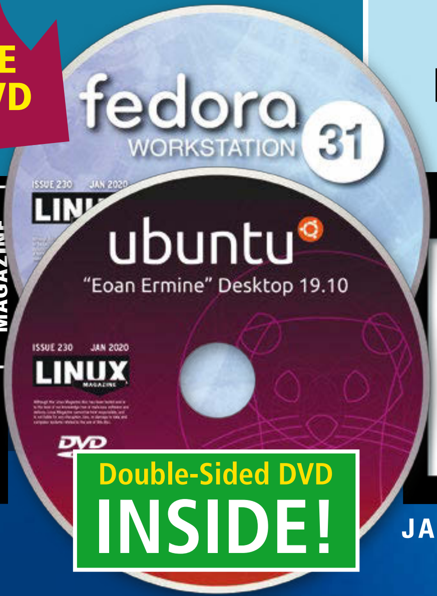
FREE DVD



FLATPAK RISES
New age package system reaches the desktop

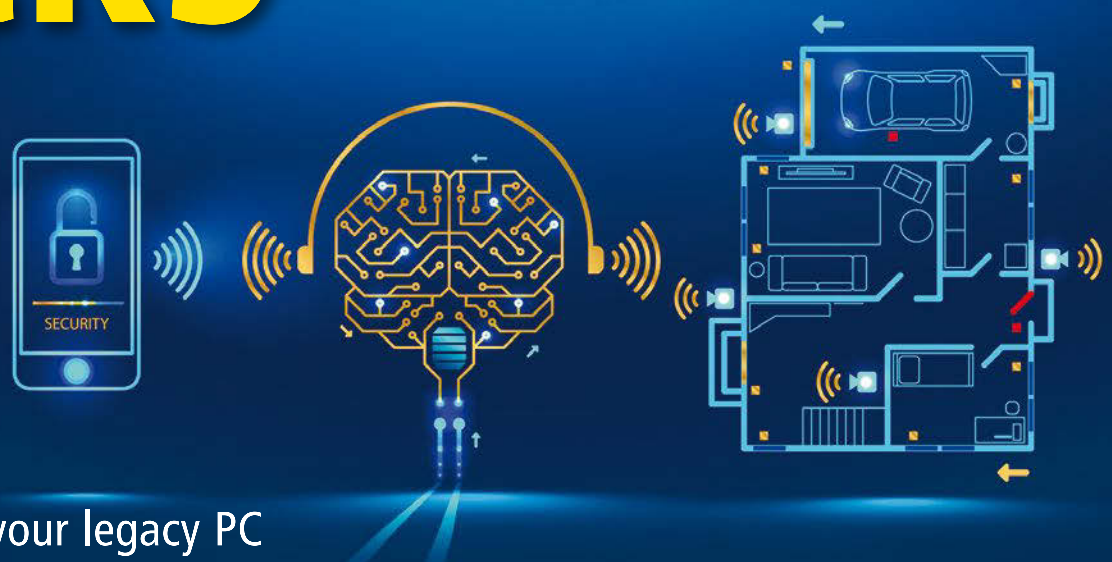
AUTOMATION TRICKS

LINUX PRO MAGAZINE

JANUARY 2020

AUTOMATION TRICKS

Building custom solutions for the intelligent home



Make Alexa talk to a Raspberry Pi

Mozilla WebThings
An open platform for IoT

PXE Boot
Breathe new life into your legacy PC

Fun with Python
Build a giant electronic scoreboard

Symphytum
Simple database for everyday things

bpftrace
Watching the Linux Kernel

LINUXVOICE

- Push notifications with Gotify
- maddog calls for more women coders
- Managing ebooks with Calibre

FOSSPicks
- Blender 2.8
- Amass security tool
- Cookbook recipe manager

Tutorial
- Build your own Mastodon client

Issue 230
Jan 2020
US$ 15.99
CAN$ 18.50

CALIFORNIA DREAMING

Dear Reader,

We in Linux publishing have spent a lot of time holding Microsoft accountable for all the FUD and monkey business they have subjected us to through the years, so it is only fair to acknowledge them when they take a positive step. Microsoft has actually been doing better recently – I have written about Redmond's newfound support for Linux and their open sourcing of core development tools. This month the big news is the announcement that Microsoft will "honor California's new privacy rights throughout the United States."

A little over a year ago, the State of California passed the California Consumer Privacy Act (CCPA), which will take effect on January 1, 2020. The CCPA is a landmark bill that takes on the pertinent and perplexing issue of data privacy in the Internet age. The act establishes the following rights for residents of California:

- The right to know what personal information is collected, used, shared, or sold, both as to the categories and specific pieces of personal information;
- The right to delete personal information held by businesses and, by extension, a business's service provider;
- The right to opt-out of the sale of personal information. Consumers are able to direct a business that sells personal information to stop selling that information. Children under the age of 16 must provide opt-in consent, with a parent or guardian consenting for children under 13;
- The right to non-discrimination in terms of price or service when a consumer exercises a privacy right under CCPA.

The law could have big implications on how Internet companies capture and market user data. California is too big of a market for the big companies to ignore, so they will be forced to comply with it – at least for California residents. Companies that wish to restrict the new rules to only California viewers will need some way of sorting out who is or isn't from California and offering two different web pages for the California and non-California views, a layer of complication that could cause other companies to join Microsoft in simply applying the rules for everyone, but companies that are heavily dependent on selling data might find it difficult to give up the revenue.

According to Microsoft VP Julie Brill, "We are strong supporters of California's new law and the expansion of privacy protections in the United States that it represents. Our approach to privacy starts with the belief that privacy is a fundamental human right and includes our commitment to provide robust protection for every individual. This is why, in 2018, we were the first company to voluntarily extend the core data privacy rights included in the European Union's General Data Protection Regulation (GDPR) to customers around the world, not just to those in the EU who are covered by the regulation. Similarly, we will extend CCPA's core rights for people to control their data to all our customers in the U.S."

CCPA isn't perfect and doesn't solve *all* the problems related to data privacy. For instance, it only applies to large companies and companies that derive over half their revenue from selling consumer information. The companies that it does apply to are probably out there right now developing workarounds. Still, the CCPA is a significant step back in a world that has recently witnessed a continual march to fewer restrictions and more data mining.

At some level, everything that goes on in the business world is about business. This is all good press for Microsoft, but beyond the PR benefits, it is also an interesting chess move for a company that was left a little behind by the Internet giants. Google, Facebook, and other Internet titans are built from the ground up around the dubious endeavor of extracting value from their users' lives. Microsoft is a bit of a newcomer in this space. Anything Microsoft can do to shake up the market and force competitors out of their comfort zones is good for Microsoft – and also good for us in this case.

I have a feeling the last chapter in the story of the search for user privacy hasn't been written yet. The CCPA could prove unworkable, which will strengthen the hand of industry lobbyists, who will certainly be looking to dial up their game. A backlash could flash, which would require consumer advocates to really bear down to get what they want. And maybe, just maybe, in all the smoke and dust (full disclosure: I'm an optimist), the U.S. Congress will actually step up to their responsibilities and pass some meaningful privacy legislation at the national level.

In the meantime, I commend Microsoft for getting this one right.

Joe

Joe Casad,
Editor in Chief

LINUX MAGAZINE

WHAT'S INSIDE

Home automation is no longer the stuff of science fiction. This month we explore some versatile tools for Linux users who are interested in IoT but don't want to surrender the freedom of open platforms and open source. Elsewhere inside:

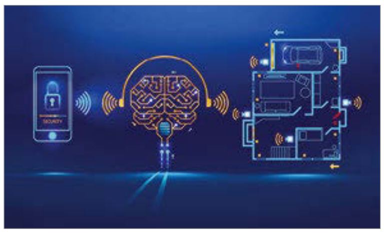
- **PXE Boot with TinyCore** – Your old computer can live on even without a hard drive. Boot across the network with PXE (page 44).
- **Symphytum** – Create a simple and practical database for your notes, recipes, or pine cone collection (page 48).

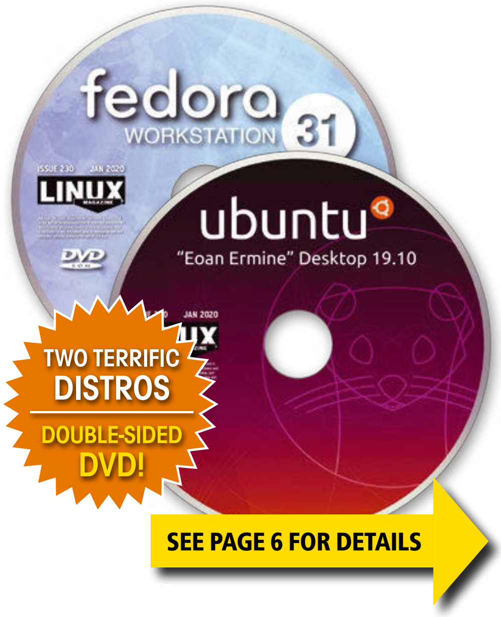
Turn to MakerSpace for a real-world project with the Python Remote Objects Library (Pyro), and check out LinuxVoice for a look at the Gotify push notification tool.

SERVICE

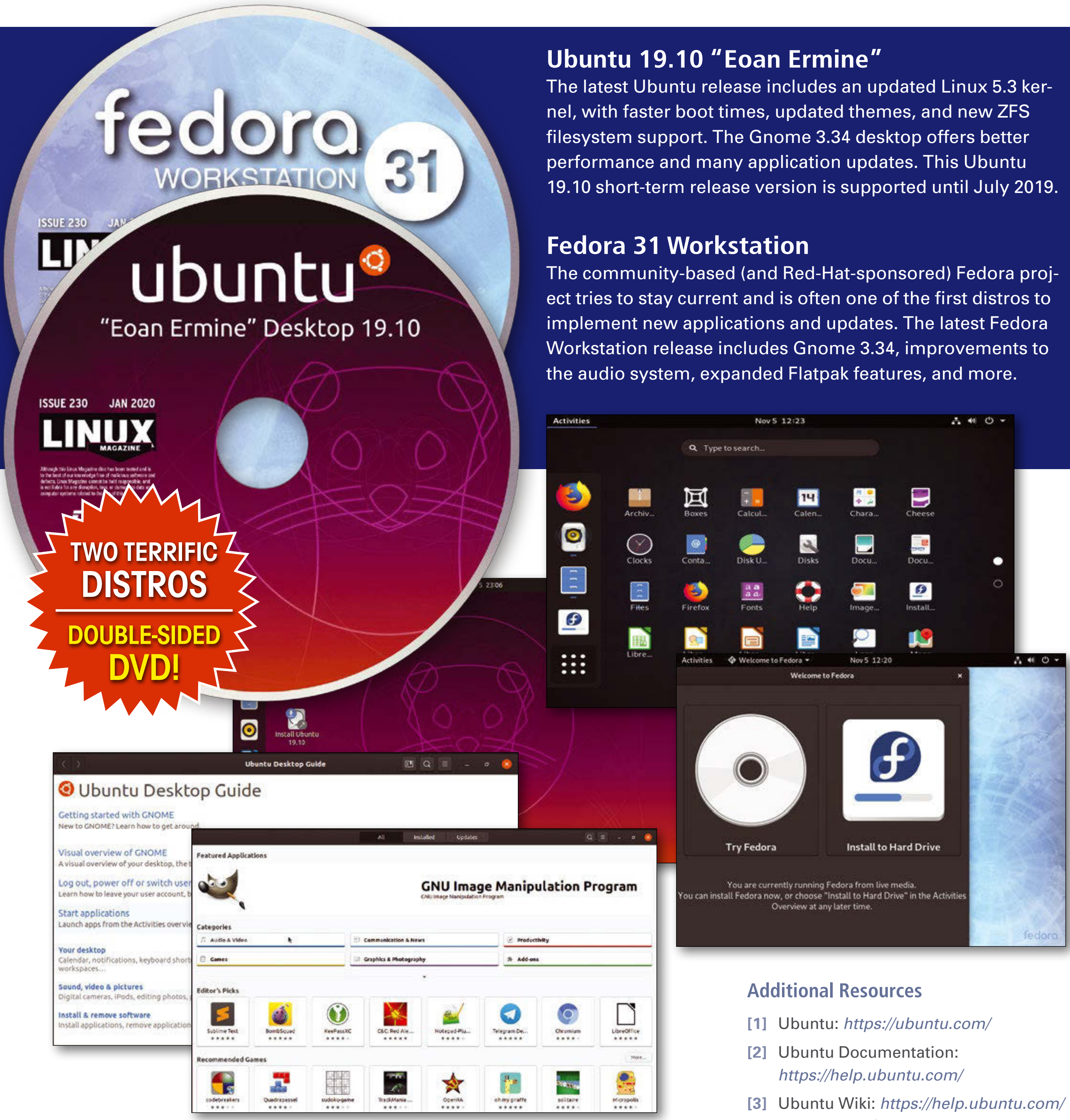
NEWS

COVER STORIES




fedora WORKSTATION 31
ubuntu "Eoan Ermine" Desktop 19.10
TWO TERRIFIC DISTROS
DOUBLE-SIDED DVD!
SEE PAGE 6 FOR DETAILS

On the DVD



You are currently running Fedora from live media.
You can install Fedora now, or choose "Install to Hard Drive" in the Activities Overview at any later time.

Ubuntu 19.10 "Eoan Ermine"

The latest Ubuntu release includes an updated Linux 5.3 kernel, with faster boot times, updated themes, and new ZFS filesystem support. The Gnome 3.34 desktop offers better performance and many application updates. This Ubuntu 19.10 short-term release version is supported until July 2019.

Fedora 31 Workstation

The community-based (and Red-Hat-sponsored) Fedora project tries to stay current and is often one of the first distros to implement new applications and updates. The latest Fedora Workstation release includes Gnome 3.34, improvements to the audio system, expanded Flatpak features, and more.

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

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

Additional Resources

[1] Ubuntu: *https://ubuntu.com/*

[2] Ubuntu Documentation: *https://help.ubuntu.com/*

[3] Ubuntu Wiki: *https://help.ubuntu.com/community/CommunityHelpWiki*

[4] Fedora Workstation: *https://getfedora.org/en/workstation/*

[5] Fedora Documentation: *https://docs.fedoraproject.org/en-US/fedora/f31/*

openSUSE + LibreOffice Conference

LibreOffice
The Document Foundation

openSUSE®

A Conference for
Open Source Developers
is Coming this Fall

events.opensuse.org

NEWS

Updates on technologies, trends, and tools

THIS MONTH'S NEWS

Microsoft Edge Coming to Linux

For the longest time, any Linux user needing to work with a Microsoft browser had few options. There was always IEs4Linux, but that option tended to install out-of-date, buggy versions of the software. Users could also run a version of Windows within a virtual machine, but that meant actually running Windows.

All of that changes in 2020, when Microsoft Edge comes to Linux. In the "State of the Browser: Microsoft Edge" session at Ignite 2019, it was finally announced that Microsoft was, in fact, bringing their new browser to Linux (*https://myignite.techcommunity.microsoft.com/sessions/79341?source=sessions*).

The new Microsoft browser is built around the open source Chromium browser (*https://www.chromium.org/*), but this won't simply be a rebuild and rebrand. Microsoft plans on being actively involved as a contributor to Chromium's open source development. That means any development work done for Edge could find its way to Chromium. So, even if users don't opt to install Edge Chromium on Linux, if they use Chromium they will benefit from Microsoft-contributed work.

