gradients. It's all beautifully designed and rendered. Every element is drawn perfectly at the native resolution of your display. Even in a web browser, it's fast to use. It may not be ideal, but it does also mean you can collaborate on an illustration, and it will even work on a smartphone or Android tablet. You can import images and use both raster and vector images as your sources and output, and HDR and WCG are supported for color rendering. The back-end render engine is "resolution-agnostic," which means your designs are kept as algorithms until they're finally exported at whatever size or resolution you need, as either a PNG, JPEG, or SVG file. Even raster images become infinitely scalable, with no pixelation, thanks to Graphite's own scaling algorithms.

The properties for the path generator allow for transformations, solid fills, and gradient fills, with different types of strokes for the lines. It's these properties, and the properties of the other tools, that you can revisit at any time by clicking on the node for the edit in the graph. And it's exactly this kind of nonlinear editing that makes Graphite so powerful. It means you can busy yourself with the important parts of a design or illustration, such as its layout and arrangement, without worrying about the exact details such as widths or colors because these can always be adjusted later. If you're doing work for a client, it means you can always go back to earlier steps in the design process to change properties without affecting later edits or values. This approach could also be potentially automated, and Graphite hopes to incorporate this parametric editing into studio production environments.

While using a web browser isn't ideal, you can at least self-host the project and run the browser locally if you prefer. A native application would be ideal, however, and this is promised. As the project has been written with Rust, with its own framework, this work should hopefully be more feasible than it might be for other, less modern projects. With a native application this might become one of the best open source solutions available.



1. Canvas: Due to the virtual nature of the canvas, it can be any size, including infinitely large. **2. Tools palette:** As with many such tools, you can draw with predefined shapes or freehand, with options for alignment and layer priority. **3. Properties:** Everything about a shape can be edited as a parameter, either in the *Properties* tab or the node editor. **4. Stroke and gradients:** Change how things are drawn and how they're filled. **5. Guides:** Alignment lines appear dynamically as you scale and move objects. **6. Rotate and scale:** Every element is algorithmically scaled and drawn. **7. Node editor:** Layers can be alternatively edited from nodes for every operation supported by the editor.

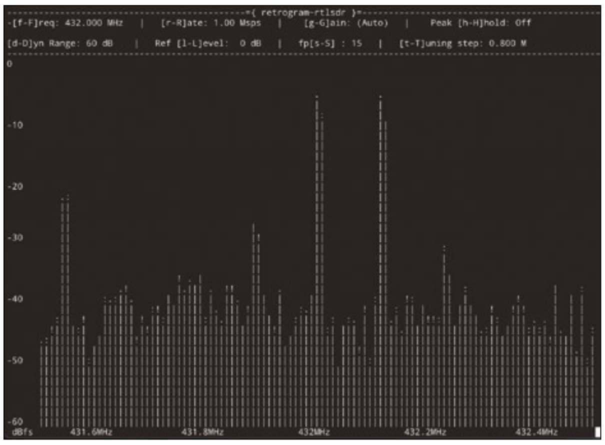
Project Website
https://graphite.rs/

Radio scanner

retrogram~rtlsdr

There's a lot you can now do with a humble USB digital television receiver, including much that has nothing to do with watching actual television. There are projects for capturing satellite transmissions, intercepting air traffic control messages, and for monitoring garage doors, automatic car door locks, and weather stations – and with retrogram, the wide-band radio spectrum of your surroundings. Retrogram~rtlsdr is a command-line utility that works with any Linux-compatible RTL-SDR (DVB-T) USB device, and there are many of them. Retrogram~rtlsdr transforms one of these cheap dongles into a histogram-bouncing, ASCII-art-generating radio scanner, and it works because so many of these devices are built generically for a global market. Through necessity, they're often capable of receiving data across a huge range of frequencies, especially when paired with a third-party aerial or antenna.

By default, retrogram~rtlsdr will tube itself to a frequency of 100,000,000Hz at a rate of 1,000,000 samples per second (device permitting). This will show a noisy histogram with highs and lows likely signifying nothing. But by pressing *F* or *f*, you can move up and down the frequency range, respectively, with configurable turning steps, gain control, sampling rate, frame rate, and dynamic range all controlled in real time with keys that are optionally shown onscreen. It can be fascinating scanning through the frequency ranges, trying to spot devices or common signals around you. But it can also be genuinely useful. The binary easily builds and runs on a Raspberry Pi, which can then be physically positioned or moved for maximum effect. Retrogram~rtlsdr can then scan the area, perhaps for interference problems, or to better see which devices on which frequencies might be interpreted with, for instance, RTL433. Without a scanner such as this, you would have to script your own solution that steps through a range of frequencies to see if they contain a signal that can be decoded. Retrogram~rtlsdr makes this process much easier to manage and looks very cool in the process.



```
---------------------------------------={ retrogram-rtlsdr }=---------------------------------------
-[f-F]req: 432.000 MHz   |   [r-R]ate: 1.00 Msps   |   [g-G]ain: (Auto)   |   Peak [h-H]hold: Off

[d-D]yn Range: 60 dB   |   Ref [l-L]evel:  0 dB   |   fp[s-S] : 15   |   [t-T]uning step: 0.800 M
-----------------------------------------------------------------------------------------------------
```

Turn your command line into a local wideband radio scanner with a cheap SDR USB device and retrogram~rtlsdr.

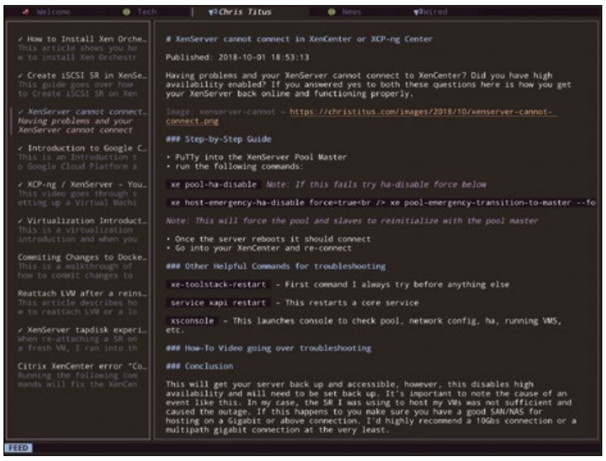
Project Website
https://github.com/r4d10n/retrogram-rtlsdr

Terminal RSS Reader

goread

RSS readers are brilliant because an RSS feed will only provide the skeleton or introduction of a story, or an update, so you can decide for yourself whether to invest more time in reading. Not only does this help to reduce the amount of time you waste browsing the Internet, it can be a conduit for useful information and updates outside your usual browsing habits. More importantly, RSS allows anyone to side step the increasingly cynical world of pop-ups, newsletters, Google's DRM, and any opt-out marketing that accompanies so many web pages. However, most modern RSS readers either exist within the web browser itself, as a service, or as a desktop application, which makes the web an all too tempting single click away.

Goread is a command-line RSS reader that will parse a simple configuration file containing links to RSS feeds and present these within its curses-based UI. This keeps the web a desktop away, and helps you to stay on top of your feed without resorting to a browser. The panel to left of the main view contains a categorized list of feeds pulled directly from the configuration file, while the main view shows the content list for any selected feed. As with other readers, feeds can be selected individually or through different layers of aggregation according to their category. Putting several feeds in a "News" category, for example, will either list every story combined from the RSS feeds in that category or the feeds for a specific source. Each source will open as a new tab, which can then be selected with the tab key! The configuration file can also be used to change the colors used to demarcate categories, feeds, and stories, and the command line feels like a natural place for all this. If you need to follow a link to the upstream story, you need to either copy and paste from your terminal or rely on a terminal configured to respond to links. It's that extra step that makes goread so useful in keeping you updated and distraction free.



```
✓ How to Install Xen Orche…
This article shows you ho
w to install Xen Orchestr

✓ Create iSCSI SR in XenSe…
This guide goes over how
to Create iSCSI SR on Xen

✓ XenServer cannot connect…
Having problems and your
XenServer cannot connect

✓ Introduction to Google C…
This is an Introduction t
o Google Cloud Platform a

✓ XCP-ng / XenServer - You…
This video goes through s
etting up a Virtual Machi

✓ Virtualization Introduct…
This is a virtualization
introduction and when you

Commiting Changes to Docke…
This is a walkthrough of
how to commit changes to

Reattach LVM after a reins…
This article describes ho
w to reattach LVM or a lo

✓ XenServer tapdisk experi…
When re-attaching a SR on
a fresh VM, I ran into th

Citrix XenCenter error "Co…
Running the following com
mands will fix the XenCen
```

```
# XenServer cannot connect in XenCenter or XCP-ng Center

Published: 2018-10-01 18:53:13

Having problems and your XenServer cannot connect to XenCenter? Did you have high
availability enabled? If you answered yes to both these questions here is how you get
your XenServer back online and functioning properly.

Image: xenserver-cannot - https://christitus.com/images/2018/10/xenserver-cannot-
connect.png

### Step-by-Step Guide

• PuTTy into the XenServer Pool Master
• run the following commands:

 xe pool-ha-disable  Note: If this fails try ha-disable force below

 xe host-emergency-ha-disable force=true<br /> xe pool-emergency-transition-to-master --fo

Note: This will force the pool and slaves to reinitialize with the pool master

• Once the server reboots it should connect
• Go into your XenCenter and re-connect

### Other Helpful Commands for troubleshooting

 xe-toolstack-restart  - First command I always try before anything else

 service xapi restart  - This restarts a core service

 xsconsole  - This launches console to check pool, network config, ha, running VMS,
etc.

### How-To Video going over troubleshooting

### Conclusion

This will get your server back up and accessible, however, this disables high
availability and will need to be set back up. It's important to note the cause of an
event like this. In my case, the SR I was using to host my VMs was not sufficient and
caused the outage. If this happens to you make sure you have a good SAN/NAS for
hosting on a Gigabit or above connection. I'd highly recommend a 10Gbs connection or a
multipath gigabit connection at the very least.
```