Of course, one looming question remains: Will Linux users give Microsoft's browser a chance? Only time will tell. The official release of Edge Chromium for Windows and macOS is January 15. As of now, there is no definitive release date, nor any indication as to how Edge Chromium will be installed on Linux (be it official packages, snaps or flatpaks, or some other method).

Open Invention Network Backs Gnome Project Against Patent Troll

The Gnome Project was recently sued by a company called Rothschild Patent Imaging for a patent related to the Shotwell photo manager. The Gnome community has just announced that it is counter-suing Rothschild, which they refer to as a patent troll (*https://www.gnome.org/news/2019/10/gnome-files-defense-against-patent-troll/*).

Keith Bergelt, OIN's CEO, said (*https://www.zdnet.com/article/open-invention-network-comes-to-gnomes-aid-in-patent-troll-fight/*) in his keynote at Open Source Summit, Europe, "Rothschild is a bad company. This is an entity that's antithetical to the goals of innovation. It will sue foundations. It will sue not for profits. It will sue individuals. It will sue corporations. Their playbook is to establish a pattern of wins through relatively modest settlements," which can get other businesses to pay up without a fight.

Gnome turned down the offer to settle for a five-figure sum in order to sue Rothschild and challenge the patent. The Gnome community has established the "Gnome Patent Troll Defense Fund" (*https://secure.givelively.org/donate/gnome-foundation-inc/gnome-patent-troll-defense-fund*) to raise money for this suit and similar attacks.

© Daniele Carabini, 123RF

Fedora 31 Released

The Red-Hat-sponsored Fedora community has announced the release of Fedora 31, the latest version of Red Hat's community distribution (*https://www.redhat.com/en/about/press-releases/fedora-31-now-generally-available*).

Fedora comes in many different editions – each targeting a different workload. Fedora Workstation and Fedora Server are aimed at developers using Fedora for development and then testing their apps on servers. Other editions include Fedora CoreOS, Fedora IoT and Fedora Silverblue.

Fedora Workstation is among the most popular distributions and is reportedly the preferred distro of Linus Torvalds. Fedora 31 Workstation comes with Gnome 3.34 and many tools and features for general users as well as developers. Gnome 3.34 brings significant performance enhancements, which will be especially noticeable on lower-powered hardware.

Fedora 31 Workstation also expands the default uses of the Wayland graphics system, including allowing Firefox to run natively on Wayland under Gnome instead of the XWayland backend.

According to Matthew Miller, the Fedora Project Leader, "The Fedora Project aims to bring leading-edge innovation to our users, and Fedora 31 delivers on that by bringing some of the latest advancements in open source technology to the operating system."

One of the reasons Torvalds and many other developers use Fedora is the fact that it is often one of the earliest distributions to introduce new libraries and packages, which developers can test against their own projects.

Fedora 31 comes with updated compilers and languages, including NodeJS 12, Perl 5.30, and Golang 1.13. Additionally the "python" command will now refer to Python 3.

It also comes with support for Cgroupsv2, bringing kernel-level support for the latest features and functionality around cgroups in the base packages of Fedora 31.

Fedora 31 also adds support for RPM 4.15, the latest version of the RPM Package Manager for enhanced performance and stability across all versions of Fedora.

All editions of Fedora are available for free. You can download them here (*https://getfedora.org/*).

openSUSE OBS Can Now Build Windows WSL Images

As Windows Subsystem for Linux (WSL) is becoming a critical piece of Microsoft's cloud and data-center audience, openSUSE is working on technologies that help developers use distributions of their choice for WSL. Users can run the same WSL distribution that they run in the cloud or on their servers.

The core piece of openSUSE's WSL offering is the WSL appx files, which are basically zip files that contain a tarball of a Linux system (like a container) and a Windows exe file, the so called launcher.

An openSUSE blog explains (*https://lizards.opensuse.org/2019/10/09/opensuse-wsl-images-in-obs/*) that "building a container is something SUSE's Open Build Service (OBS) can already do fully automatic by means of Kiwi. The launcher as well as the final appx however is typically built on a Windows machine using Visual Studio by the developer."

As a result of this work, OBS can actually build the WSL appx from sources. Anyone can build their WSL distribution. However, since the files are signed by openSUSE and not Microsoft, you will need additional steps to run them on Windows 10 machines.

© Sergei Popov, 123RF

MORE ONLINE

Linux Magazine
www.linux-magazine.com

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

High-Performance Python – Distributed Python • Jeff Layton
As the last few articles have pointed out, Python is becoming an important language in HPC and is arguably the most important language in data science.

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

Test JavaScript programs with Karma
Tim Schürmann
Web developers can test increasingly complex JavaScript applications in multiple browsers with the Karma testing environment.

Static code analysis finds avoidable errors
Tobias Eggendorfer
Static code analysis tools like JSLint, Splint, RATS, and Coverity help you find code vulnerabilities.

DNSSEC-aware DNS caching with Unbound
Hans-Cees Speel
If you don't have access to a DNSSEC-aware name server, you can set up your own with Unbound.

Sudo Vulnerability

'sudo' is one of the most useful Linux/UNIX commands, allowing users without root privileges to manage administrative tasks. However, a new vulnerability was discovered in sudo that gives users root privileges.

"When sudo is configured to allow a user to run commands as an arbitrary user via the ALL keyword in a Runas specification, it is possible to run commands as root by specifying the user ID -1 or 4294967295," according to the sudo advisory (*https://www.sudo.ws/alerts/minus_1_uid.html*).

The vulnerability allows users with sudo privileges to run commands as root even if the Runas specification explicitly disallows root access as long as the ALL keyword is listed first in the Runas specification.

Sudo developers have already released a patch to fix the vulnerability. Update your systems now.

Hetzner Launches New Ryzen-Based Dedicated Root Servers

Hetzner is an internet hosting company and data center operator out of Germany that provides dedicated hosting, shared web hosting, virtual private servers, managed servers, domain names, SSL certificates, storage, and cloud solutions. Recently the company announced the launch of a new line of Ryzen-based dedicated root servers that offer a significant boost in performance for customers across all of their services.

According to Tommy Giesler, Product Manager for Dedicated Root Servers at Hetzner Online, "All four servers are built to handle applications that have high multithreading requirements." He continued that "they're also great as general entry level servers for people with heavy workloads."

The new lineup consists of the AX41 and AX41-NVME, which are based on the Ryzen 5 3600 CPU (with 6 cores and 12 threads), and can be combined with either two 2TB HDDs or two 512GB NVMe SSDs. Each of those servers has 64GB of DDR4 RAM. The AX41 and AX41-NVMe start at $39 a month, with a one-time setup fee of $39. Customers can opt to upgrade the memory on those servers to ECC RAM for just $5 a month for increased data integrity.

A step up from the base models are the AX51 and AX51-NVMe. These servers are based on the Ryzen 7 3700X (with 8 cores and 16 threads), and can be combined with either two 8TB HDDs or two 1024GB NVMe SSDs. Both models include 64GB of DDR4 ECC RAM. The AX51 and AX51-NVMe are available starting at $59 a month plus a one-time setup fee of $59.

For more information visit *https://www.hetzner.com/dedicated-rootserver/matrix-ax*.

IBM Joins the Mayflower Autonomous Ship Project

IBM has announced that it is joining the Mayflower Autonomous Ship project. The Mayflower project, led by the marine research organization ProMare, has the goal of building and sailing an autonomous ship across the Atlantic from Plymouth, England to Plymouth, Massachusetts, to commemorate the 400th anniversary of the Pilgrims who landed in America in 1620. IBM will provide AI for the mission and will use its expertise to help the new Mayflower "navigate autonomously and avoid ocean hazards."

According to the press release, "Putting a research ship to sea can cost tens of thousands of dollars or pounds a day and is limited by how much time people can spend onboard – a prohibitive factor for many of today's marine scientific missions," said Brett Phaneuf, a Founding Board Member of ProMare and Co-Director of the Mayflower Autonomous Ship project (together with fellow Board Member Fredrik Soreide). "With this project, we are pioneering a cost-effective and flexible platform for gathering data that will help safeguard the health of the ocean and the industries it supports."

If successful, this will be the first self-navigating, full-sized vessel to cross the Atlantic.

Zack's Kernel News

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

By Zack Brown

Trusted Computing and Linux

Sumit Garg posted a new version of the Trusted Keys subsystem for the Linux kernel, essentially targeting support for Trusted Platform Module (TPM) devices.

The general idea behind TPM technology is that the TPM chip manages access to a given device by encrypting its firmware and creating a corresponding hash value that is stored on a central server. When the system tries to use the device, the TPM hashes the firmware and compares it with what's on the central server. If they match, the user can use the device. Otherwise, they can't.

The goal is to prevent computer system owners from controlling their own systems and to give control to large companies such as Microsoft, who can then make decisions about what software is or is not allowed to be used on that system.

The benefits are enormous! For example, streaming copyrighted content can be handled without fear of piracy, because the large company can prevent pirating software from running on the system. That's the theory.

The drawback is that users can't control their own computers, and they get locked into a dependent relationship with whichever company controls their system. Naturally, there is a lot of money and energy being put into getting these types of patches through the gauntlet of kernel maintainers and up through Linus Torvalds, for inclusion in the main kernel tree.

Linus has traditionally been willing to accept Trusted Platform patches, but only to the extent that they helped, rather than hindered, users' abilities to control their own systems. You can imagine the debates between developers trying to implement features to wrest control of users' systems, and Linus cherry-picking only those aspects of those patches that actually kept control in the hands of users.

In the current discussion, Mimi Zohar asked for more information about the key signing and verification process. In particular, she wanted to know if the TPM's secret key, which it used for generating all the other keys, could ever be accessed by the user. Sumit replied no that this wasn't possible. The key was permanently locked into the TPM chip and represented part of the system-on-a-chip (SoC) service offered by the company producing the TPM device.

Sumit's code was split into several patches, and these were examined independently.

The first two patches enabled registering shared memory with the Trusted Execution Environment (TEE). The TEE is the environment that needs to be created by the various TPM devices, such that it has control over the movement of all data, to ensure that nothing happens that goes against the controlling company's policies. If a fully isolated environment cannot be created, the company can't verify its own control.

The third patch added support for blocking user access to the TEE to obtain the TPM's trusted keys. If the users could access those trusted keys, they could potentially violate the integrity of the TEE.

And Sumit's remaining several patches added support for the TEE's trusted keys.

Janne Karhunen remarked that he had implemented something similar to this. However, instead of supporting an external controlling company, he said, "my thought was to support any type of trust source. Remote, local, or both. Just having one particular type of locally bound 'TEE' sounded very limited, especially when nothing from the TEE execution side is really needed for supporting the kernel crypto. What you really need is the seal/unseal transaction going somewhere and where that somewhere is does not matter much. With the user mode helper in between, anyone can easily add their own thing in there."

Sumit pointed out that a generic TEE, of the sort Janne had described, was already in the Linux kernel and pointed to `Documentation/tee.txt` for reference.

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**.

Sumit also mentioned that his patches supported arbitrary "trust sources," so long as they implemented a few special library functions.

But Sumit also questioned some of Janne's statement – particularly the idea of having a user-mode helper standing in the middle of the trusted network. Sumit said, "Isn't actual purpose to have trusted keys is to protect user-space from access to kernel keys in plain format? Doesn't user mode helper defeat that purpose in one way or another?"

Janne remarked in reply, "CPU is in the user mode while running the code, but the code or the secure keydata being [used] is not available to the 'normal' userspace. It's like microkernel service/driver this way. The usermode driver is part of the kernel image and it runs on top of a invisible rootfs." Janne continued, "I chose the userspace plugin due to this; you can use userspace aids to provide any type of service. Use the crypto library you desire to do the magic you want."

The debate continued in very polite terms, but the battle lines were, once again, clearly drawn. Janne underscored the issue in a subsequent email, saying, "Does the TEE you work with actually support GP [Global Platform standards] properly? Can I take a look at the code?" The Global Platform TEE standard is an open framework for multiple service providers to work together to include their separate products in the secured TEE environment.

Janne continued, "Normally the TEE implementations are well-guarded secrets and the state of the implementation is quite random. In many cases keeping things secret is fine from my point of view, given that it is a RoT [Root of Trust] after all. The secrecy is the core business here. So, this is why I opted the userspace 'secret' route – no secrets in the kernel, but it's fine for the userspace."

That's the key debate: "No secrets in the kernel" means the human owner of the computer has control of the system and can implement anything they want in conjunction with the TEE.

Janne also remarked, "The fundamental problem with these things is that there are infinite amount of ways how TEEs and ROTs can be done in terms of the hardware and software. I really doubt there are 2 implementations in existence that are even remotely compatible in real life. As such, all things TEE/ROT would logically really belong in the userland."

From the perspective of the corporate control advocates, however, giving the machine owner this level of control reduces the TEE's security. As Sumit put it:

"In our case TEE is based on ARM TrustZone which only allows TEE communications to be initiated from privileged mode. So why would you like to route communications via user-mode (which is less secure) when we have standardized TEE interface available in kernel?"

He asked Janne to "elaborate here with an example regarding how this user-mode helper will securely communicate with a hardware based trust source with other user-space processes denied access to that trust source?"

Janne explained, "The other user mode processes will never see the device node to open. There is none in existence for them; it only exists in the ramfs based root for the user mode helper."

Janne added, "Layered security is generally a good thing, and the userspace pass actually adds a layer, so not sure which is really safer?"

As I read this exchange, Janne is attempting to goad Sumit into affirming that the additional security he wants is exactly the elimination of the machine owner's ability to keep control. Janne is apparently essentially saying, "Security issues? What security issues?" And inviting Sumit to say that it's still possible for the machine owner to insert whatever they want into the TEE pipeline, which, of course, is exactly what Linux itself is supposed to be able to do.

But Janne got more and more explicit as the conversation proceeded. At one point he said, "The fundamental problem with the 'standardized kernel tee' still exists – it will never be generic in real life. Getting all this [patch submission] in the kernel will solve your problem and sell this particular product, but it is quite unlikely to help that many users."

And, even more explicitly, Janne remarked, "there is no way to convince op-tee or any other tee to be adopted by many real users. Every serious user can and will do their own thing, or at very best, buy it from someone who did their own thing and is trusted. There is zero chance that samsung, huawei, apple, nsa, google, rambus, payment system vendors … would actually share the tee (or probably even the interfaces). It is just too vital and people do not trust each other anymore."

The discussion petered out shortly afterwards. However, Sumit did not give up and submitted more patches later. Again, the owner-friendly elements were seen as acceptable, while the rest was seen as still problematic.

In this kind of debate, I ask myself if these sorts of features are inevitable in Linux. Will Linux definitely some day support the kind of Trusted Computing platform that could lock users out of controlling their own system? In other words, is there some sort of conceivable scenario in which these companies sneak a certain set of features through the development process and then Linus finds himself unable to undo those changes, because it would break too much user space that has already come to depend on it?

Another way of putting it might be: What if we discovered, today, that a basic element of networking could be used to implement this kind of Trusted Computing in Linux? Would Linus be willing to remove that element, knowing that it was generally regarded as essential? Or would he accept as inevitable the creation of these Linux-based Trusted Computing features?

Load Balancer Improvements

Vincent Guittot pointed out that the Linux load balancer had gotten a bit out of whack recently. Various improvements had made certain heuristics pointless, but those heuristics had not yet been removed. He also pointed out that not all CPU imbalances were based on load, while the load balancer calculated everything based on load. Consequently, Vincent felt there was room for further improvement along those lines.