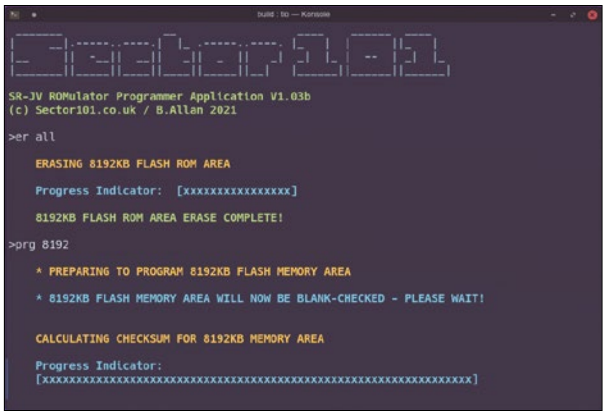
Access your favorite news stories without distractions from the command line with the wonderful goread RSS reader.

Project Website
https://github.com/TypicalAM/goread

Serial communications

tio

Serial ports used to be the standard way for computers to communicate with peripherals, whether they were printers, modems, or other computers. This was because a serial connection was easy to implement, both from a hardware and from a software perspective. There was one pin for sending data, another pin for receiving data, and a few more pins to help detect a connection and to manage the transmission. Software would send a burst of data and then wait to see if it was acknowledged. All a user had to do was make sure both ends of the connection were using the same transmission rate and error correction values. It was then a simple matter of sending, receiving, and interpreting the data. All of this could be done with a simple serial terminal application, and these used to be incredibly common. Even the Linux terminals we use every day are descended from serial communication devices, although with the exception of the venerable `screen`, many have lost their ability to talk to a serial port directly.
The lack of a modern serial terminal utility wouldn't be a problem if serial communication were no longer a thing, but there are still many devices that do connect to our computers via a simulated serial port over USB, including many Arduino and embedded devices. If you need to talk to these devices directly, you still need to use a serial terminal. As mentioned, `screen` still works, and Minicom is another good option, but neither are especially adaptive. You augment them with your own scripts, but they otherwise behave very much like a BBS terminal. This is where tio can help. Tio is a modern tool to help with simple serial input and output. Like Minicom, it can be used for interactive sessions, but it's far more adaptable and often works without the prerequisite trial and error. It defaults to a sensible configuration and can guess device names, show statistics, log to a file, pipe data capture or data transmission, redirect to a socket, and access a port at the same time as Minicom. If you ever find yourself in need of talking to something across a serial port, tio should be first on your list of resources.



```
SR-JV ROMulator Programmer Application V1.03b
(c) Sector101.co.uk / B.Allan 2021

>er all

    ERASING 8192KB FLASH ROM AREA

    Progress Indicator:  [xxxxxxxxxxxxxxxx]

    8192KB FLASH ROM AREA ERASE COMPLETE!

>prg 8192

    * PREPARING TO PROGRAM 8192KB FLASH MEMORY AREA

    * 8192KB FLASH MEMORY AREA WILL NOW BE BLANK-CHECKED - PLEASE WAIT!

    CALCULATING CHECKSUM FOR 8192KB MEMORY AREA

    Progress Indicator:
    [xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx]
```

Tio can talk to your serial devices and can be used to capture and send data in ways other purpose-built tools cannot.

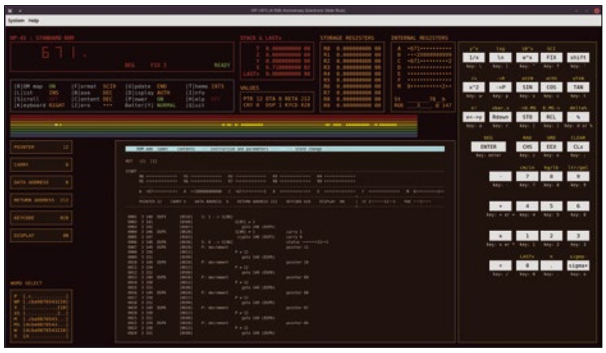
Project Website
https://github.com/tio/tio

Calculator emulator

HP-1973

We're used to thinking of emulation in terms of old games consoles or arcade machines, but there's also a thriving community of enthusiasts for old digital calculators. Their enthusiasm is there for many of the same reasons, too: a mixture of nostalgia, functionality, and modern capability that makes revisiting old hardware fulfilling in ways not possible earlier. One of the best examples of this is a software recreation of the HP-45 scientific calculator, called HP-1973. The HP-45 was launched in 1973 and was Hewlett Packard's second-ever scientific pocket calculator, and their successor to the first, the HP-35. Both had distinctively square and diminutive seven-segment displays and offered various trigonometric functions alongside basic calculator functionality, with the latter adding nine storage registers, a four-level stack, and a Shift key for accessing many more options from the same keys.

All of this has been recreated in HP-1973, a graphical desktop application that will run wherever Python 3 is installed. It recreates all the original features from the included ROM image and boasts both a realistic red simulation of the screen and image-driven button inputs, complete with shift functions. Most importantly, function of the original device has been expanded with introspection and monitoring tools to give you a peek at what's happening inside the calculator. The output display includes a map of the ROM data, the four-level stack, the internal register values, and a decompiled execution list. You can even interact with the decompiled execution list just as you might a debugger, with options to jump through the code and change values. It's a wonderful way to see what was happening within the black plastic case of the original as you fight with the input Reverse Polish notation, where operators follow their operands, to generate output.



Grab a copy of the original manual and spend a rainy afternoon learning the ins and outs of the world's second pocket scientific calculator.

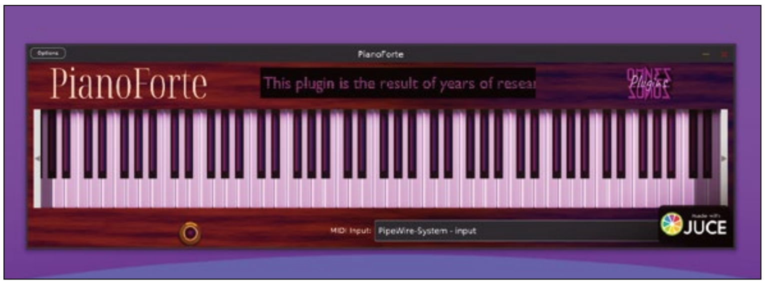
Project Website
https://sarahkmarr.com/retrohp1973.html

Piano plugin

Piano Forte

Despite a lack of professional musicians, Linux has become a wonderful platform for inventive music and sound processing, as well as professional-quality effects and plugins. One of the best of these is the commercial and proprietary Pianoteq. Pianoteq is a virtual piano that produces stunning results, not from audio recorded samples of a real piano, but from its own mathematical models of how a piano works. It calculates the audio energy and pressure from how a piano's hammers physically excite each string, how each string resonates with the others, and how everything resonates within the frame and its case. Thanks to the way everything interacts within the model, nothing quite compares to the sound Pianoteq can generate, which can be either exactly like a specific piano model, or tweaked and reprogrammed into a purely fictional instrument.

The Piano Forte Audio Plugin takes a very similar approach to generating piano-like sounds, and it has the distinct advantage of being open source. It's still at an early stage, but it can already generate a decent piano sound that is better than a simple SoundFont and can sound more natural to play. Like Pianoteq, the sound is entirely generated by algorithms, but unlike Pianoteq, the algorithms have been built with machine learning, using a neural network that's been trained to produce a piano-like sound. The size of the neural network is tiny compared to some of the numbers we're now used to hearing, just 8KB in size with 1,500 parameters, but it has at least been trained with freely available piano sample sets. Building an instrument like this is utterly unique. What's perhaps more interesting is that the same technique could be used to replicate all kinds of sounds, or even open up a new kind of synthesis, and it's wonderful that this technology is open source.


PianoForte
This plugin is the result of years of resear
MIDI Input: PipeWire-System - input
JUCE
Piano Forte sounds more like an electric piano than a real piano, but it's still a great achievement.

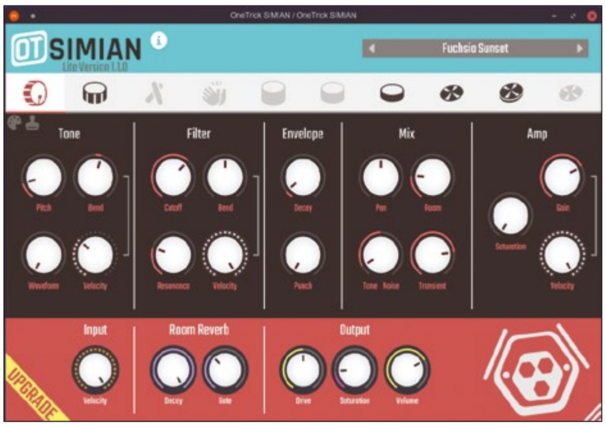
Project Website
https://github.com/tesserato/PianoForte

Drum synth

OneTrick SIMIAN

You can't work on most modern music without a drum beat, and it's something that computers have been able to help with for a long time. Even the humble Commodore 64 was great at drums. At a time when the cost of a drum machine was substantial, Commodore's bread-bin home computer could run a three-channel drum machine from a low-cost compact cassette called Micro-Rhythm. It may not have been a Simmons SDS-V, but it was pure, unrivaled 8-bit chunkiness. The Simmons SDS-V was also synonymous with the 1980s, as professional drummers replaced their acoustic kits with space-age hexagonal pads. It too was driven from its own, sound engine which was famous for producing space-age alien shooting tom fills. but rather more expensive, and it's this that OneTrick SIMIAN hopes to emulate.

OneTrick SIMIAN is a VST3 synth plugin that's been developed to recreate the original 10 sounds produced by the Simmons computer, including the kick drum, snare drum, claves, and claps, and especially the lo-fi samples for the cymbals. On top of these unique sound generators, which are all available consecutively and will need to be programmed from your MIDI device or application, there's a low-pass filter, an envelope, and an amplifier section with overdrive. The output can even be compressed through a 1176-style limiter, which was itself a costly device at the time, and the output can sound absolutely brilliant. All the settings are saved as a "kit" and 33 factory kits are included by default. The plugin is open source, but you'll need to either build it yourself, use the tagged "free" version, or pay a reasonable fee for the supported commercial version. This seems a great way to fund a niche project such as this, especially when there are so few drum synths that sound so good, and none on Linux. It's something every finger-tapping Linux user should try.

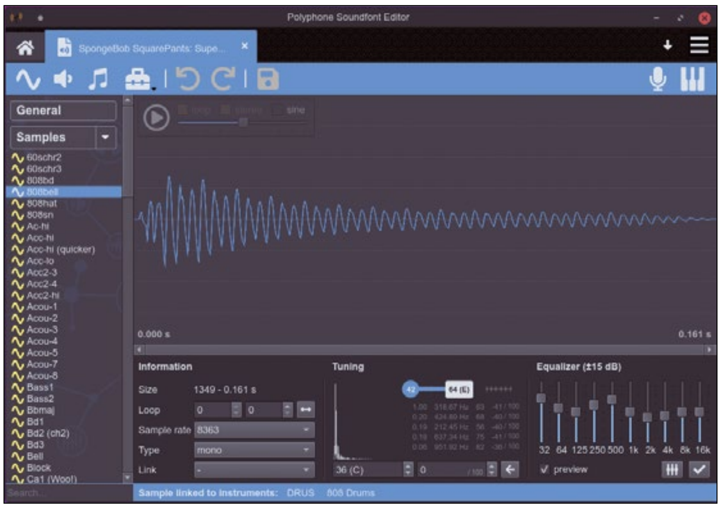


The drum synth even includes a gated reverb so you can recreate that drum break from Phil Collins' "In the Air Tonight."

Project Website
https://punklabs.com/ot-simian

Audio sample editor

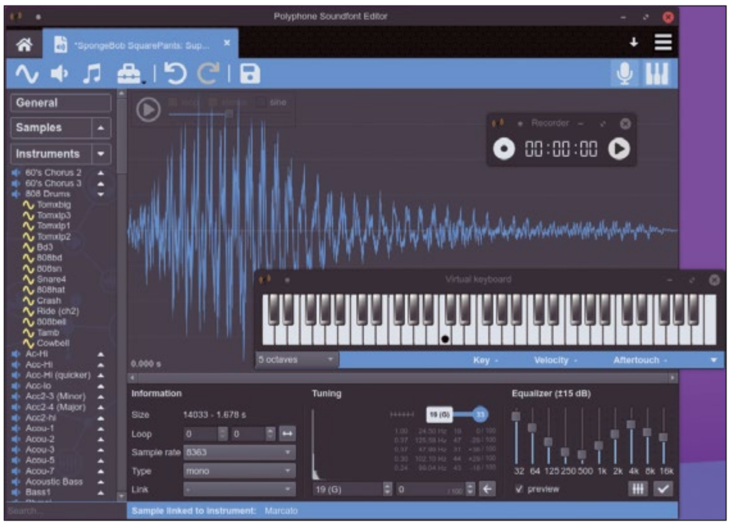
Polyphone


Not only does Polyphone make a great librarian for your entire SoundFont library, it's equally good at playback and even includes several high quality effects.

SoundFonts have been around since the dawn of PC sound cards in the early 1990s. Back then, they allowed games developers and musicians to create music from short snippets of sound (samples) that could be augmented with a few parameters to describe their amplitude over time, with perhaps some filtering, to recreate an instrument. These parameters and deviations also meant several instruments could be built from the same samples, or from different sets of the same samples, giving musicians a wide musical palette within the limited capabilities of those early PCs.

Remarkably, SoundFonts remain popular, and not just in the gaming community. For electronic musicians, SoundFonts have become the standard for cross-platform and cross-application sample-based instruments. A SoundFont can now define an instrument with multiple samples per key, split by the speed you hit a key, across a range of notes, and split across the keyboard. An extended set of parameters deal with those original envelopes, filters, effects, and MIDI control, among dozens of other values. All of which is described in the open SoundFont technical specification from 2006, which became the text-based file format for SoundFont files themselves. This self-documenting and open format has helped to create the thriving online community, which is happy to share its creations (both as samples and as instruments), with many Linux clients, including FluidSynth.

But outside of text-editors, there are very few SoundFont editors to help import samples, organize them into instruments, edit their parameters, and export something you can share. This is what Polyphone does. Polyphone is a well-established Qt-based desktop application that can manage hundreds of samples used in hundreds of instruments, with clever batch import and export functionality, and an ability to access almost every feature in the SoundFont specification. Best of all, it can help graphically by showing sample waveforms and loop points, with these being used as backdrops for amplitude and modulation envelope parameters. There's also a parametric equalizer, a waveform and loop-point editor, tables for mapping samples across key and velocity ranges, modulation sources, and destinations for controllers. You can also create sounds within the application itself, with both primitive recording and editing functions for spontaneous sampling.
There are many hidden options, graphical tweaks, and shortcuts that really help when dealing with potentially hundreds of samples to edit or creating hundreds of instruments, all of which can then be exported without having to edit individual text files. This is what you need when faced with a directory of WAV files that you want to encapsulate within SoundFonts with as little fuss as possible. Polyphone does this. You don't even need to leave the application, because you can preview the sounds from within the application. This is exactly what you need when dealing with banks of samples, such as those from an orchestra or a set of drum machines. Everything can be exported as a single file or as individual instruments. Similarly, you can use Polyphone to rename individual samples within instruments as well as the instruments themselves, which is useful if you're trying something more experimental with the SoundFonts it produces, rather than sample fodder for recombination. If you ever need to do something to a SoundFont, Polyphone will be able to do it.


Polyphone can add envelopes, map samples by velocity or note, enable a filter, and pipe controller data to internal parameters.

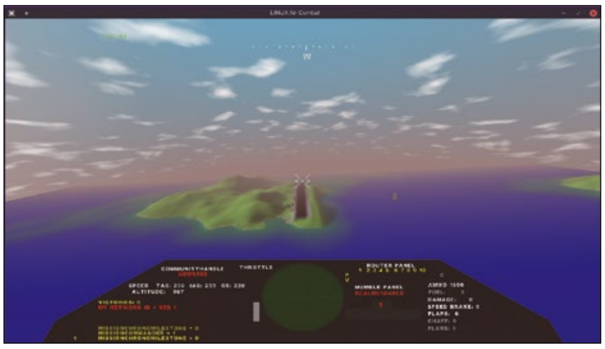
Project Website
https://www.polyphone-soundfonts.com

Flight simulator

Linux Air Combat

Back in the early days of the Commodore Amiga there were a couple of flight simulators that could finally put the potential behind 3D graphics to use. One of those was called Falcon and the other was called F/A-18 Interceptor. Falcon was published by MicroProse and took a very simulator-centric point of view. It had amazing, detailed graphics (for the time) and a complex series of controls to master. Falcon became a classic and is still highly regarded. However, Amigas of the time were too slow to update Falcon's ambitious graphics, and its smoothness was poor. Interceptor, on the other hand, was published by Intellisoft and took the opposite approach. Its graphics were simplistic but effective, as were its flight model and complexity. The end result was a super-smooth 3D flight and combat simulator, which was fast and a lot of fun to play. This is what Linux Air Combat is to the more traditional and complex FlightGear.

Linux Air Combat also looks a little simplistic, but that means it's also particularly smooth and playable, just like F/A-18 Interceptor. It even looks a little like Interceptor with added cloud textures and Gouraud shading over the three-dimensional landscape. This means Linux Air Combat will even run on a Raspberry Pi, and a new build has been created specifically to run on Valve's Steam Deck. The game itself is a 3D-flight simulator loosely based around the aircraft and scenarios of World War II. This includes weapons, radar facilities, heads-up-display, and accurate flight performance according to your payload. There are four tutorial missions to get you started, and a huge variety of multi-axis controllers can be used to accurately control your aircraft with 45 mappable functions. You can even play with a friend on the same computer or with more friends online. It's a lot of fun to play, and a lot easier to get into than its substantial feature list might imply.