He posted some patches to clean up things. Among other things, he consolidated the balancing logic into only three functions – one to identify the busiest group of processes, another to check if there's an imbalance, and a third to decide which processes to move in order to balance the load better.

Peter Zijlstra was very happy to see these patches; he and Valentin Schneider offered technical suggestions and documentation fixes. The three of them went back and forth for awhile, without disputes or controversies. It was a very forward-moving collaboration.

This is no guarantee that the code will go into the kernel. Yes, it's excellent to make the load balancer more meaningful and remove arbitrary logic and so on. And it's excellent to see unidirectional progress in the mailing list discussion. However, there are still obstacles that might arise between Vincent's patch set and inclusion in the main kernel tree – security issues and whatnot.

The main problem, especially with something like the load balancing code, is simply the impossibility of knowing how people use their systems. Obviously, if the kernel knew exactly how the system would be used, it would be trivial to balance out all of those processes between the various CPUs. But since use cases vary from person to person, we can never have such knowledge. And often the final obstacle to improving the load balancer is simply that, regardless of the intelligence behind a given patch, there is simply no way to know if it's actually better than what was there previously. So, to be accepted, a load balancer patch might need to make a large, clearly noticeable improvement, when, ironically, more subtle and delicate changes might in fact be the better way to go.

New Random Number Handling

Andy Lutomirski submitted some patches to improve the Linux kernel's random number generation routines. First, he added a `getentropy()` function to provide a little entropy for use in generating a stream of random numbers. The idea is that entropy is itself a bit of randomness, taken from, for instance, the time delays between keyboard key presses. Then that number can be fed into a random number generator that will produce a stream of random numbers based on it. If you feed the same entropy in each time, you get the same stream of "random" numbers – not so random anymore. But if you have a good source of entropy, you can always have a fresh random number to start with, and therefore a truly unpredictable stream of random numbers.

However, as Andy pointed out, you don't always want this. Sometimes a bit of code wants random numbers, but not because they need to be cryptographically secure. Sometimes it just wants something, anything, so long as it is different than what came before. Andy's code would guarantee that it would provide a "best effort" at obtaining entropy, without actually requiring anything like true entropy.

The point of this is that the Linux kernel would normally wait for enough entropy to build up in the system, before allowing one of these entropy requests to return to the calling routine. And this is definitely still important for various cases. But in the cases where it's not, Andy's patches speed things up by not forcing the user code to wait for the build up of a suitable amount of entropy.

Andy added reassuringly, "This series should not break any existing programs. `/dev/urandom` is unchanged. `/dev/random` will still block just after booting, but it will block less than it used to. `getentropy()` with existing flags will return output that is, for practical purposes, just as strong as before."

Theodore Y. Ts'o remarked that this was actually a really big change. He felt that the timing was not right in the development cycle for a patch "of this magnitude." He added, "The reason for this is because at the moment, there are some PCI compliance labs who believe that the 'true randomness' of `/dev/random` is necessary for PCI compliance and so they mandate the use of `/dev/random` over `/dev/urandom`'s 'cryptographic randomness' for that reason. A lot of things which are thought to be needed for PCI compliance that are about as useful as eye of newt and toe of frog, but nothing says that PCI compliance (and enterprise customer requirements :-) have to make sense."

Ted added, "It may be that what we might need to really support people (or stupid compliance labs) who have a fetish for 'true randomness' [is] to get a better interface for hardware random number generators than `/dev/hwrng`. Specifically, one which allows for a more sane way of selecting which hardware random number generator to use if there are multiple available, and also one where we mix in some CRNG as a whitening step just [in] case the hardware number generator is busted in some way. (And to fix the issue that at the moment, if someone evil fakes up a USB device with the USB manufacturer and minor device number for a ChosKey device that generates a insecure sequence, it will still get blindly trusted by the kernel without any kind of authentication of said hardware device.)"

Ted's idea was to find a way to hook `/dev/random` into any available hardware random number generator to satisfy those users who needed truly random numbers.

But Andy thought this might not be a kernel issue at all. He saw no reason why the PCI folks couldn't be satisfied by a userspace source of randomness. He remarked, "it should be straightforward to write a little CUSE program that grabs bytes from RDSEED or RDRAND, TPM, ChaosKey (if enabled, with a usb slot selected!), and whatever other sources are requested and, configurable to satisfy whoever actually cares, mixes some or all with a FIPS-compliant, provably-indistinguishable-from-random, definitely not Dual-EC mixer, and spits out the result. And filters it and checks all the sources for credibility, and generally does whatever the user actually needs. And the really over-the-top auditors can symlink it to `/dev/random`."

Pavel Machek also replied to Andy's original post, asking for some better justification of the patches than Andy had given. And Andy explained, "The random code is extremely security sensitive, and it's made considerably more complicated by the need to support the blocking semantics for `/dev/random`. My primary argument is that there is no real reason for the kernel to continue to support it."

There was no further discussion, but Ted was right that Andy's patch would be a big change – not necessarily to the behavior of the kernel at all, but just to the resources offered by the kernel to user code. Depending on how much time Linus Torvalds wanted to give users to adapt their code to this new randomness situation, Andy's patches would have to be timed carefully, to appear early in the development cycle leading to the next official kernel release. ■■■

18 YEARS OF LINUX!
Issues 1-214
Searchable Linux Archive: 19,000 Pages of Practical Know-How
All the Tricks
All the Hacks
All the Apps
ISSUE 215
OCT 2018
LINUX MAGAZINE
DVD ROM
THE COMPLETE LINUX MAGAZINE ARCHIVE

Using voice-controlled interfaces via Amazon Alexa

Listening to the Word

Want to add voice activation to your IoT environment? Create an Alexa skill. *By Jens-Christoph Brendel and Martin Mohr*

If you want to control your own home automation environment with Amazon Alexa using natural language, you have two options. Either resort to a prebuilt Alexa skill, as offered by the vendors of some automation components, or write a skill of your own.

If you can find a prebuilt skill that performs the task you want to automate, you can accomplish the automation with just a few short steps; however, the possibilities are limited to the set of options that have already been provided by third-party programmers. If you want to reach other devices – or even if you just want to execute a series of actions that don't fall easily within Alexa's existing skill set, you need to write the skill yourself.

This article shows how to build the front end of your Alexa automation by getting Alexa to communicate with a Raspberry Pi. Once you establish the link to the RaspPi device, you can train the Pi to perform any number of basic functions on your IoT home network.

Before you jump out and start from scratch, however, it pays to take a careful look at the prebuilt options. Alexa supports a number of prebuilt skills that provide easy access to existing automation systems.

Prebuilt Skills

Alexa's built-in skills are the fastest and easiest way to automate – if you can find a skill that does what you need. Many prebuilt skills tie in with existing automation systems and IoT environments. (See the box entitled "Alexa in Harmony.")

Prebuilt Alexa Skills are also available for wireless socket outlets, but only for

Alexa in Harmony

One example of a ready-made skill is Logitech's Harmony universal remote control with its hub. The hub is a central transmitter that sends signals via infrared, Bluetooth, or WLAN to the devices that the user wants to operate. The remote control you hold in your hand no longer talks directly to the TV, stereo, or video player, but to the hub, which in turn talks to the devices. Thanks to an Alexa skill, this hub can now be operated by voice, which gives you the ability to talk to a wide and diverse range of home electronics devices.

The basic switch-on and switch-off commands can be combined to create actions. For example, if – as a TV viewer – you use a sound bar for better sound quality or surround sound, you can always switch it on and off along with the TV set by linking the virtual on/off switches of both devices in a single sequence (Figure 1).

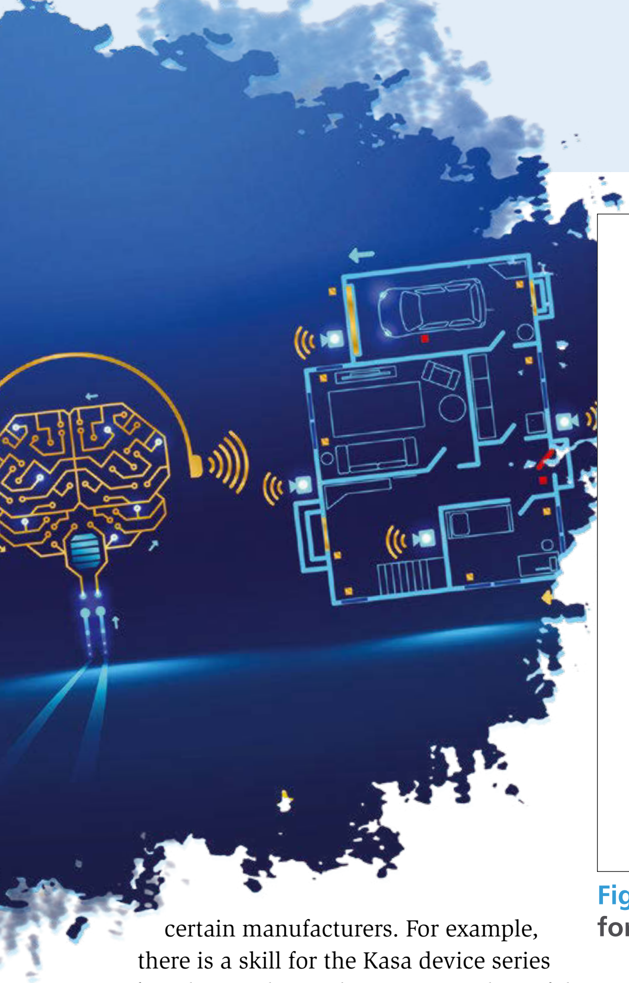
These actions can then be triggered again using an Alexa voice command; in other words, a terse "Alexa, good night" is enough to switch off the TV and sound bar, dim the lights, lower the blinds, and lock the apartment door.

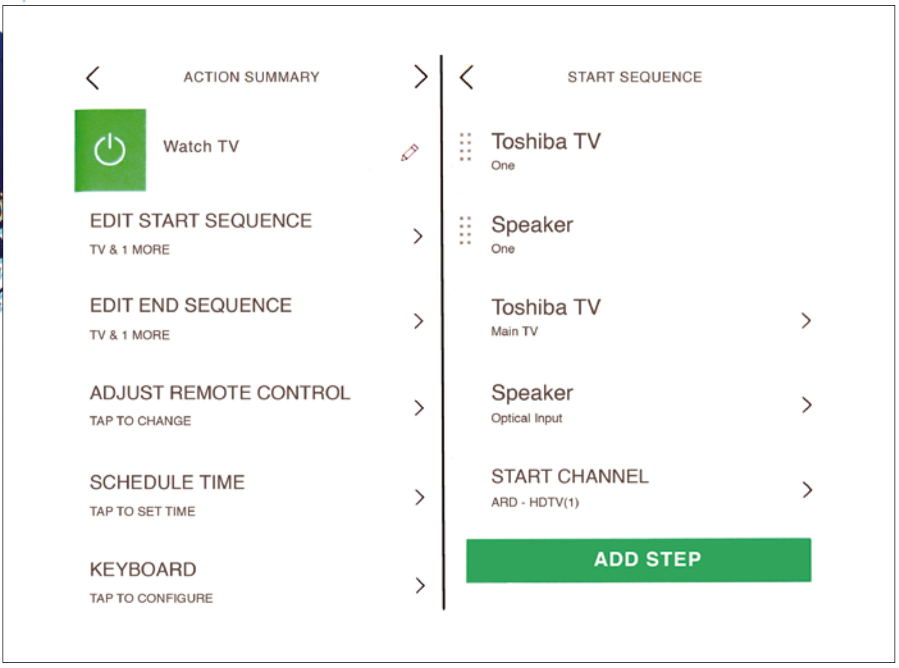
What do you need to do to achieve this? The first step is to define the actions in the Harmony app on your smartphone or tablet. The app is available for Android and iOS. Harmony controls over 270,000 different devices from most major manufacturers, including Bose, Philips, Denon, Sonos, Hue, and Deutsche Telekom. The actions can also be assigned to buttons on the remote control, so that pressing a button triggers a whole cascade of commands. In this example, however, Alexa will trigger the actions.

You can obtain the *Harmony* skill from the Amazon Alexa App Store in the Alexa app on your smartphone or tablet. The skill sports a blue logo. Watch out! An older version of this skill with a red logo named *Harmony Second Hub* is no longer recommended because it forced you to say the words "with Harmony" with all commands.

After you download, you need to enable the skill and log in with the same credentials that you use for your Logitech account. The Alexa skill then automatically gathers information about the Harmony actions.

If you want, you can fine-tune the prestored wording for the voice commands or the device name, but this is not absolutely necessary. Voice commands like "Alexa, switch on the TV" or "Alexa, switch on the Xbox" will work – provided you previously defined a corresponding action in the Harmony app. You can say "Alexa, switch to channel 3" or "Alexa, turn up the volume by four increments" and so on.




ACTION SUMMARY

Watch TV

EDIT START SEQUENCE
TV & 1 MORE

EDIT END SEQUENCE
TV & 1 MORE

ADJUST REMOTE CONTROL
TAP TO CHANGE

SCHEDULE TIME
TAP TO SET TIME

KEYBOARD
TAP TO CONFIGURE

START SEQUENCE

Toshiba TV
One

Speaker
One

Toshiba TV
Main TV

Speaker
Optical Input

START CHANNEL
ARD - HDTV(1)

ADD STEP

Figure 1: Two screens of the Harmony app to define a start sequence for the TV set and sound bar.

certain manufacturers. For example, there is a skill for the Kasa device series (wireless socket outlets, cameras, lamps) by TP-Link. You first need to set up the socket outlet in a TP-Link Kasa app and assign a name. You can then select the device and add it to the Alexa app. If you named the socket outlet "reading lamp" in the first step because it operates the standard lamp next to your comfortable armchair, you can then turn on the lights by saying "Alexa, reading lamp on."

The setup for a prebuilt skill is usually simple and convenient. If you are happy with the basic functions of popular devices, you will not be motivated to become a skilled programmer yourself. At times, however, you might want to combine several actions or use functions that are not included in the repertoire of ready-made skills. In this case, you will have to program a skill yourself.

Keep in mind that, if you don't trust Alexa when it comes to data protection, self-programming will not help much. Whether you use a prebuilt skill or program the skill yourself, everything you tell Alexa is routed through the Amazon server and stored there.

DIY Alexa Skill

The skill programmer faces two quite different challenges: First, you need to ensure that the computer that will execute the actions, a Raspberry Pi in this example, receives and interprets the voice command from Alexa and learns what to do. In addition to the RaspPi, one of Amazon's smart Echo series speakers is also needed to receive the spoken instructions and forward them to the servers.

Secondly, you need to program the action that you want carried out. The example in this article only looks at step 1. Communication between Alexa and your RaspPi is the foundation; from there, you can program your Pi to perform any task that makes sense for your network. In this example, the Raspberry Pi will switch on one of its eight LEDs to indicate that it has received and understood an instruction (Figure 2).

A first generation Raspberry Pi is powerful enough for the experiment; it runs the latest version of Raspbian "Stretch" Lite. The Lite version of the operating system does without a graphical user interface and therefore copes particularly well despite the frugal hardware resources. A GUI is not necessary for this project anyway. The Lite version of Raspbian lacks some libraries and tools that need to be installed before you

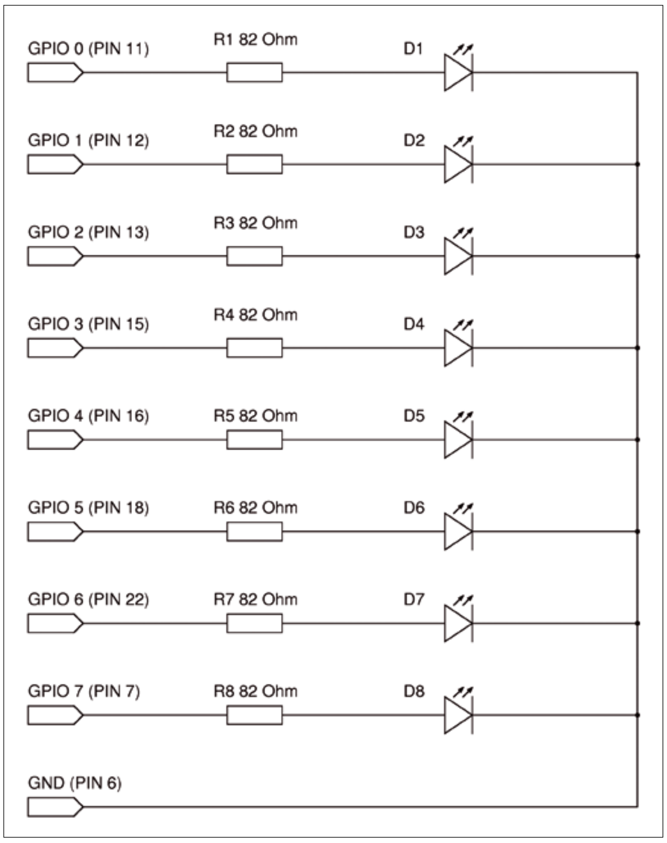


Figure 2: The schematic for the eight LEDs.

can install the Flask web framework (written in Python), which is required for this example.

These installations are handled by the commands in the first four lines of Listing 1, including the installation of Flask via Pip, the Python module management tool. The process takes some time – this might be a good time for you to take a coffee break.

Tunnel Builder

The Raspberry Pi still lacks a tool for building a secure tunnel between the RaspPi and the Amazon servers. This is where the Ngrok tunneling service comes in. Ngrok is very easy to install and configure. With the free version of Ngrok, however, the URL of the tool changes after each restart. If this is too much of an annoyance for you, you have to bite the bullet and go for a commercial version. The basic cost is around $5 per month.

To install the program, download the Linux ARM version from the Ngrok website and transfer the zip archive to the Raspberry Pi. Then unpack the archive and call Ngrok directly in a terminal (Listing 1, Lines 5 and 6). The options of the command in Line 6 tell Ngrok to forward port 5000 from localhost and listen for HTTP requests there. The output should be similar to Figure 3.

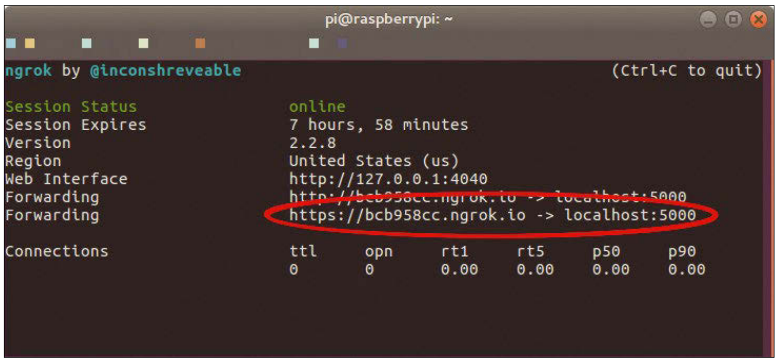
Each time a connection is established, Ngrok generates new random URLs. You need to make a note of the address marked red in the figure: you will need it later on as a communication endpoint in the Alexa skills.

Control Program

Since the running Ngrok server is now blocking the current terminal, it makes sense to open a sec-

Listing 1: Setting up Flask and Ngrok

```
01 $ sudo apt update
02 $ sudo apt upgrade
03 $ sudo apt install python2.7-dev python-dev python-pip wiringpi
04 $ sudo pip install flask-ask
05 $ unzip /home/pi/ngrok-stable-linux-arm.zip
06 $ ./ngrok http 5000
```



```
pi@raspberrypi: ~
ngrok by @inconshreveable                                      (Ctrl+C to quit)

Session Status                online
Session Expires               7 hours, 58 minutes
Version                       2.2.8
Region                        United States (us)
Web Interface                 http://127.0.0.1:4040
Forwarding                    http://bcb958cc.ngrok.io -> localhost:5000
Forwarding                    https://bcb958cc.ngrok.io -> localhost:5000

Connections                   ttl     opn     rt1     rt5     p50     p90
                              0       0       0.00    0.00    0.00    0.00
```

Figure 3: The output from Ngrok. The HTTPS URL is required later on.

Listing 2: house.py

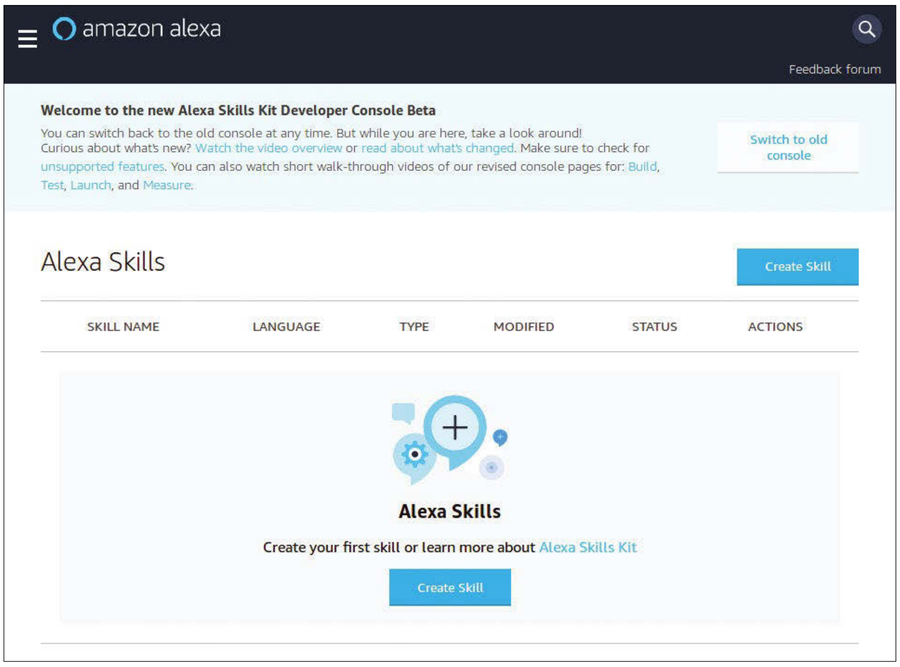
```
# house.py

import logging
import time
import RPi.GPIO as GPIO
from flask import Flask, render_template
from flask_ask import Ask, statement, question, session

GPIO.setmode(GPIO.BCM)
app = Flask(__name__)
ask = Ask(app, "/")
logging.getLogger("flask_ask").setLevel(logging.DEBUG)

@ask.launch
def greeting():
GPIO.setup(17, GPIO.OUT) # GPIO 0
GPIO.setup(18, GPIO.OUT) # GPIO 1
GPIO.setup(27, GPIO.OUT) # GPIO 2
greeting = render_template('greeting')
return question(greeting)

@ask.intent("TVIntent",mapping={'status': 'status'})
def TV(status):
print 'test=>{}'.format (status)
if status == "on":
print 'ON'
GPIO.output(17, GPIO.HIGH) # GPIO 0
if status == "off":
print 'OFF'
GPIO.output(17, GPIO.LOW) # GPIO 0
status = render_template('TV', status=status)
return question(status)

@ask.intent("LampIntent",mapping={'status': 'status'})
def lamp(status):
if status == "on": GPIO.output(18, GPIO.HIGH) # GPIO 1
if status == "off": GPIO.output(18, GPIO.LOW) # GPIO 1
status = render_template('lamp', status=status)
return question(status)

@ask.intent("SocketIntent")
def socket(status):
if status == "on": GPIO.output(27, GPIO.HIGH) # GPIO 2
if status == "off": GPIO.output(27, GPIO.LOW) # GPIO 2
status = render_template('socket', status=status)
return question(status)

if __name__ == '__main__':
app.run(debug=True)
```

Listing 3: templates.yaml

```
greeting: Gooday
TV: TV is {{status}}
light: Light is {{status}}
socket: Socket is {{status}}
```

ond terminal window. In it, create a text file named `house.py` with the contents from Listing 2. This Python program calls the Alexa skill later on. The program provides a server that controls three devices: a TV set, a lamp, and a socket outlet.

For each of the three devices, the program has what is known as an intent, which controls exactly this one function. The corresponding decorators `@ask.intent()` define the functions that the intent then executes as an action when called.


amazon alexa
Feedback forum

Welcome to the new Alexa Skills Kit Developer Console Beta

You can switch back to the old console at any time. But while you are here, take a look around! Curious about what's new? Watch the video overview or read about what's changed. Make sure to check for unsupported features. You can also watch short walk-through videos of our revised console pages for: Build, Test, Launch, and Measure.

Switch to old console

Alexa Skills

Create Skill

SKILL NAME	LANGUAGE	TYPE	MODIFIED	STATUS	ACTIONS

Alexa Skills

Create your first skill or learn more about Alexa Skills Kit

Create Skill

Figure 4: The (still empty) list of Alexa skills on an Amazon Web Services account.

Say Hello!

The `@ask.launch()` decorator displays a friendly greeting text. In this example, the Australian "Gooday" is used, a greeting that would be appropriate at any time of day.

In the functions, the commands necessary for controlling the GPIOs are executed. The Flask framework abstracts the complexity behind a skill very well. Further data can be found in the `templates.yaml` resource file. The file must reside in the same directory as the Python program that communicates with the skill. This file contains texts that the skill will use. This makes it easier to adjust the speech output if necessary, for example to change the greeting or to enter other devices. The quite compact resource file for the example from this article is shown in Listing 3.

The shell `python house.py` command starts the Python server. Ngrok and the Python program simply keep running after that. Of course, you cannot close any of the terminal windows afterwards. You need to make sure that Ngrok is running when the Python server enters the scene, otherwise the connection between the Echo skill and the server program cannot be established.

Setting up an Alexa Skill

For the next step when programming an Alexa skill, the Echo device you use must be activated. You first need an Amazon Web Services (AWS) account, which can be obtained free of charge on the AWS homepage. All you need is a credit card and a mobile phone connection.

You can now create a new Alexa skill via the AWS account. The easiest way to get there is via the list of current Alexa skills, which is empty for a new user account (Figure 4). There the user clicks on the *Create Skill* button to create a new skill.

The skill should then be given a descriptive name and a desired language. A mouse click on the *Next* button takes you directly to the next step of choosing the Custom model: it offers the highest degree of flexibility.

Another click on the button *Create Skill* closes the wizard. Now the homepage belonging to the new skill appears (Figure 5). Here you can set further parameters of the skill. The video under "How to get started" at the top of the page helps you to get started with the subtleties of skill creation.

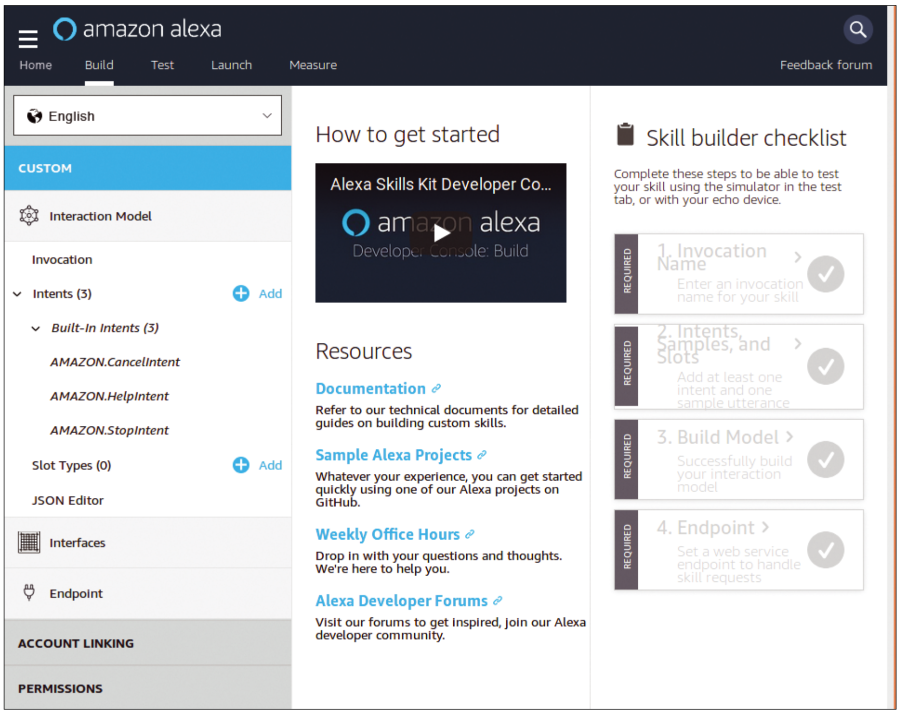
Skills in Detail

To understand more exactly how an Alexa skill is composed, it is worth taking a look at its individual components. The skill describes an interaction model that summarizes all behaviors and components. The interaction model itself consists of a number of components.

The invocation is the name of the Alexa skill and serves as a keyword for its activation. You'll need to follow some rules when assigning names – the Alexa website explains them in detail.

An intent describes an action that the user wants to perform within the skill. The individual intents require at least one defined keyword (sample utterance), which starts the action. If required, several sample utterances can be stored for each intent – for example, the variants "affirmative," "exactly," and "yeah" for the simple "yes." The more meaningful the utterances for an intent, the more reliable the skill becomes.

The slot types are self-defined data types that are used within the skills. The idea behind the slot types is similar to the enumeration data types known from almost all programming languages. For example, when asking for a car brand, the slot type could be *car brand* and contain values such as *BMW*, *Audi*, or *Ford*.


amazon alexa
Home Build Test Launch Measure Feedback forum

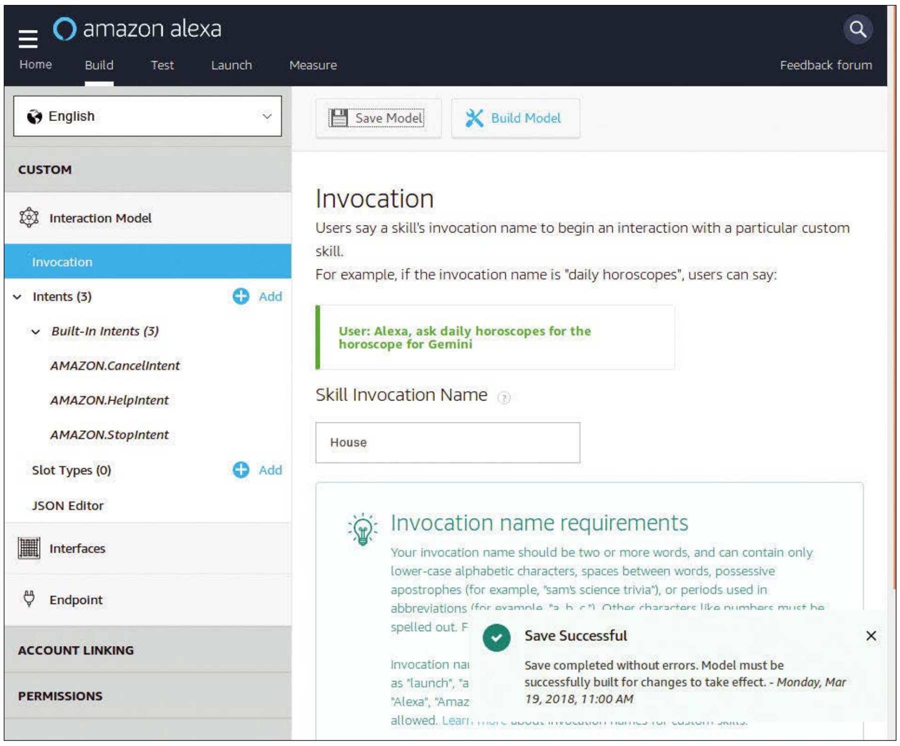
English
CUSTOM
Interaction Model
Invocation
Intents (3) Add
Built-In Intents (3)
AMAZON.CancelIntent
AMAZON.HelpIntent
AMAZON.StopIntent
Slot Types (0) Add
JSON Editor
Interfaces
Endpoint
ACCOUNT LINKING
PERMISSIONS

How to get started
Alexa Skills Kit Developer Co...
amazon alexa
Developer Console: Build

Resources
Documentation
Refer to our technical documents for detailed guides on building custom skills.
Sample Alexa Projects
Whatever your experience, you can get started quickly using one of our Alexa projects on GitHub.
Weekly Office Hours
Drop in with your questions and thoughts. We're here to help you.
Alexa Developer Forums
Visit our forums to get inspired, join our Alexa developer community.