One of the latest updates to Linux Air Combat is a build optimized specifically for the Steam Deck.

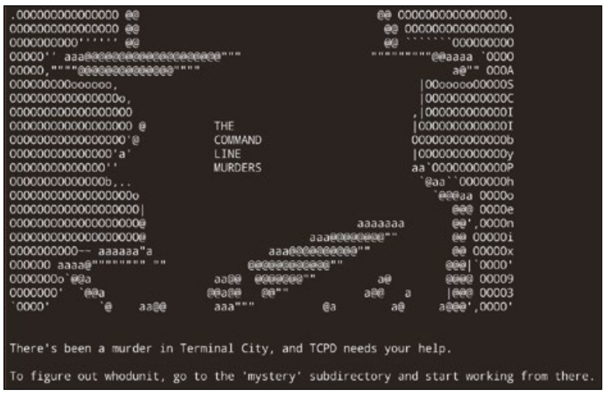
Project Website
https://askmisterwizard.com/2019/LinuxAirCombat/LinuxAirCombat.htm

Terminal

The Command Line Murders

You only have to look through these pages to see that the command line is being used more than ever. This is the opposite of what many of us thought would happen as graphical environments became more efficient and easy to use. Instead, the minimalist, distraction-free terminal, with its remote convenience and low resource use, are fueling a command-line renaissance. This can be a bit intimidating for people who may want to use the command line but perhaps lack the agency or confidence to learn its sometimes unforgiving ways. This is where The Command Line Murders can help.

The Command Line Murders, through necessity, is a text-based game where you need to discover the perpetrator. The brilliant idea behind this is that it's not a standalone game or binary you need to run but is instead a game that exists entirely within the filesystem. The first thing you need to do, for example, is clone the repository or download and unarchive the ZIP file. The next step is to then type `cat instructions` in the root directory to set the scene and read your next steps. To then solve the mystery, you need to navigate through the top-level directories describing the crime scene, the directories interviews, and all the locations they mention. There are over 600 of these files in total, which is a substantial body of work, and one that the developers should be proud of, especially considering the narrative thread that ties everything together. If you get stuck, there's a comprehensive cheat sheet (in Markdown, and as a PDF) that works just as effectively as a command-line guide outside of the game. But when you do successfully make it to the end of the game, not only will you know the solution, you will have also, almost inadvertently, mastered the command line.



```
There's been a murder in Terminal City, and TCPD needs your help.

To figure out whodunit, go to the 'mystery' subdirectory and start working from there.
```

The Command Line Murders is a brilliantly constructed set of over 600 files that teach you to use the command line while you solve a murder mystery.

Project Website
https://github.com/veltman/clmystery

Creating backups with Duplicati backup software

At the Touch of a Button

Duplicati lets you create backups of your data in next to no time – both locally and in the cloud. BY FERDINAND THOMMES

It can't be said often enough: Backups are important. This is something you painfully realize at the moment when you need to restore something and don't have a backup. This magazine regularly presents backup applications and strategies to help readers avoid this kind of situation. Restoring the data in case of an emergency is equally as important as creating a backup, and something you definitely need to test up front.

The more valuable the data, the more important it is to have a well thought out backup strategy. For example, you could store backups locally for ease of access, but at the same time keep another backup in a secure location outside your home. This can be accomplished with a wide variety of tools for a wide variety of requirements. Duplicati [1] lets you do both with ease.

Duplicati Lab

Continuously evolving since 2009, Duplicati promises to securely store encrypted, incremental, and compressed backups on your home network, in the cloud, and on remote file servers. The open source backup client is licensed under the LGPL, and available for Linux, macOS, and Windows. On Linux it can be controlled using the GUI or at the command line.

Duplicati works with a large number of network protocols and services [2]. They include Amazon S3, Box, Dropbox, Google Cloud and Google Drive, Microsoft Azure and OneDrive, OpenStack Storage (Swift), SSH (SFTP), FTP, and WebDAV. The software also works with USB sticks or other external data carriers. You can also store backups on a NAS or on another computer on the same network.

This means that you don't have to trust the interfaces and the encryption techniques offered by proprietary services such as Dropbox, OneDrive, and others, but can ensure that your data is AES-256 encrypted prior to transmission. Alternatively, you can choose GNU Privacy Guard (GPG) for this purpose. Each backup includes a local SQLite-based database that stores information about the remote backup.

Installation

Duplicati is not available in the repositories of most Linux distributions; only Arch Linux provides it in the AUR. The website gives you the source code and packages for macOS, Windows, Synology NAS, Debian, Fedora, and other DEB and RPM-based distributions. A ZIP archive supports portable use without installing or command line-only

Listing 1: Creating a Systemd Service

```
[Unit]
Description=Duplicati web-server
After=network.target
[Service]
Nice=19
IOSchedulingClass=idle
EnvironmentFile=-/etc/default/duplicati
ExecStart=/usr/bin/duplicati-server $DAEMON_OPTS
Restart=always
[Install]
WantedBy=multi-user.target
```

Listing 2: Starting a Systemd Service

```
$ sudo systemctl enable duplicati.service
$ sudo systemctl daemon-reload
$ sudo systemctl start duplicati.service
$ sudo systemctl status duplicati.service
```

operation. The current version for all platforms is 2.0.7.1. You install the Linux packages in the usual way with `apt install` or with `dfn install` on Fedora. Alternatively, you can set up Duplicati via software stores such as Discover on the Plasma Desktop or Gnome Software.

Mono or Not?

One more note before installing: Duplicati works with Mono, an open-source implementation of Microsoft's .NET framework, which is somewhat frowned upon by purist Linux users. The installation occupies up to 400MB on the hard disk due to Mono. Before the first startup, you need to create Duplicati as a systemd service. To do this, open the `/etc/system/system/duplicati.service` file and enter the contents from Listing 1. Then, enable the service using the commands from Listing 2.

After this preliminary work, the software will ask you at first start if more than one user is using the installation; if yes, it will tell you to assign a password. After doing so, a clear-cut interface appears. It launches automatically after you press the button in your browser's taskbar, thanks to a small web server running on port 8200. The advantage of this is that Duplicati also runs on devices such as a NAS without a display.

Your first task is to check the settings. Among other things, you can set the password here, if needed, and grant access on your home network by checking the box. You can also specify how long you want the system to wait after start-up or wake-up before starting a backup it missed previously.

If necessary, you can choose a different language for the user interface or switch to dark mode. It is best to leave the default, *Beta*, as the update channel setting. In the case of anonymous usage statistics, you can decide if and what you want developers to know about your usage of Duplicati. Last but not least, there are many advanced settings available in *Settings*; I will not be going into these settings here.

Besides the settings, the sidebar provides options for adding backup and restore tasks. A click on *Home* shows the current status with backups you already created and the next scheduled task. To the right, you can pause the backup process or limit the connection bandwidth.

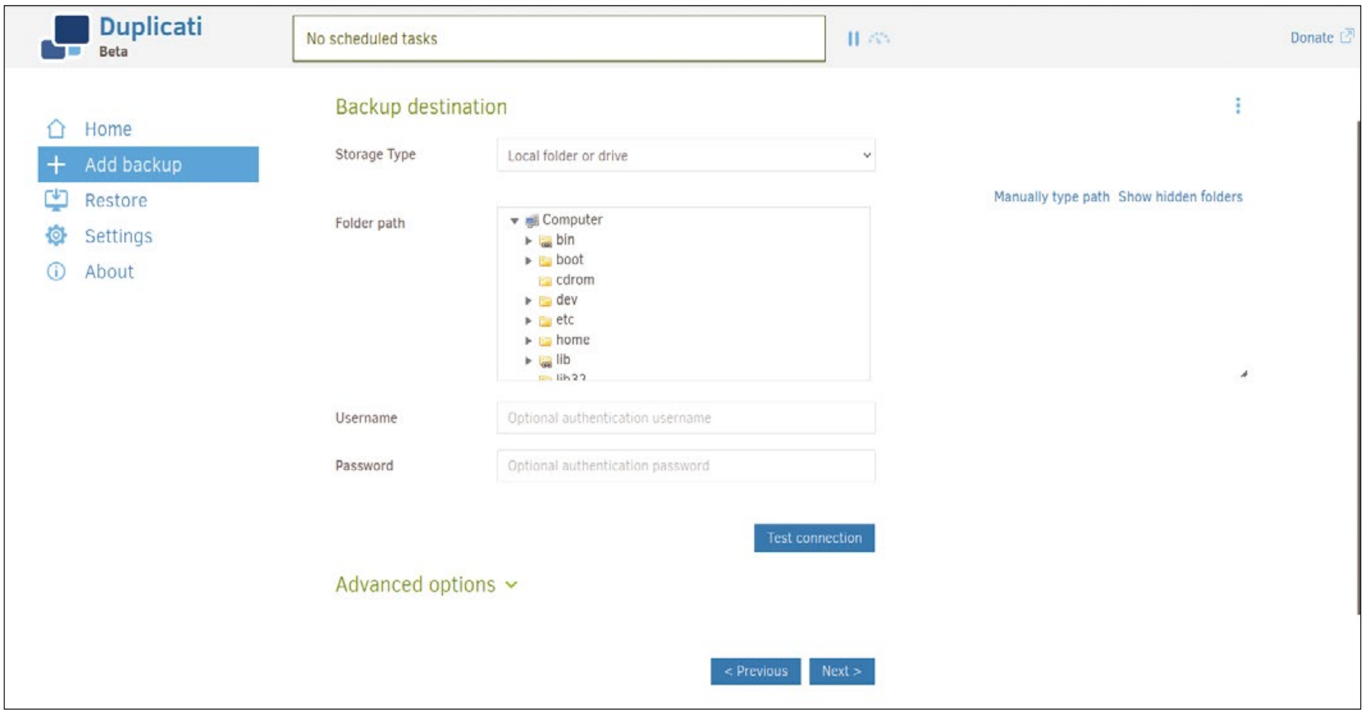
First Backup

To create the configuration for a backup, first create a folder on the target that will store the backups. You can save all the backups in a single folder or create a separate directory, including subfolders, for each backup to ensure a better overview. In testing, I created backups on an external drive, on Google Drive, and on a web server via WebDAV.