Skill builder checklist
Complete these steps to be able to test your skill using the simulator in the test tab, or with your echo device.
REQUIRED 1. Invocation Name – Enter an invocation name for your skill
REQUIRED 2. Intents, Samples, and Slots – Add at least one intent and one sample utterance
REQUIRED 3. Build Model – Successfully build your interaction model
REQUIRED 4. Endpoint – Set a web service endpoint to handle skill requests

Figure 5: The web page for the newly created skill.


amazon alexa
Home Build Test Launch Measure Feedback forum

English
CUSTOM
Interaction Model
Invocation
Intents (3) Add
Built-In Intents (3)
AMAZON.CancelIntent
AMAZON.HelpIntent
AMAZON.StopIntent
Slot Types (0) Add
JSON Editor
Interfaces
Endpoint
ACCOUNT LINKING
PERMISSIONS

Save Model Build Model

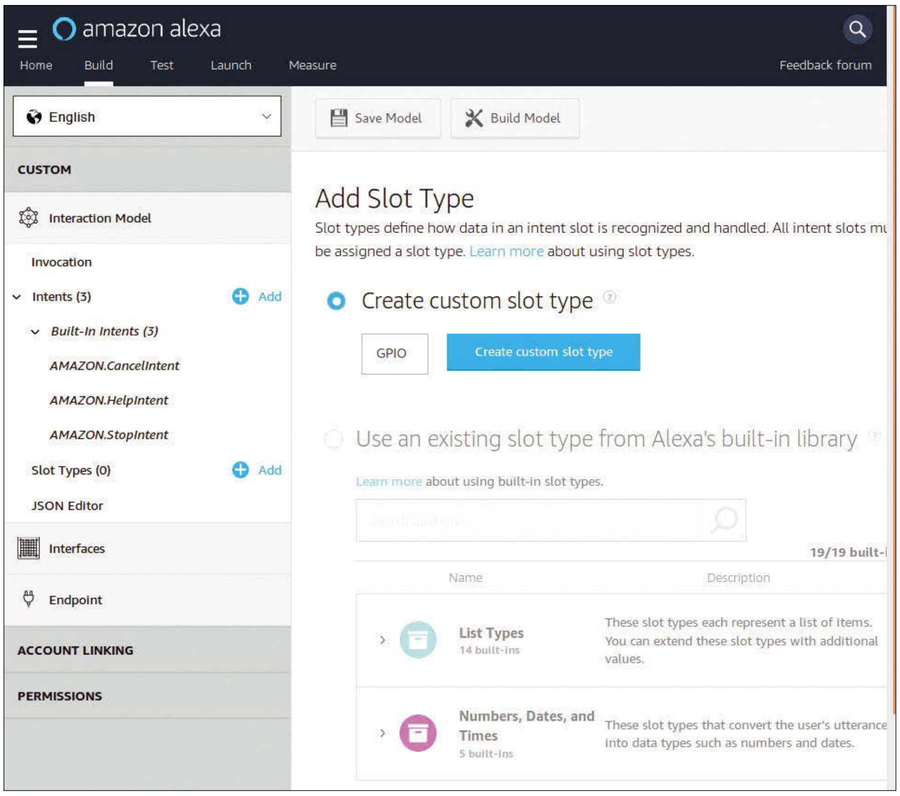
Invocation
Users say a skill's invocation name to begin an interaction with a particular custom skill.
For example, if the invocation name is "daily horoscopes", users can say:
User: Alexa, ask daily horoscopes for the horoscope for Gemini

Skill Invocation Name
House

Invocation name requirements
Your invocation name should be two or more words, and can contain only lower-case alphabetic characters, spaces between words, possessive apostrophes (for example, "sam's science trivia"), or periods used in abbreviations (for example, "a. b. c."). Other characters like numbers must be spelled out. F
Invocation na
as "launch", "a
"Alexa", "Amaz
allowed. Lear

Save Successful
Save completed without errors. Model must be successfully built for changes to take effect. - *Monday, Mar 19, 2018, 11:00 AM*

Figure 6: Finally, you define the activation keyword for the skill.


amazon alexa
Home Build Test Launch Measure Feedback forum
English
Save Model Build Model
CUSTOM
Interaction Model
Invocation
Intents (3) Add
Built-In Intents (3)
AMAZON.CancelIntent
AMAZON.HelpIntent
AMAZON.StopIntent
Slot Types (0) Add
JSON Editor
Interfaces
Endpoint
ACCOUNT LINKING
PERMISSIONS

Add Slot Type
Slot types define how data in an intent slot is recognized and handled. All intent slots mu
be assigned a slot type. Learn more about using slot types.
Create custom slot type
GPIO Create custom slot type
Use an existing slot type from Alexa's built-in library
Learn more about using built-in slot types.
19/19 built-i

Name	Description
List Types 14 built-ins	These slot types each represent a list of items. You can extend these slot types with additional values.
Numbers, Dates, and Times 5 built-ins	These slot types that convert the user's utterance into data types such as numbers and dates.

Figure 7: **Create a slot type via the interface.**

With the JSON editor, a simple text editor, the user edits the individual components of the skill directly in the interface. Although this task requires appropriate know-how, it makes it easier to keep track of complex skills.

In the *Interfaces* section, the user integrates multimedia content directly into an Alexa skill. Currently you will find three interfaces: Audio Stream, Video Stream, and Display Interface.

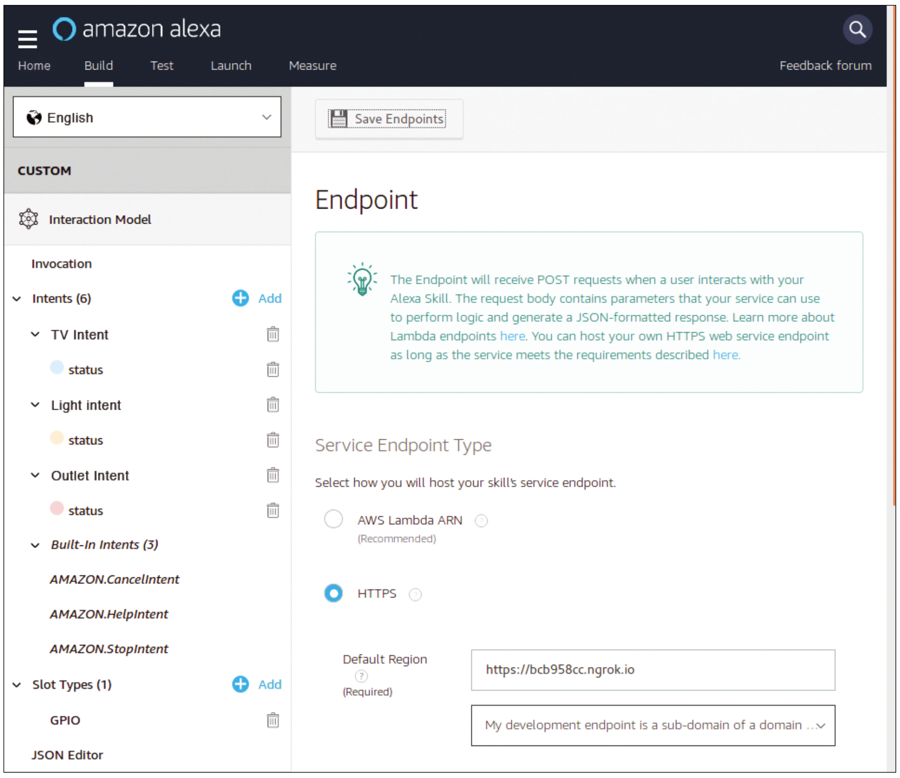
The endpoints determine which services the Alexa skill should communicate with on the Internet. In the present case, Raspberry Pi, which is linked to the skill via the Ngrok service, acts as the endpoint.

The sample skill needs to control three devices: a TV set, the light, and a socket outlet. To implement this, you create an intent for each of the three devices. Each intent supports the commands that name the corresponding device and transmit its status. The *House* command acts as the keyword. Define it below *Invocation* and then press the *Save Model* button (Figure 6). From experience, I can only advise readers to save more often than you might think necessary, rather than wondering later on why the skill does not work as it should.

Slots and Intents

A slot type named *GPIO* transmits the status of the individual devices. *GPIO* can assume one of two values: *on* and *off*. The slot type can already be created at this point. Press *Add* to the right of the *Slot Types* entry in the sidebar.

When you get there, select a custom slot type and assign the name of *GPIO* (Figure 7). Then, on the following page, assign the *on* and *off* values. To save the values, remember to press *Save Model* at the top.


amazon alexa
Home Build Test Launch Measure Feedback forum
English
Save Endpoints
CUSTOM
Interaction Model
Invocation
Intents (6) Add
TV Intent
status
Light intent
status
Outlet Intent
status
Built-In Intents (3)
AMAZON.CancelIntent
AMAZON.HelpIntent
AMAZON.StopIntent
Slot Types (1) Add
GPIO
JSON Editor

Endpoint
The Endpoint will receive POST requests when a user interacts with your Alexa Skill. The request body contains parameters that your service can use to perform logic and generate a JSON-formatted response. Learn more about Lambda endpoints here. You can host your own HTTPS web service endpoint as long as the service meets the requirements described here.

Service Endpoint Type
Select how you will host your skill's service endpoint.
AWS Lambda ARN (Recommended)
[✓ HTTPS]
Default Region (Required): https://bcb958cc.ngrok.io
My development endpoint is a sub-domain of a domain ...

Figure 8: **Creating an endpoint.**

On top of this, you now create the intents. You could do this manually by clicking your way through the individual dialog boxes. However, to reach this goal far faster, simply copy the content from Listing 4 to the interaction model using the JSON editor.

Make sure that the intents in the interaction model use the same names as in the Python program, otherwise you can expect trouble later on when you try to execute the skill.

In addition, the skill must be saved after each change to the interaction model and then rebuilt by clicking on *Build Model*. If you forget this step, nothing happens and you keep the old model.

Now only the communication between the skill and the Python program is missing. Click on *Endpoint* and select the *HTTPS* option. In the *Default Region* field, enter the URL currently used by Ngrok. In the drop-down box below, also enable the option labeled *My development endpoint is a subdomain of a domain that has a wildcard certificate from* *a certificate authority* (Figure 8). A click on *Save Endpoints* in the dialog header saves the changes.

After all configuration work has been completed, you need to build the skill for the first time. The *Build Model* switch appears on all pages where the model can be changed. Click on *Invocation* and then on *Build Model*.

For an initial test, change to the tab labeled with the name of the skill. Test mode can be enabled using the slider below the tab bar. Then enter the activation keyword, *House*, in the text box. The skill should then immediately welcome you in Australian. The connected devices can then be switched. The LEDs on the Raspberry Pi's GPIO prove that the skill works.

Controlling this with Amazon Echo should work the same way: The *Start house* voice command loads the skill, then the *TV on* command switches on the TV set.

Conclusion

Alexa provides an easy path for adding voice activation to your Raspberry Pi automations, but you'll need to set up a tunnel and do a little basic programming. This article offers a simple example for how to get Alexa talking to your RaspPi. Once you have established a communication channel, the possibilities for your voice-activated Raspberry Pi are endless. ■■■

Listing 4: Intents

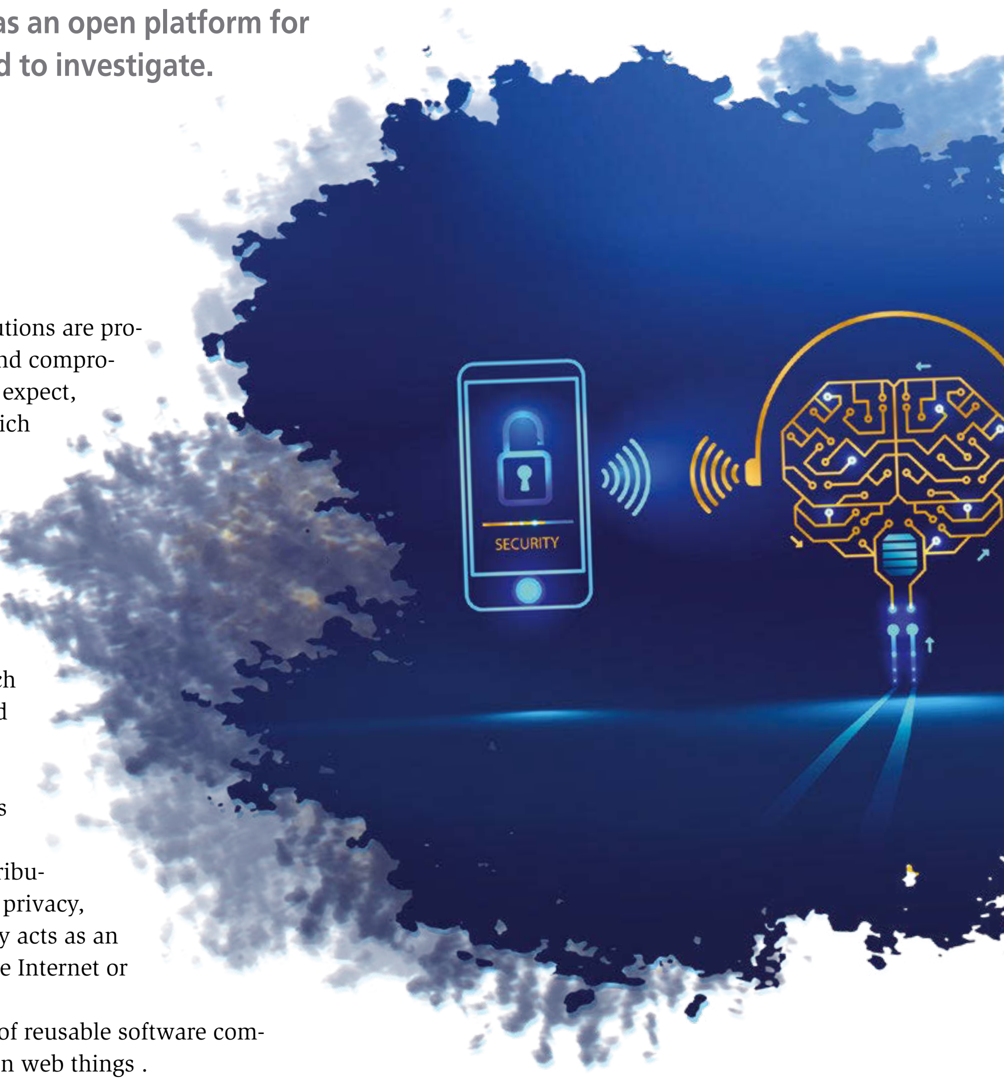
```
# intents.json
{
 "languageModel": {
     "invocationName": "house",
     "intents": [
       {
         "name": "AMAZON.CancelIntent",
         "slots": [],
         "samples": []
       },
       {
         "name": "AMAZON.HelpIntent",
         "slots": [],
         "samples": []
       },
       {
         "name": "AMAZON.StopIntent",
         "slots": [],
         "samples": []
       },
       {
         "name": "TVIntent",
         "slots": [
           {
             "name": "status",
             "type": "GPIO"
           }
         ],
         "samples": [
           "TV {status}",
           "tv {status}"
         ]
       },
       {
         "name": "LampIntent",
         "slots": [
           {
             "name": "status",
             "type": "GPIO"
           }
         ],
         "samples": [
           "lamp {status}"
         ]
       },
       {
         "name": "SocketIntent",
         "slots": [
           {
             "name": "status",
             "type": "GPIO"
           }
         ],
         "samples": [
           "socket {status}"
         ]
       }
     ],
     "types": [
       {
         "name": "GPIO",
         "values": [
           {
             "id": "",
             "name": {
               "value": "aus",
               "synonyms": []
             }
           },
           {
             "id": "",
             "name": {
               "value": "on",
               "synonyms": []
             }
           }
         ]
       }
     ]
   }
 }
```

Linux lab: Mozilla's WebThings Gateway

Crash Landing

The smart home is gaining momentum, and Mozilla joins the fray. Mozilla WebThings is billed as an open platform for managing IoT devices. We decided to investigate.