For example, to store a backup on an external SSD in the `Duplicati/` folder you created earlier, click *Add Backup* and decide in the first step whether you want to create a new configuration or use an existing one. You will typically choose a new configuration here. After clicking *Next*, you will see five items in a bar at the top that guide you to the finished backup. They are titled *General*, *Destination*, *Source Data*, *Schedule*, and *Options* and are largely self-explanatory.

Use *General* to assign a name and, optionally, a description for the backup. You can then choose a cryptographic method, for which you need to

Figure 1: In Duplicati, you first need to specify the target and then the source of the backup. To define the *Storage Type* you can expand the list of supported protocols and services.



Storage Type: Local folder or drive

Folder path:
```
▾ Computer
  ▸ bin
  ▸ boot
    cdrom
  ▸ dev
  ▸ etc
  ▸ home
  ▸ lib
    lib32
```

Username: Optional authentication username

Password: Optional authentication password

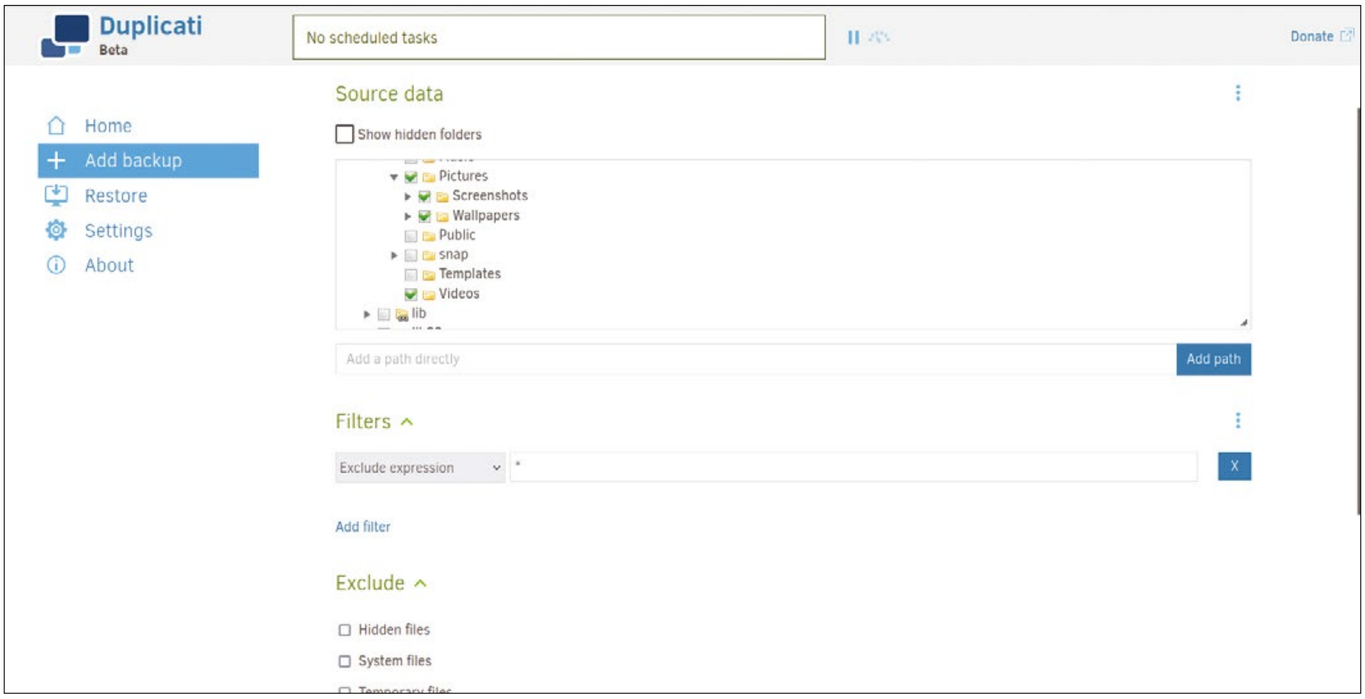
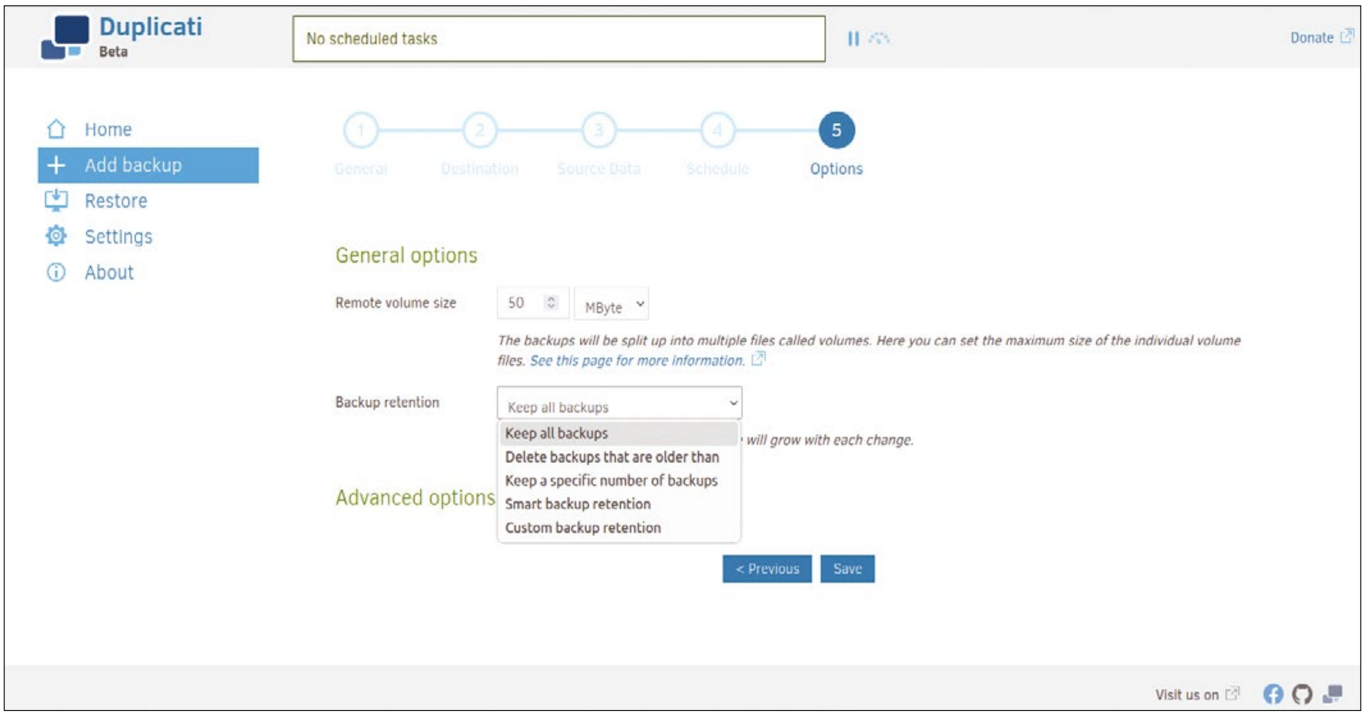


Figure 2: Select the source data you want to back up and then decide which filters to set and which data to exclude.

provide a passphrase. The following applies: The stronger the password, the better the encryption. Alternatively, you can disable encryption. In the next step, select the target (Figure 1) and click *Test connection*. Assuming the software can access the target, you can press *Next* to continue.

The third step is to select the source data (Figure 2). The data can be filtered by predefined criteria to exclude files from the backup. You then need to define a schedule that determines the backup frequency. If the tool misses a specified backup time, for example, because the computer was not switched on or the target was unreachable, Duplicati automatically catches up at the next opportunity.

In the final step (Figure 3), you need to determine the size of the chunks that Duplicati uses to store your data and specify how many backups you want to keep and for how long. The *Backup retention* option takes this decision off your plate and meaningfully reduces the number of backups it keeps as the backups become older.



The backups will be split up into multiple files called volumes. Here you can set the maximum size of the individual volume files. See this page for more information.

will grow with each change.

Figure 3: The last configuration step relates to the general settings. Determine the size of the chunks and the retention time for the backups.