By Erik Bärwaldt



Many Internet of Things (IoT) solutions are proprietary tools that limit choice and compromise user privacy. As one might expect, the open source community, which has always prized freedom and openness, has been hard at work on solutions that would avoid the many problems associated with vendor lock-in.

The Mozilla project recently added a promising new technology to the IoT mix. Mozilla's WebThings [1] is an implementation of the Web of Things architecture, which attempts to build the IoT around proven and well known Internet technologies, such as REST, HTTP, and JSON.

According to Mozilla's website, WebThings consists of two primary components:

- WebThings Gateway [2] – a software distribution for smart home gateways focused on privacy, security, and interoperability. The gateway acts as an interface between the IoT network and the Internet or local network.
- WebThings Framework [3] – a collection of reusable software components to help developers build their own web things .

WebThings, which is open source and freely available on GitHub [4] supports a number of home automation protocols, including Zigbee [5], which has been under development for more than 15 years, and Z-Wave [6].

An open platform for IoT with multiple protocol support is a very promising development – but does WebThings work now with real-world IoT devices? We decided to find out.

Hardware

WebThings Gateway runs on a conventional Raspberry Pi (models 1 to 4) with a modified Raspbian operating system. Mozilla also offers an image for the Turris Omnia WLAN router [7], and the WebThings developers are working on a version for OpenWRT-based devices. Users control devices with the Things graphical user interface.

To enable communication between the computers on the network and Zigbee or Z-Wave IoT devices, you'll also need to plug a suitable gateway device into the Raspberry Pi. For our tests, we used the ConBee II Zigbee [8] USB stick. UZB, a Z-Wave USB stick, is also available [9].

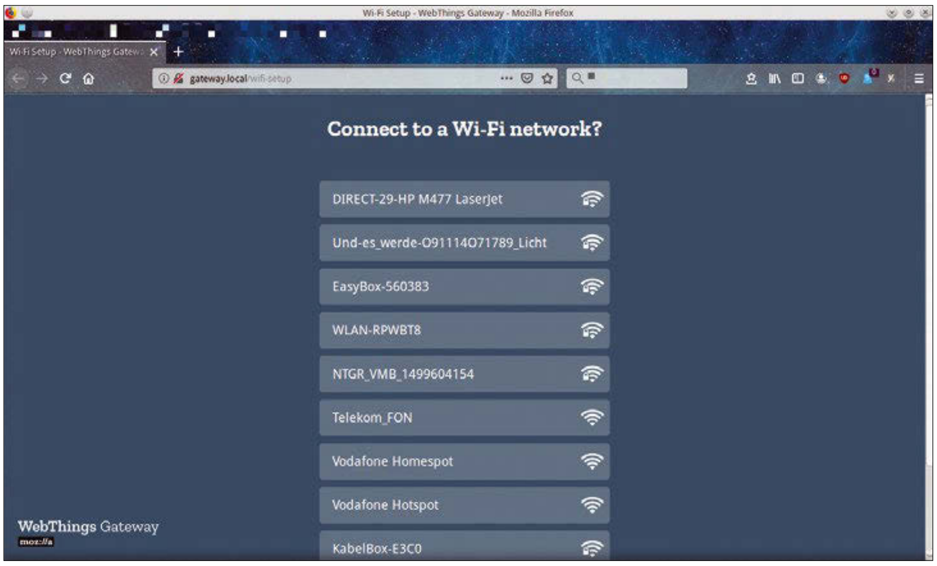


Figure 1: **WLAN detection already works reliably with Mozilla's WebThings Gateway.**

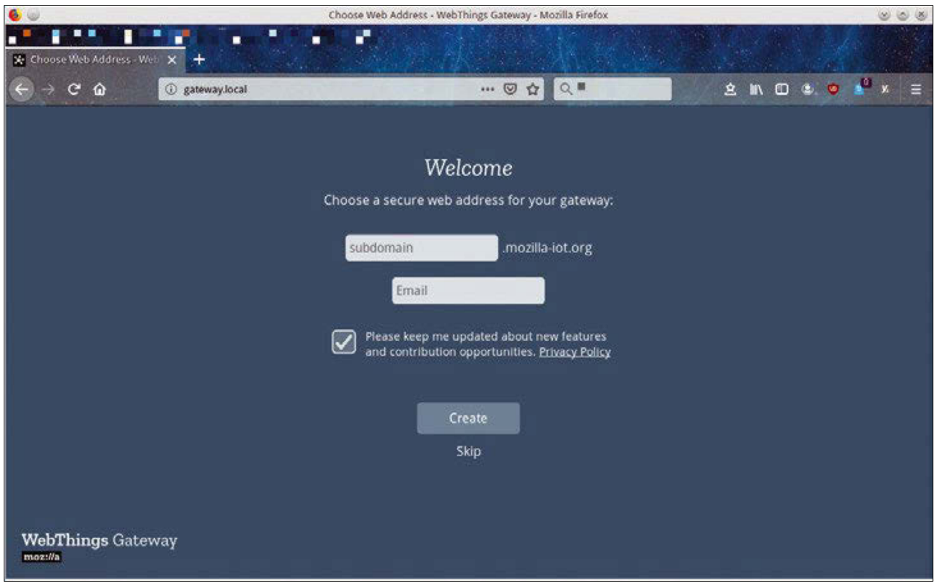

Choose Web Address - WebThings Gateway - Mozilla Firefox
gateway.local
Welcome
Choose a secure web address for your gateway:
subdomain
.mozilla-iot.org
Email
Please keep me updated about new features and contribution opportunities. Privacy Policy
Create
Skip
WebThings Gateway
mozilla

Figure 2: **Thanks to a separate subdomain, WebThings allows access via the Internet.**

First Contact

We used a Raspberry Pi 4B as our gateway computer. As end devices, we used several Zigbee-capable lamps by manufacturers Aurora, Enlite, Müller-Licht, and Philips. We also added a switchable OSRAM socket to the Zigbee network.

To build the WebThings Gateway system for the Raspberry Pi, download the image from the project website and unpack the archive. Then transfer the resulting image to a MicroSD card, which acts as the boot medium. (See the instructions on the Mozilla WebThings website [10].) Then plug the ConBee II stick into the Raspberry Pi and start the system.

Next set up a WLAN named *WebThings Gateway nnnn*, where *nnnn* is a random string of characters and digits. Users will connect to this WLAN hotspot on other computers. Open a web browser and type *http://gateway.local* in the address bar; this opens a list of available wireless networks. Select your WLAN from the list and associate your Rasp Pi with the desired WLAN by entering the WPA2 key (Figure 1).

Then log in to this WLAN on your computer and call the *http://gateway.local* URL once again. In the browser, Mozilla's gateway creates a subdomain that allows access to the gateway from the Internet. The keys for secure access to the gateway using the HTTPS protocol are generated in the background (Figure 2). You may have to forward the gateway IP address and port on the router.

In the next step, create a new user account in the browser. Enter the desired name, password, and email address. You are then taken to an almost empty screen that prompts you to search for new equipment. To start the search, press the plus button bottom right.

The system now searches the local network for connected Zigbee devices and lists them in the browser window. Alternatively, you can use the *Add by URL* link to manually integrate smart devices on the WLAN via their IP addresses.

The system settings can be accessed by opening the menu hidden behind the hamburger icon top left in the browser window. In the `Settings` submenu, first change the system settings. Use the system settings to create additional users and integrate other devices, which Mozilla refers to as Things, into the system. In addition, you can modify the WLAN or the subdomain configuration (Figure 3).

Modules

To operate most devices, you will need to retroactively install add-ons. Right from the start, only the WebThings, Z-Wave, and Zigbee modules are active. The WebThings module is used to integrate smart end devices that can be addressed via WLAN into Mozilla's home automation software. You can install additional modules later to access terminal devices, communicate via Bluetooth, and more (Figure 4).

Clicking on the plus button in the bottom right corner of the add-ons list display takes you to a list of additional modules. Click on the + *Add* button to integrate the module with the system.

Protocols

The protocol function in the Settings menu lets you obtain useful information such as the energy consumption data.

The *Logs* item opens a new view. Select one of the devices integrated into the system for the protocol function

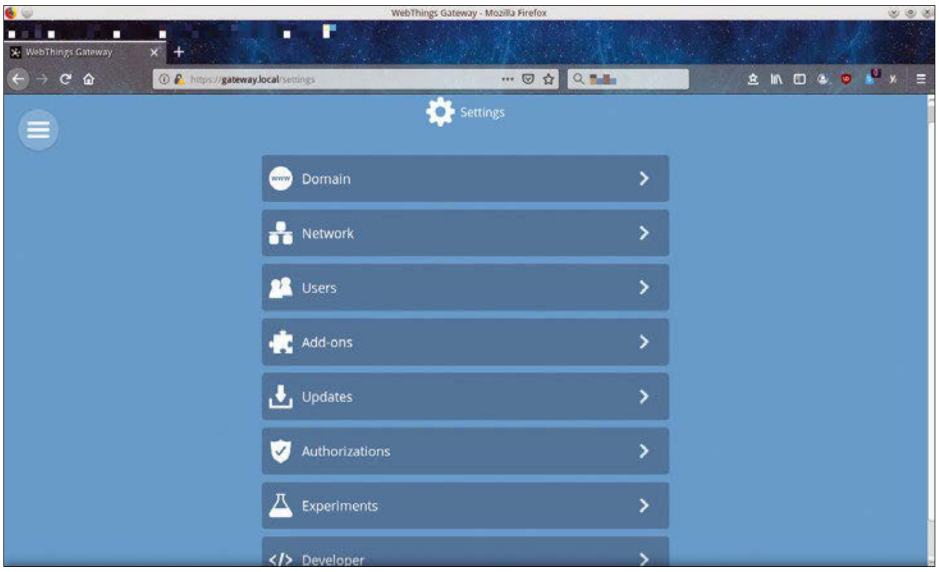


Figure 3: The WebThings Settings dialog offers several options.

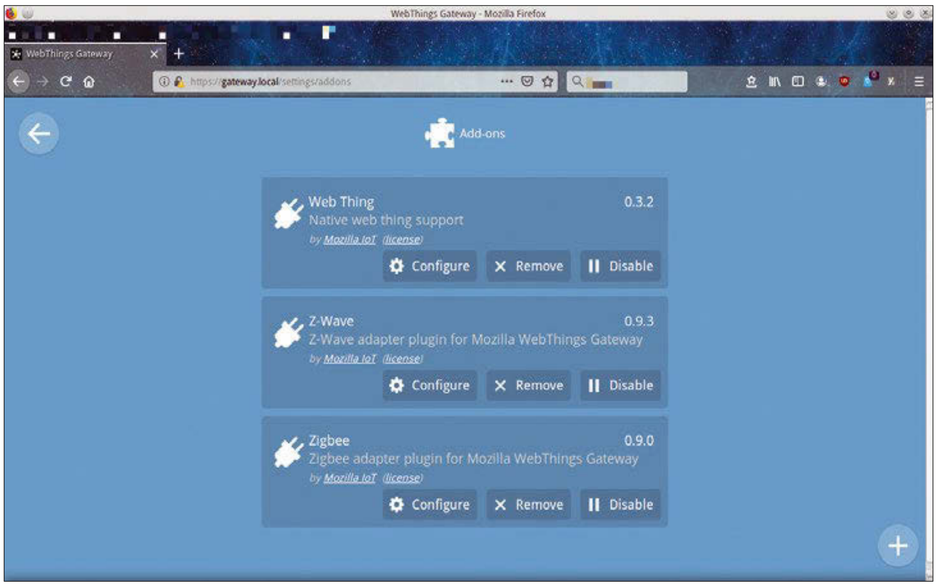


Figure 4: Add-on modules let you add additional devices to the network.

by clicking the plus button. In this dialog, you can also decide how long WebThings should store the logs. You can also create a floor plan of your home using the Floorplan dialog in the Settings menu. Enter the locations of all end devices in the floor plan. If you have a large number of integrated devices, the floor plan will definitely help you maintain an overview.

Defining Rules

Using the entry Rules in the Settings menu, you can define rules for controlling terminal devices. Define rules by dragging and dropping components from the device list. For instance, you can switch a terminal device on or off at a predefined time with just a few mouse clicks. To create new rules, press the plus button bottom right in the window.

Catastrophic

In our hands-on session, Mozilla's WebThings gave us a catastrophic first impression. Although the user interface is intuitive, and logically structured, there are only a few modules – often for devices that are no longer manufactured. Some of the devices do not even exist in Europe.

In our lab, we were unable to control even one of the half dozen end devices by manufacturers Aurora Lighting, Enlite, Müller-Licht, Philips, and OSRAM with WebThings. Mozilla's compatibility list does at least support Philips lamps and OSRAM socket adapters [11].

Hue lighting by Philips will only harmonize with WebThings if you reset the lamps to the factory settings. But to do this, you need a Philips control unit. Many popular vendors on today's home automation market are simply missing from the WebThings compatibility list.

Conclusions

WebThings has a promising approach to making the smart home accessible to users who do not want to transfer their data to the cloud. In addition, Mozilla's new project supports a number of different protocols and also controls WLAN and Bluetooth systems.

However, the project still suffers from a massive lack of development work in the field of add-ons for end devices. At this point, it is hardly possible to integrate today's devices, such as intelligent lamps or motion and temperature sensors, into WebThings. WebThings seems to be miles away from a solution for end users that is suitable for everyday use. ■■■

Info

[1] WebThings: *https://iot.mozilla.org*

[2] WebThings Gateway: *https://iot.mozilla.org/gateway*

[3] WebThings Framework: *https://iot.mozilla.org/framework/*

[4] WebThings on GitHub: *https://github.com/mozilla-iot/*

[5] Zigbee protocol: *https://en.wikipedia.org/wiki/ZigBee*

[6] Z-Wave protocol: *https://en.wikipedia.org/wiki/Z-Wave*

[7] Turris Omnia: *https://www.turris.cz/en/omnia/*

[8] ConBee II: *http://phoscon.de/en/conbee2*

[9] UZB: *https://z-wave.me/uzb/*

[10] Documentation: *https://iot.mozilla.org/docs/gateway-getting-started-guide.html*

[11] Compatibility list: *https://github.com/mozilla-iot/wiki/wiki/Supported-Hardware#adapters*

FHEM: Setup, practical use, and alternative interfaces

At Your Service

If you want to equip your home with smart technology, you will need to deal with a variety of providers and what are often incompatible standards. FHEM is a free integration platform that houses the building blocks under one roof and offers visually appealing interfaces. *By Jörg Hofmann*

FHEM [1] is an open source server for home automation. Your FHEM system can control Internet of Things (IoT) devices, such as lamps, thermostats, shutters, audio equipment, and all the other gadgets that populate a smart home. You can interact with FHEM through a smartphone front end or even over the web.

FHEM supports several different home automation protocols and has a number of different interfaces for the user to interact with directly or through a script. The modular architecture currently includes more than 430 modules.

The quality of an open platform depends on the commitment and skills of the community backing it. That commitment is evident in the case of FHEM. The FHEM Forum, the official contact point for enthusiasts, now has more than 20,000 members.

Although the number of available software modules was quite small just a few years ago, experienced users can now turn to countless modules to integrate not only Homematic (the classic wireless standard), but also widely used standards in the smart home sphere, such as Zigbee (used by Philips Hue and Ikea Trådfri, among others) or Z-Wave. Cable-based standards such as OneWire, DMX, or the KNX professional standard are also supported.

Using the official Tesla API, you can even read out your electric vehicle's status values, such as the current battery charge level. As soon as the battery is fully charged, FHEM can then trigger speech output to say, via a Sonos speaker, "Your Tesla is ready to go."

Setting Up

FHEM is based on the Perl programming language and is therefore genuinely lightweight when it comes to hardware requirements. In the early days, FHEM would even run on a FRITZ!Box router until the vendor, AVM Software, put a stop to this – officially for security reasons.

The popular, single-board Raspberry Pi is a worthy successor with more than sufficient resources. On a basic system, FHEM can be installed with just a few commands at the console. Start by importing the required repository key and adding the repository (Listing 1, lines 1 and 2).

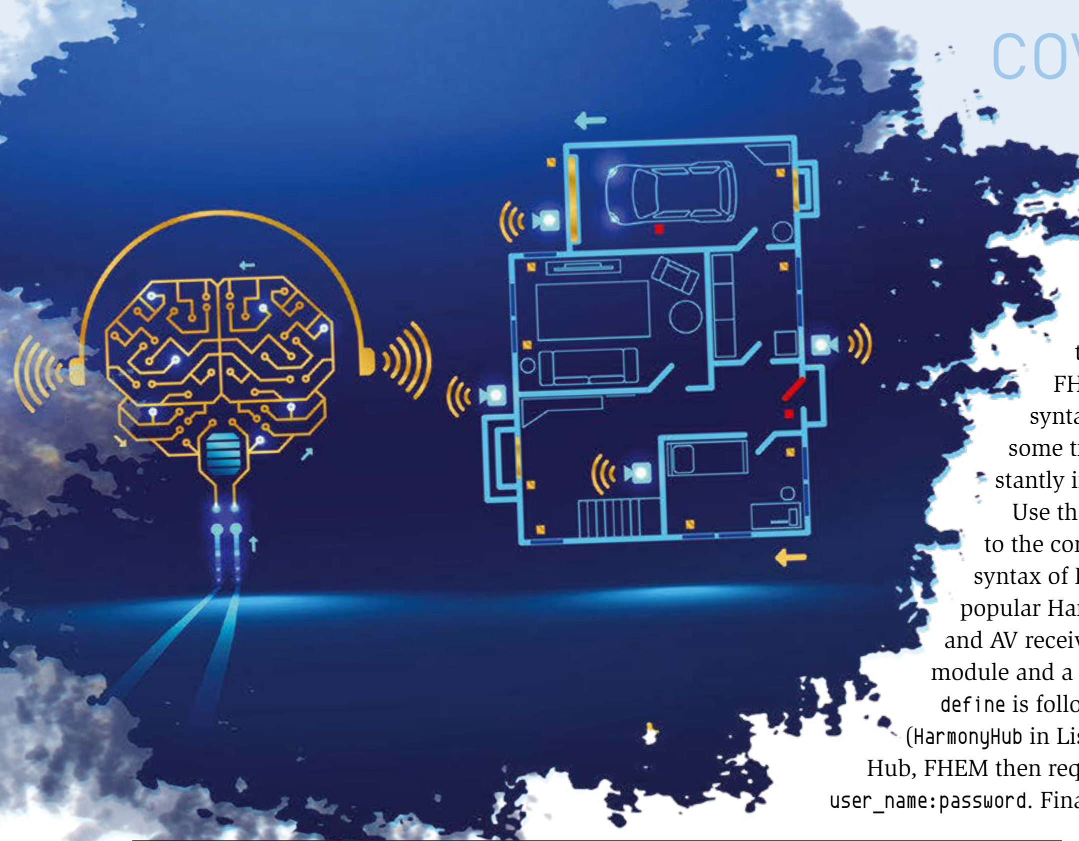
Then update the package sources and install FHEM (lines 3 and 4). Depending on the configuration, you may need to run the commands with `sudo` to have the necessary system permissions. Occasionally, the repository does not work properly, in which case, you will need to manually install Perl and FHEM [2].

First Steps

Once the installation is complete, you can access the FHEM web interface on port 8083 by typing `http://<RaspPi_IP_address>:8083` in a web browser. The FHEM interface (Figure 1) uses a simple, unpretentious, and functional design – don't bother looking for graphical components. This simple approach is an advantage for experienced users, but it does raise the bar for newcomers.

Listing 1: Installing FHEM

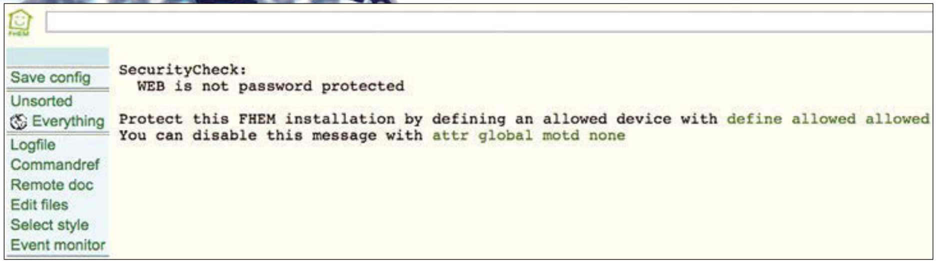
```
01 $ wget -qO - http://debian.fhem.de/archive.key | apt-key add -
02 $ echo "deb http://debian.fhem.de/nightly/ /" >> /etc/apt/sources.list
03 $ apt-get update
04 $ apt-get install fhem
```



The central element in the standard interface is the white bar at the top, where the user can enter commands. You can issue commands to configure and manage the system. FHEM even comes with a very useful syntax check that has been available for some time and that the community is constantly improving.

Use the `define` command to add a new device to the configuration. A command based on the syntax of Listing 2, for example, integrates the popular Harmony Hub, which can control your TV and AV receivers via a built-in infrared transmitter module and a Bluetooth transceiver.

`define` is followed by the desired FHEM device name (`HarmonyHub` in Listing 2). In the case of the Harmony Hub, FHEM then requires the login data in the form of `user_name:password`. Finally, enter the device's IP address and press the Enter key.

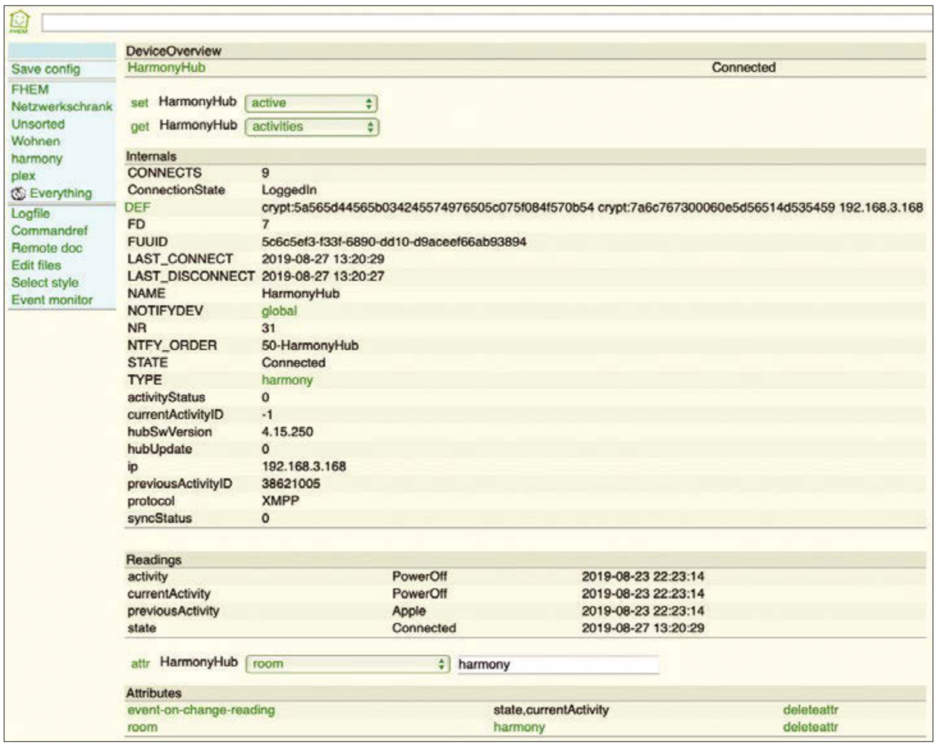


```
SecurityCheck:
  WEB is not password protected

Protect this FHEM installation by defining an allowed device with define allowed allowed
You can disable this message with attr global motd none
```

Figure 1: The simple FHEM web interface can be reached on the default port 8083.

Listing 2: Integrate Harmony Hub

```
define HarmonyHub harmony <I>User<I>:<I>Password<I> <I>IP_Address<I>
```



```
DeviceOverview
HarmonyHub                                                    Connected

set  HarmonyHub  [active      ]
get  HarmonyHub  [activities  ]

Internals
CONNECTS          9
ConnectionState   LoggedIn
DEF               crypt:5a565d44565b034245574976505c075f084f570b54 crypt:7a6c767300060e5d56514d535459 192.168.3.168
FD                7
FUUID             5c6c5ef3-f33f-6890-dd10-d9aceef66ab93894
LAST_CONNECT      2019-08-27 13:20:29
LAST_DISCONNECT   2019-08-27 13:20:27
NAME              HarmonyHub
NOTIFYDEV         global
NR                31
NTFY_ORDER        50-HarmonyHub
STATE             Connected
TYPE              harmony
activityStatus    0
currentActivityID -1
hubSwVersion      4.15.250
hubUpdate         0
ip                192.168.3.168
previousActivityID 38621005
protocol          XMPP
syncStatus        0

Readings
activity                  PowerOff         2019-08-23 22:23:14
currentActivity           PowerOff         2019-08-23 22:23:14
previousActivity          Apple            2019-08-23 22:23:14
state                     Connected        2019-08-27 13:20:29

attr  HarmonyHub  [room        ]  harmony

Attributes
event-on-change-reading           state,currentActivity          deleteattr
room                              harmony                        deleteattr
```

Figure 2: The FHEM module used for integrating the Harmony Hub.

The configuration will become second nature after you have added a couple of devices. But be careful: Every configuration change to FHEM has to be confirmed by pressing the *Save config* button in the web interface. This step adds the new settings to the `fhem.cfg` file.

As soon as you have successfully integrated the Harmony Hub, you can click on the `HarmonyHub` FHEM device and view the values that appear in the *DeviceOverview* window (Figure 2). These values are known as internals and readings in FHEM speak.

The readings are of interest for automation rules, which I will discuss later. In the case of the Harmony Hub, the readings are fairly terse, but interesting nonetheless. For example, the `currentActivity` reading provides information on the current activity. You can see, for example, whether television mode is enabled or whether the controllable devices are in standby mode.

Conversely, you can send switching commands from FHEM to the Harmony Hub via the `set` commands, which are also displayed automatically and can be selected via drop-down menus. In this way, you can simulate input from the Harmony Remote.

Getting Started with Automation

The whole thing becomes exciting as soon as you start using the available

Listing 3: FHEM console commands

```
01 attr HarmonyHub event-on-change-reading state,currentActivity
02 define LampSwitchOff notify HarmonyHub.currentActivity:.starting.Apple set Living room lamp off
03 define News at *20:00:00 set HarmonyHub activity Television;; set HarmonyHub command TV Number 1
```

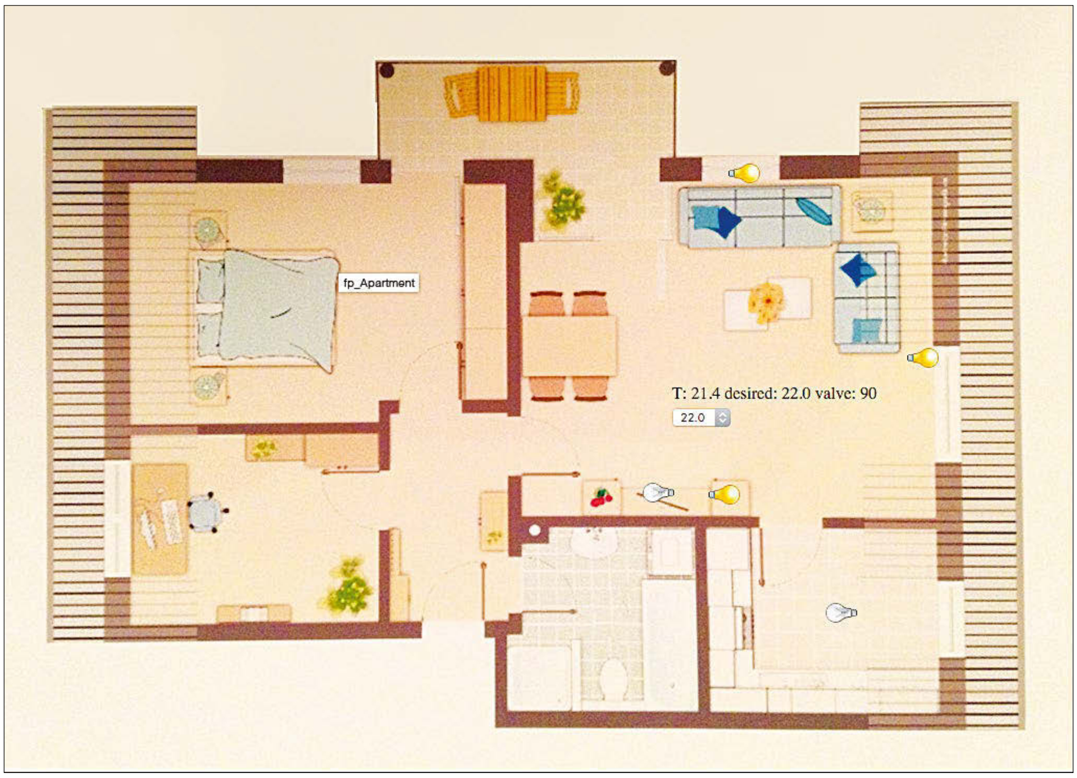


Figure 3: The FHEM floor plan offers users a 2D visualization.

readings to trigger other devices. The `currentActivity` reading, which reacts in real time to changes in the Harmony Remote's selected activity, is useful for triggering actions. To enable FHEM to also use this reading for automation purposes later on, you still need an attribute. The FHEM console command from line 1 of Listing 3 enables the Harmony Hub device to actually trigger an activity when changes to the `state` and `currentActivity` readings occur.