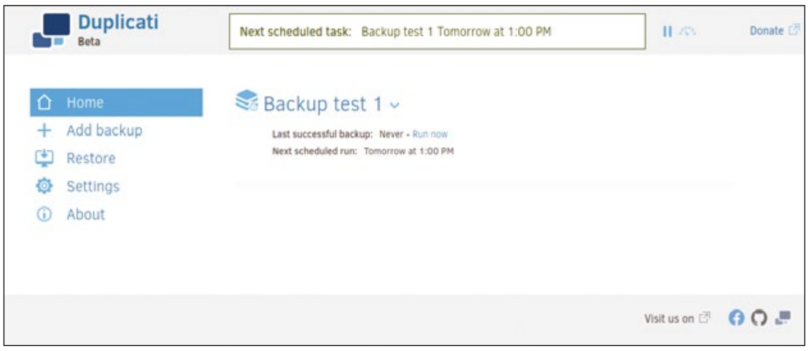
Now save the configuration, which will then appear in *Home* (Figure 4). You can trigger a backup manually or simply wait for the first scheduled backup to happen. A drop-down box lets you display additional options for each backup. Among other things, you can view a logfile of the various backups in this way (Figure 5).

Google Drive

The procedure for backing up to a directory on Google Drive is not that much different from backing up locally. Select Google Drive as the target and specify the path along with the folder you created for the backup earlier on. Then click on *AuthID* to establish the connection between Duplicati and Google Drive (Figure 6). The rest of the procedure follows the steps for a local backup (Figure 7).

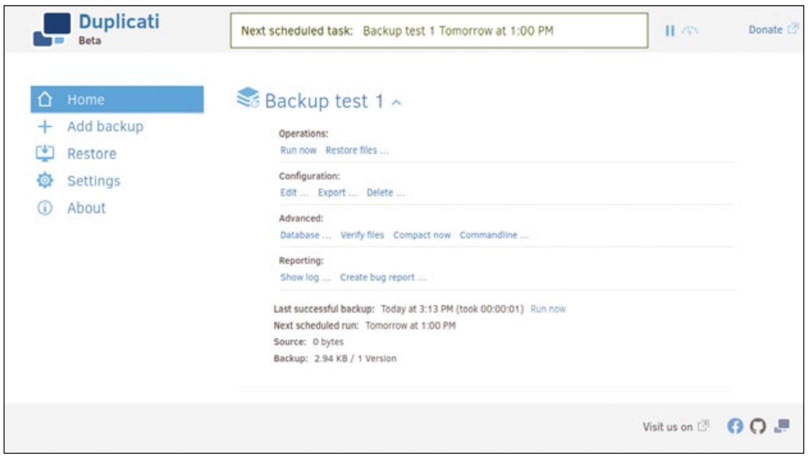
WebDAV

A Nextcloud instance on a remote web server accessible via WebDAV is another potential target. It makes sense to create the *duplicati* folder up front (Figure 8). For the backup, select *WebDAV* as the storage type and check the *Use SSL* box. For the *Server and port*, enter, for example, `nextcloud.example.com` followed by a path, such as `remote.php/webdav/duplicati`. You can usually leave out the port. After entering the username and password for Nextcloud, first check the connection and then start the backup (Figure 9).



Last successful backup: Never - Run now
Next scheduled run: Tomorrow at 1:00 PM

Figure 4: Once the backup is complete, you will see its details in the *Home* tab.



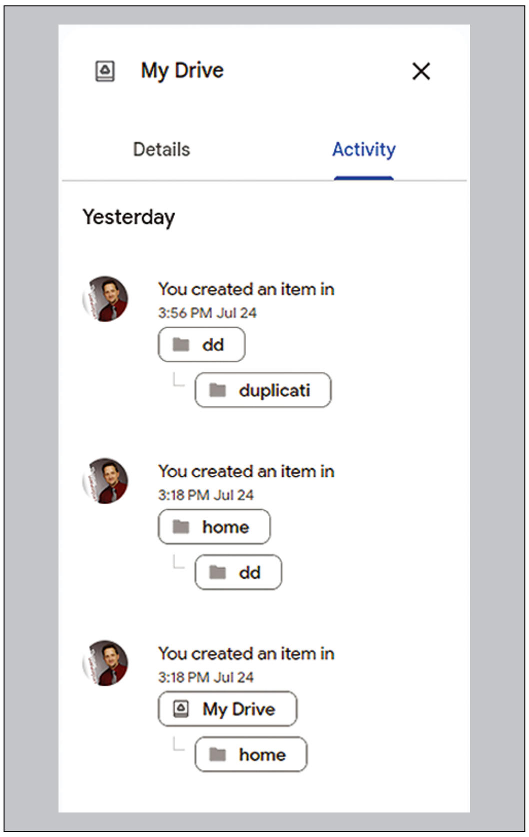
Last successful backup: Today at 3:13 PM (took 00:00:01) Run now
Next scheduled run: Tomorrow at 1:00 PM
Source: 0 bytes
Backup: 2.94 KB / 1 Version

Figure 5: In the *Home* tab, you can open the options for the individual backups by clicking on the drop-down icon to the right of the backup name.

Restoring

Equally important to creating a successful backup is the certain knowledge that the restore process will actually work in the event of an emergency. After selecting *Restore* from the menu, next select the backup to restore. Then press *Next* and select the desired backup (if you have several). Further down, check the box for the entire backup or the boxes for the individual files you want to restore.

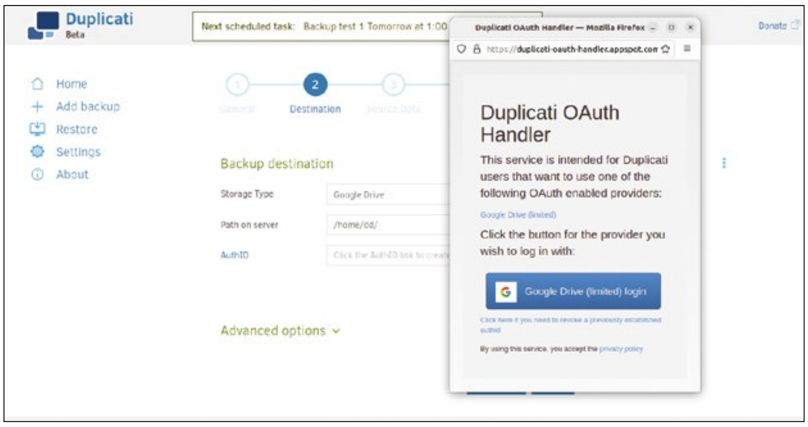
In the *Restore options* window, then specify whether you want Duplicati to restore the files to the original location, or choose a different location – even a different computer – if necessary (Figure 10). In addition, you need to



You created an item in
3:56 PM Jul 24

You created an item in
3:18 PM Jul 24

You created an item in
3:18 PM Jul 24


Duplicati
Beta
Next scheduled task: Backup test 1 Tomorrow at 1:00
Donate
Home
Add backup
Restore
Settings
About
Destination
Backup destination
Advanced options
Duplicati OAuth Handler — Mozilla Firefox
https://duplicati-oauth-handler.appspot.com
Google Drive (limited) login

Storage Type: Google Drive
Path on server: /home/dd/
AuthID: Click the AuthID link to create

Duplicati OAuth Handler

This service is intended for Duplicati users that want to use one of the following OAuth enabled providers:

Google Drive (limited)

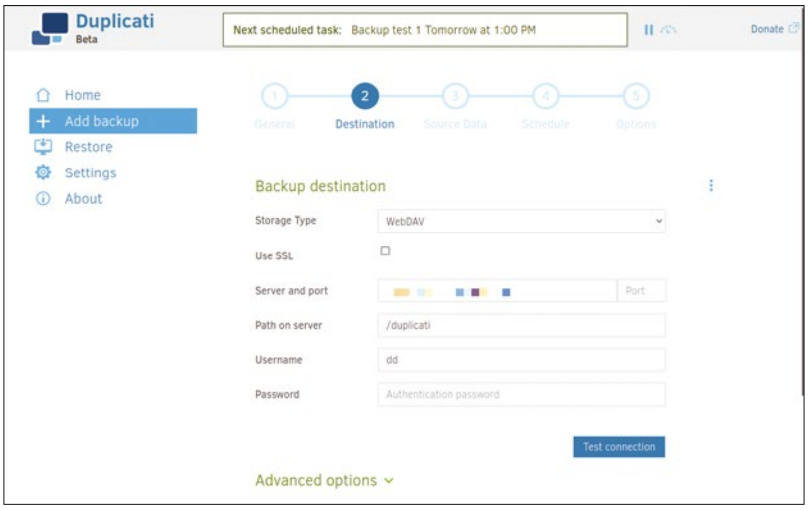
Click the button for the provider you wish to log in with:

Click here if you need to revoke a previously established authid

By using this service, you accept the privacy policy

Figure 6: Authenticate to Google Drive from Duplicati by creating an AuthID, which is automatically copied into the field provided. There is nothing standing in the way of creating a backup now.

Figure 7: Transferring to Google Drive was not exactly a fast process in testing, but, at the end of the day, the encrypted data ended up in the right place.

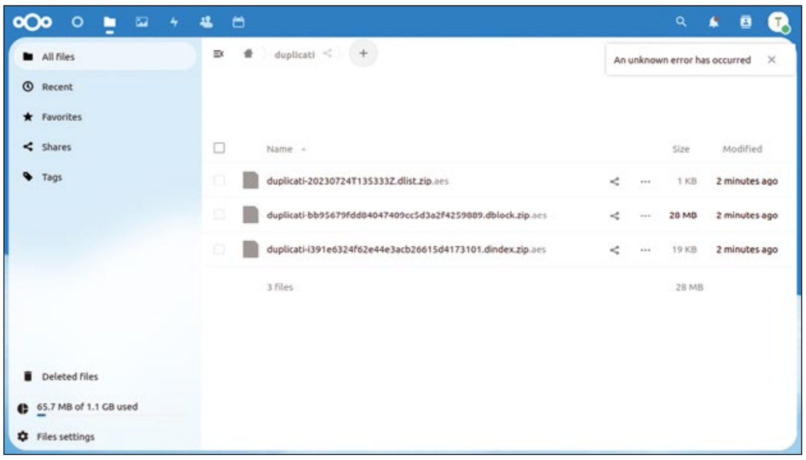


Next scheduled task: Backup test 1 Tomorrow at 1:00 PM

Backup destination

Storage Type	WebDAV
Use SSL	☐
Server and port	Port
Path on server	/duplicati
Username	dd
Password	Authentication password

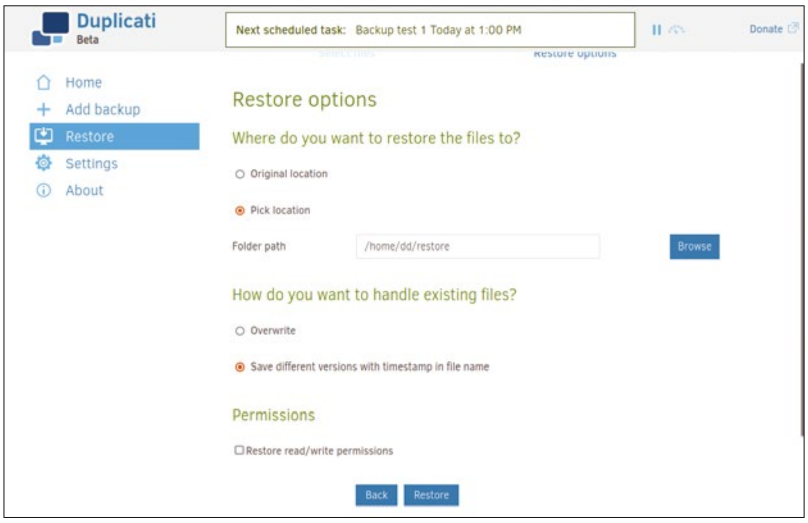
Figure 8: Duplicati supports any service that supports a WebDAV connection. I chose a Nextcloud instance, which runs on a web server on the network.



An unknown error has occurred

Name	Size	Modified
duplicati-20230724T135333Z.dlist.zip.aes	1 KB	2 minutes ago
duplicati-bb95679fdd04047409cc5d3a2f4259809.dblock.zip.aes	28 MB	2 minutes ago
duplicati-i391e6324f62e44e3acb26615d4173101.dindex.zip.aes	19 KB	2 minutes ago
3 files	28 MB	

Figure 9: The backup to Nextcloud also worked without any trouble, and Duplicati made full use of the available bandwidth.



Next scheduled task: Backup test 1 Today at 1:00 PM

Restore options

Where do you want to restore the files to?

○ Original location

● Pick location

Folder path /home/dd/restore

How do you want to handle existing files?

○ Overwrite

● Save different versions with timestamp in file name

Permissions

☐ Restore read/write permissions

Figure 10: After selecting the backup, you decide whether you want to restore the data to the original location or somewhere else – this can also be another computer. Besides this, you can overwrite existing data or have versions of the data, like with Git.

specify whether to overwrite the existing data or whether you want to keep multiple versions with a timestamp in the file name. You can also check *Restore read/write permissions* if required. All you need to do now is press *Restore*, and your data will be put back where it belongs.

Conclusions

Duplicati is a very easy to use backup tool with multiple backup options. The software guides you step by step to your objective with an easy-to-understand menu. Operations are identical for all backup options and on all platforms, with just a few differences relating to the target specification. This simplifies use on multiple operating systems. Opened or locked files are backed up without your intervention thanks to the VSS service [3] on Windows and the Logical Volume Manager (LVM) on Linux.

The range of backup targets, intuitive controls, and the fact that you can manage the encryption options make Duplicati a recommendable tool. It makes no difference whether the target resides locally, on your organization's servers, or in a third-party cloud. I did not encounter any problems during testing. Although Duplicati is intuitive to use, the documentation additionally provides useful assistance in case of issues [4]. It also describes command-line operations, which open up advanced options for backups that are not available in the GUI. ■■■

Info

[1] Duplicati: *https://www.duplicati.com/*

[2] Supported services: *https://duplicati.readthedocs.io/en/latest/01-introduction/#supported-backends*

[3] VSS: *https://learn.microsoft.com/en-us/windows-server/storage/file-server/volume-shadow-copy-service*

[4] Duplicati documentation: *https://duplicati.readthedocs.io/en/latest/02-installation/*

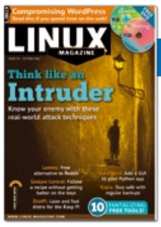
The Author

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

LINUX NEWSSTAND

Order online:
https://bit.ly/Linux-Magazine-catalog

Linux Magazine is your guide to the world of Linux. Monthly issues are packed with advanced technical articles and tutorials you won't find anywhere else. Explore our full catalog of back issues for specific topics or to complete your collection.



#275/October 2023

Think like an Intruder

The worst case scenario is when the attackers know more than you do about your network. If you want to stay safe, learn the ways of the enemy. This month we give you a glimpse into the mind of the attacker, with a close look at privilege escalation, reverse shells, and other intrusion techniques.

On the DVD: AlmaLinux 8.2 and blendOS

#274/September 2023

The Best of Small Distros

Nowadays, all the attention is on big, enterprise distributions supported by professional developers at big, enterprise corporations, but small distros are still a thing. If you're shopping for a Linux to run on old hardware, if you just want a simpler system that is more responsive and less cluttered, or if you're looking for a special Linux tailored for a special purpose, you're sure to find inspiration in our look at small and specialty Linux systems.

On the DVD: 10 Small Distro ISOs and 4 Small Distro Virtual Appliances

#273/August 2023

Podcasting

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

On the DVD: Linux Mint 21.1 Cinnamon and openSUSE Leap 15.5

#272/July 2023

Open Data

As long as governments have kept data, there have been people who have wanted to see it and people who have wanted to control it. A new generation of tools, policies, and advocates seeks to keep the data free, available, and in accessible formats. This month we bring you snapshots from the quest for open data.

On the DVD: xubuntu 23.04 Desktop and Fedora 38 Workstation

#271/June 2023

Smart Home

Smart home solutions will save you time and energy – and, did I mention, you can amaze your friends. This month we show you how to take charge of your home environment with smart devices and open source automation software.

On the DVD: SystemRescue 10.0 and Linux Lite 6.4

#270/May 2023

Green Coding

A sustainable world will need more sustainable programming. This month we tell you about some FOSS initiatives dedicated to energy efficiency, and we take a close look at some green coding techniques in Go.

On the DVD: Fedora 37 Workstation and TUXEDO OS 2

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*.

SC23

Date: November 12-17, 2023

Location: Denver, Colorado

Website: *https://sc23.supercomputing.org/*

SC23 is the international conference for high performance computing, networking, storage, and analysis. Join us in Denver for an exhilarating week of sessions, speakers, and networking. SC is an unparalleled mix of scientists, engineers, researchers, educators, programmers, and developers and who intermingle to learn, share, and grow.

MSP Global

Date: November 14-16, 2023

Location: Nürburgring, Germany

Website: *https://mspglobal.com/*

Managed Service Providers and those who work with them are coming to MSP GLOBAL to develop greater expertise in today's emerging technologies as well as get ahead of tomorrow's trends so their brands can reach higher quality, earn greater trust, and bring in more business. Go full-throttle with your MSP business at the Nürburgring Race Track.

FOSDEM 2024

Date: February 3-4, 2024

Location: Brussels, Belgium

Website: *https://fosdem.org/2024/*

FOSDEM is a free event for software developers to meet, share ideas, and collaborate. Every year, thousands of developers of free and open source software from all over the world gather at the event in Brussels. You don't need to register. Just turn up and join in!

Events

AI DevWorld	Oct 26	Santa clara, California	https://aidevworld.com/
Hybrid Cloud Conference	Oct 26	Virtual Event	https://www.techforge.pub/events/hybrid-cloud-congress-2/
AI DevWorld Live Online	Oct 31-Nov 2	Virtual Event	https://www.linux-magazine.com/
SeaGL 2023	Nov 3-4	Virtual Event	https://seagl.org/
KubeCon + CloudNativeCon North America	Nov 6-9	Chicago, Illinois	https://events.linuxfoundation.org/kubecon-cloudnativecon-north-america/
RISC-V Summit	Nov 7-8	Santa Clara, California	https://events.linuxfoundation.org/riscv-summit/
Open Source Monitoring Conference (OSMC)	Nov 7-9	Nuremberg, Germany	https://osmc.de/
SFSCON 2023	Nov 10-11	Bolzano, Italy	https://www.sfscon.it/
SC23	Nov 12-17	Denver, Colorado	https://sc23.supercomputing.org/
Linux Plumbers Conference	Nov 13-15	Richmond, Virginia	https://lpc.events/
LFN Developer & Testing Forum	Nov 13-16	Budapest, Hungary	https://events.linuxfoundation.org/
MSP Global	Nov 14-16	Nübrgring, Germany	https://mspglobal.com/
The Linux Kernel Maintainer Summit	Nov 16	Richmond, Virginia	https://events.linuxfoundation.org/
Open Source Summit Japan	Dec 5-6	Tokyo, Japan	https://events.linuxfoundation.org/
Open Compliance Summit	Dec 7-8	Tokyo, Japan	https://events.linuxfoundation.org/
Cassandra Summit	Dec 12-13	San Jose, California + Virtual	https://events.linuxfoundation.org/cassandra-summit/
FOSDEM	Feb 3-4 2024	Brussels, Belgium	https://fosdem.org/
DeveloperWeek SF Bay Area	Feb 21-23 2024	San Francisco, California	https://www.developerweek.com/
DeveloperWeek Live Online	Feb 27-29 2024	Virtual Event	https://www.developerweek.com/

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
Dena Friesen
Cover Image
© ntlstudio, 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 Kolibri Druck.

Distributed by Seymour Distribution Ltd, United Kingdom

Represented in Europe and other territories by: Sparkhaus Media GmbH, Bialasstr. 1a, 85625 Glonn, Germany.

Published monthly as Linux Magazine (Print ISSN: 1471-5678, Online ISSN: 2833-3950) by Linux New Media USA, LLC, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA. Periodicals Postage paid at Lawrence, KS and additional mailing offices. Ride-Along Enclosed. POSTMASTER: Please send address changes to Linux Magazine, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA.

WRITE FOR US

Linux Magazine is looking for authors to write articles on Linux and the tools of the Linux environment. We like articles on useful solutions that solve practical problems. The topic could be a desktop tool, a command-line utility, a network monitoring application, a homegrown script, or anything else with the potential to save a Linux user trouble and time. Our goal is to tell our readers stories they haven't already heard, so we're especially interested in original fixes and hacks, new tools, and useful applications that our readers might not know about. We also love articles on advanced uses for tools our readers *do* know about – stories that take a traditional application and put it to work in a novel or creative way.

Topics close to our hearts include:

- Security
- Advanced Linux tuning and configuration
- Internet of Things
- Networking
- Scripting
- Artificial intelligence
- Open protocols and open standards

If you have a worthy topic that isn't on this list, try us out – we might be interested!

Please don't send us articles about products made by a company you work for, unless it is an open source tool that is freely available to everyone. Don't send us webzine-style "Top 10 Tips" articles or other superficial treatments that leave all the work to the reader. We like complete solutions, with examples and lots of details. Go deep, not wide.

We have a couple themes coming up that we could use your help with. Please send us your proposals for thoughtful and practical articles on:

- **Cryptocurrencies**
- **Systemd hacks**

Describe your idea in 1-2 paragraphs and send it to: **edit@linux-magazine.com**.

Please indicate in the subject line that your message is an article proposal.

Authors

Chris Binnie	40	Jon "maddog" Hall	73
Zack Brown	12	Ralf Kirschner	80
Bruce Byfield	6, 24, 34	Vincent Mealing	71
Joe Casad	3	Graham Morrison	84
John Cofield	36	Mike Schilli	46
Mark Crutch	71	Ferdinand Thommes	75, 90
Chris Dock	28	Koen Vervloesem	16, 58, 64
Marco Fioretti	52	Jack Wallen	8

Available Starting November 3

Issue 277 / December 2023

Low Code

Programming is a major cost for many organizations – and not just development shops. Companies in all industries require custom code to support infrastructure, move product, and keep an eye on the business. Wouldn't it be easier if building a program were more like building with Lego blocks? Next month, we take a look inside the low code movement.

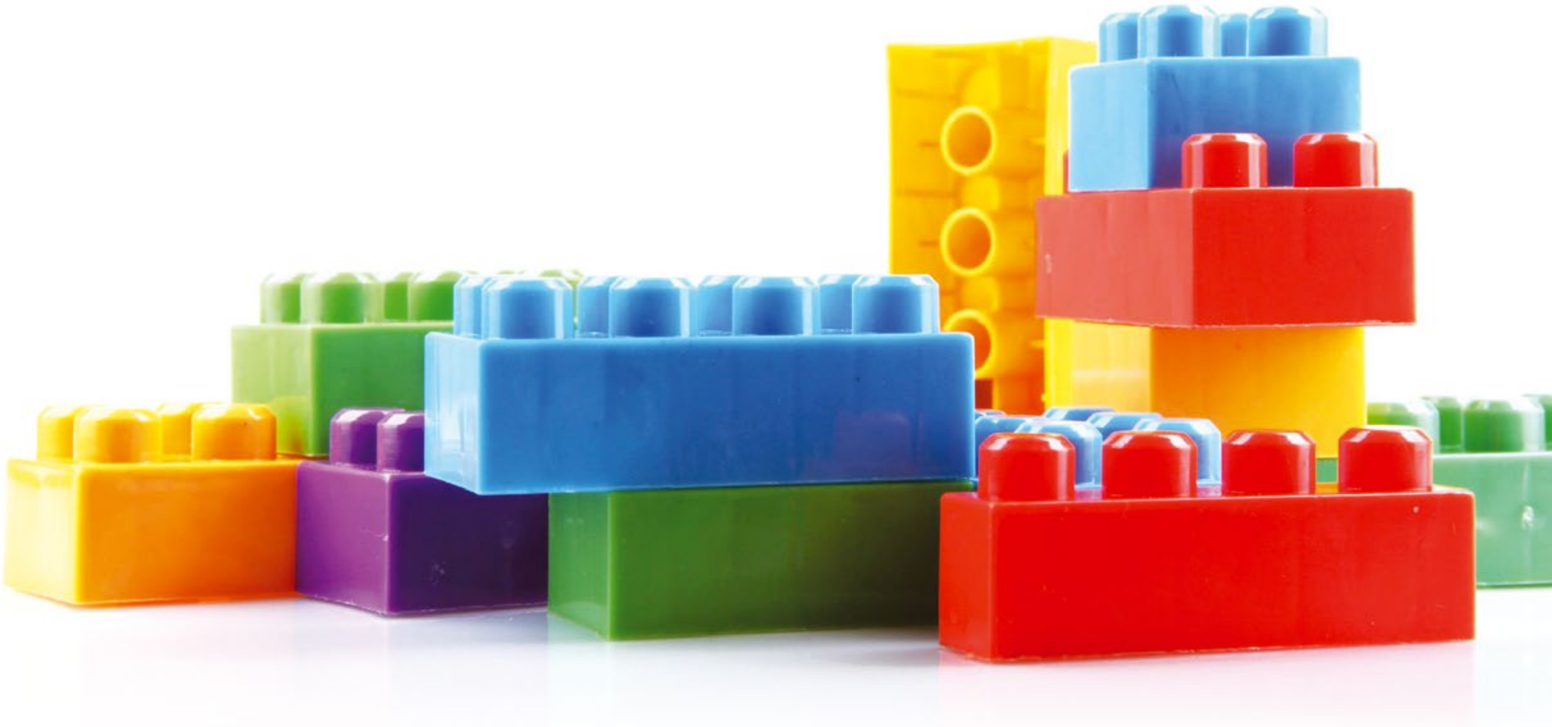


Image © nenovbrothers, 123RF.com

Ultra portable office companion

TUXEDO Aura 14 - Gen3

Aluminum chassis
2 cm thin | 1.4 kg light

14.0'' Full HD display
1920 x 1080 | 100 % sRGB

Intel Core i5-1235U
10 cores | 12 threads

USB-C Power Delivery
49 Wh battery | up to 5 hours

Linux compatible

Up to 5 Years Guarantee

Immediately ready for use

Made in Germany

German Data Privacy

German Tech Support

TUXEDO

tuxedocomputers.com

HETZNER

PURE POWER AT
IMPOSSIBLY LOW PRICES

Experience unshared vCPU power with our new
DEDICATED CLOUD PLANS!

CLOUD SERVER CCX13

- ✓ AMD EPYC™
- ✓ 2 vCPU
- ✓ 8 GB ECC RAM
- ✓ 80 GB NVMe SSD
- ✓ 20 TB traffic inclusive
- ✓ IPv4 inclusive
- ✓ Intuitive Cloud Console
- ✓ Location Germany

monthly $ **13.40**

CLOUD SERVER CCX43

- ✓ AMD EPYC™
- ✓ 16 vCPU
- ✓ 64 GB ECC RAM
- ✓ 360 GB NVMe SSD
- ✓ 40 TB traffic inclusive
- ✓ IPv4 inclusive
- ✓ Intuitive Cloud Console
- ✓ Location Germany

monthly $ **103.55**

All prices exclude VAT and are subject to the terms and conditions of Hetzner Online GmbH. Prices are subject to change. All rights reserved by the respective manufacturers.

cloud.hetzner.com