Some readings can already possess this attribute by default, although this is not always the case. It is best to use a self-assigned attribute value to define which reading is allowed to perform an activity. But beware: Manually defined attribute values override a device's default values.

The FHEM console command from line 2 of Listing 3 can be used to automatically switch off the FHEM `Living room lamp` device with an `off` command as soon as the current Harmony Hub activity switches to `starting.Apple`.

The `notify` command performs the notification, but you can also use the `doif`, `doelseif`, and `doelse` options to add conditional branching.

If you are wondering about the dots in line 2 of Listing 3: Spaces are always written as dots in order to be interpreted correctly by the Perl engine working in the background.

All events starting up in FHEM can be viewed in the `Event monitor` in the FHEM user interface.

In addition to `notify` and `doif`, there is also the `at` command, which you can use to start a certain action at a certain time every day. The FHEM console command from line 3 of Listing 3, for example, starts the Harmony Hub `Television` activity daily at 8 pm. The TV then switches to channel 1.

Alternative Interfaces

The user interface, which is fairly frugal in the eyes of many users, has been hotly debated within the FHEM scene for years. Rudolf Koenig, the founder of FHEM, was never really interested in creating a sexy user interface. Besides changing the foreground and background colors, and changing the logo, you have very few options for customizing the user interface.

A Floorplan extension (Figure 3) was created to map out FHEM objects on a 2D floor plan, visualize status messages, and let users output switching commands. As the use of mobile devices grew, however, Floorplan was marginalized due to its less than responsive design.

smartVISU

FHEM supports smartVISU, the intelligent "visualization framework for better home experience," which was originally designed to create HTML-based visualizations for home automation based on the commercial KNX system.

You can map many simple elements, such as dimmers or buttons, but also more complex components, such as a comprehensive heating control system. Users can create individual pages for rooms using standard HTML expressions. The smartVISU framework provides tags that are embedded in an HTML structure to establish a connection between the visualization and the smart home devices set up in FHEM.

In order for FHEM integration to work, some preparations have to be made on the software side. These preparations include installing the required packages – including a web server,

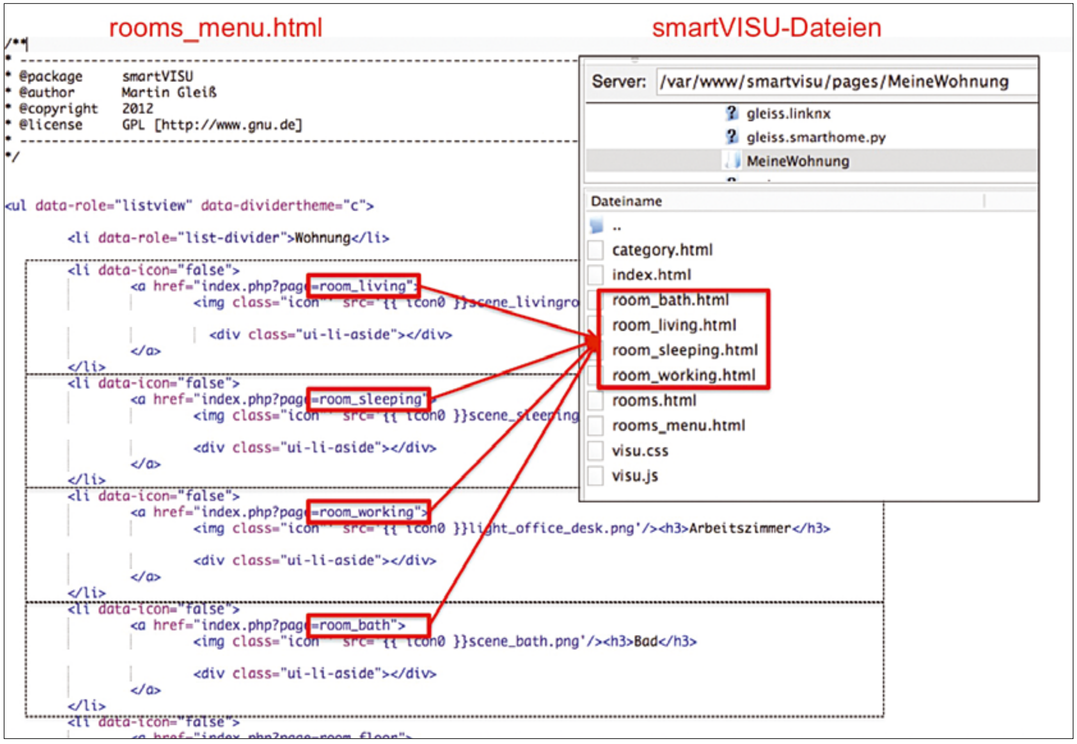
Listing 4: Preparing to Use smartVISU

```
$ apt-get update
$ apt-get -y install php5 libapache2-mod-php5 apache2 git
$ git clone https://github.com/herrmannj/smartvisu-cleaninstall.git
```

which is usually Apache 2 (Listing 4, lines 1 and 2). In addition, you need to download smartVISU (line 3), install the files in the `/var/www/smartvisu/` directory, and set the required user permissions. At the FHEM command line, you can then finally install the module and transfer it to the configuration with a `define` command (Listing 5).

Listing 5: Setting up the smartVISU Module

```
update force https://raw.githubusercontent.com/herrmannj/fronthem/master/controls_fronthem.txt

define fronthem fronthem
```



```
/**
* -----------------------------------------------------------------------
* @package      smartVISU
* @author       Martin Gleiß
* @copyright    2012
* @license      GPL [http://www.gnu.de]
* -----------------------------------------------------------------------
*/


<ul data-role="listview" data-dividertheme="c">

        <li data-role="list-divider">Wohnung</li>

        <li data-icon="false">
                <a href="index.php?page=room_living">
                        <img class="icon" src='{{ icon0 }}scene_livingro

                         <div class="ui-li-aside"></div>
                </a>
        </li>
        <li data-icon="false">
                <a href="index.php?page=room_sleeping">
                        <img class="icon" src='{{ icon0 }}scene_sleeping

                        <div class="ui-li-aside"></div>
                </a>
        </li>
        <li data-icon="false">
                <a href="index.php?page=room_working">
                        <img class="icon" src='{{ icon0 }}light_office_desk.png'/><h3>Arbeitszimmer</h3>

                        <div class="ui-li-aside"></div>
                </a>
        </li>
        <li data-icon="false">
                <a href="index.php?page=room_bath">
                        <img class="icon" src='{{ icon0 }}scene_bath.png'/><h3>Bad</h3>

                        <div class="ui-li-aside"></div>
                </a>
        </li>
        <li data-icon="false">
```

```
Server: /var/www/smartvisu/pages/MeineWohnung
  gleiss.linknx
  gleiss.smarthome.py
  MeineWohnung

Dateiname
  ..
  category.html
  index.html
  room_bath.html
  room_living.html
  room_sleeping.html
  room_working.html
  rooms.html
  rooms_menu.html
  visu.css
  visu.js
```

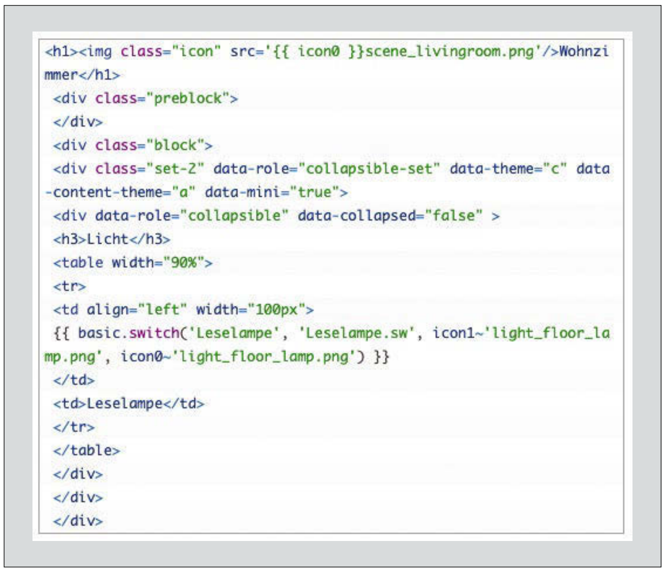
Figure 4: The smartVISU files have to be included in `rooms_menu.html`.

Now you need to look at the visualization elements that represent the devices contained on the room pages. For each room, you create an HTML file, which in turn is stored in the `rooms_menu.html` file (Figure 4). The individual devices are then entered in `rooms.html` as the `block content`. The syntax is shown in Figure 5 using a switch as an example.

It takes some hard work to map the desired objects from FHEM to smartVISU. Anyone who is not deterred by this, and is also prepared to tackle smartVISU's installation and configuration overhead, will be rewarded with a smart user interface. The interface even works extremely well on mobile devices. Step-by-step instructions (in German) are available on the web [3].

TabletUI

Meanwhile, TabletUI (Figure 6), another extensive visualization environment based on HTML, is enjoying increasing popularity. Similar to smartVISU, TabletUI lets users map information in a smart way. The GUI's focus,



```
<h1><img class="icon" src='{{ icon0 }}scene_livingroom.png'/>Wohnzi
mmer</h1>
 <div class="preblock">
 </div>
 <div class="block">
 <div class="set-2" data-role="collapsible-set" data-theme="c" data
-content-theme="a" data-mini="true">
 <div data-role="collapsible" data-collapsed="false" >
 <h3>Licht</h3>
 <table width="90%">
 <tr>
 <td align="left" width="100px">
 {{ basic.switch('Leselampe', 'Leselampe.sw', icon1~'light_floor_la
mp.png', icon0~'light_floor_lamp.png') }}
 </td>
 <td>Leselampe</td>
 </tr>
 </table>
 </div>
 </div>
 </div>
```

Figure 5: This code visualizes a switch in smartVISU.

which can display a multitude of functions simultaneously, is – as the name suggests – tablets.

If you want to use TabletUI, first update the FHEM packages using the FHEM console (Listing 6, line 1) and then import the TabletUI module (line 2). You now need to define all the desired elements in an HTML file. The `./www/tablet/index.html` file is primarily used for these element definitions.

The sample file `index-example.html`, which you can copy and save as `index.html`, is conveniently located in the same directory. This makes it possible to test the functionality more precisely.

Familiarize yourself with the syntax in order to establish the necessary links with the elements you wish to integrate from FHEM. In particular, the widgets in the TabletUI user interface need to be populated with the desired values (`data-get`), and the switching commands need to be sent to FHEM (`data-set`).

What's Next?

With its various modules and an active community, FHEM has become an important tool for open source IoT. However, the FHEM project also faces a number of (minor) problems. In particular, any enthusiast wanting to provide a new module can do so – assuming they follow a couple of rules. Accordingly, both the syntax and quality of the software can vary from module to module. It is not always easy for beginners to understand what commands need to look like.

The FHEM command reference [4] attempts to help by documenting the official modules. However, there are often delays before module changes appear in the command reference, or, in the worst case, changes may only be discussed on the FHEM forum. In case of problems, browsing through multi-page forum threads in the search for a solution may be your only hope.

In order to leverage the enormous possibilities that FHEM offers, it can also be useful to turn to commercial software. FHEM, with its wide range of modules, acts as a gateway between the third-party systems to be integrated on the one hand and a central control unit on the other. Sophisticated logic functions can often be implemented more easily with graphical solutions such as Node-RED or Loxone. FHEM is exclusively text-based, and Perl as programming language limits the possibilities in some cases.

Anyone who has ever failed to get a module to work in FHEM will understand the criticism being levied at the fact that users must resolve the software dependencies themselves. Resolving the dependencies sometimes results in unbelievably long-winded terminal commands needed to integrate the appropriate software versions into the basic system.

When you install a new system, you can quickly lose track of which dependencies have to be manually resolved, since the backup usually only contains the central FHEM configuration files. An automatic installation of the required resources, as in the case of Node-RED or openHAB, would be the easier approach for most users. But when it comes to depth of integration, you would be hard-pressed to find a system that matches FHEM. ■■■

Listing 6: Installing TabletUI

```
01 update all
02 define TabletUi HTTPSRV ftui/ ./www/tablet/ TabletUI
```

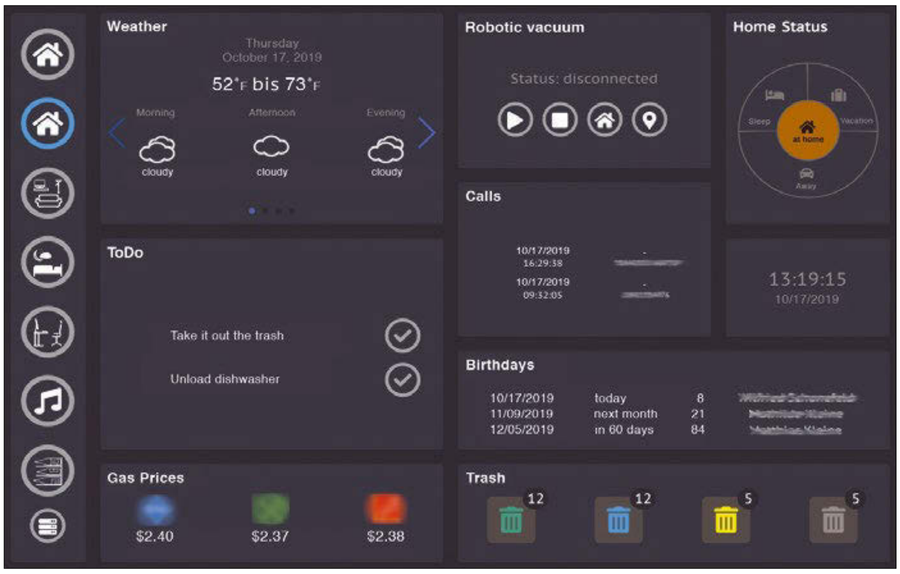


Figure 6: TabletUI offers an attractive alternative user interface.

Info

[1] FHEM: *https://fhem.de*

[2] Setting up FHEM: *https://www.meintechblog.de/2016/05/fhem-server-auf-dem-raspberry-pi-in-vweniger-als-einer-stunde-einrichten/* (German only)

[3] Setting up smartVISU: *https://www.meintechblog.de/2015/06/smartvisu-mit-fhem-die-perfekte-visualisierung-teil-1-basics/* (German only)

[4] FHEM command reference: *https://fhem.de/commandref.html*

Setting up a smart home command center with Z-Wave

Connecting Worlds

The RaZberry daughter board for your Raspberry Pi unlocks the power of the Z-Wave home automation environment. *By Christoph Langner*

Several vendors compete in the field of home automation. The solutions differ in price and also in terms of openness and interoperability: Many smart home solutions only work within the limits set by the manufacturer, and the devices only collaborate with a controller from the same company.

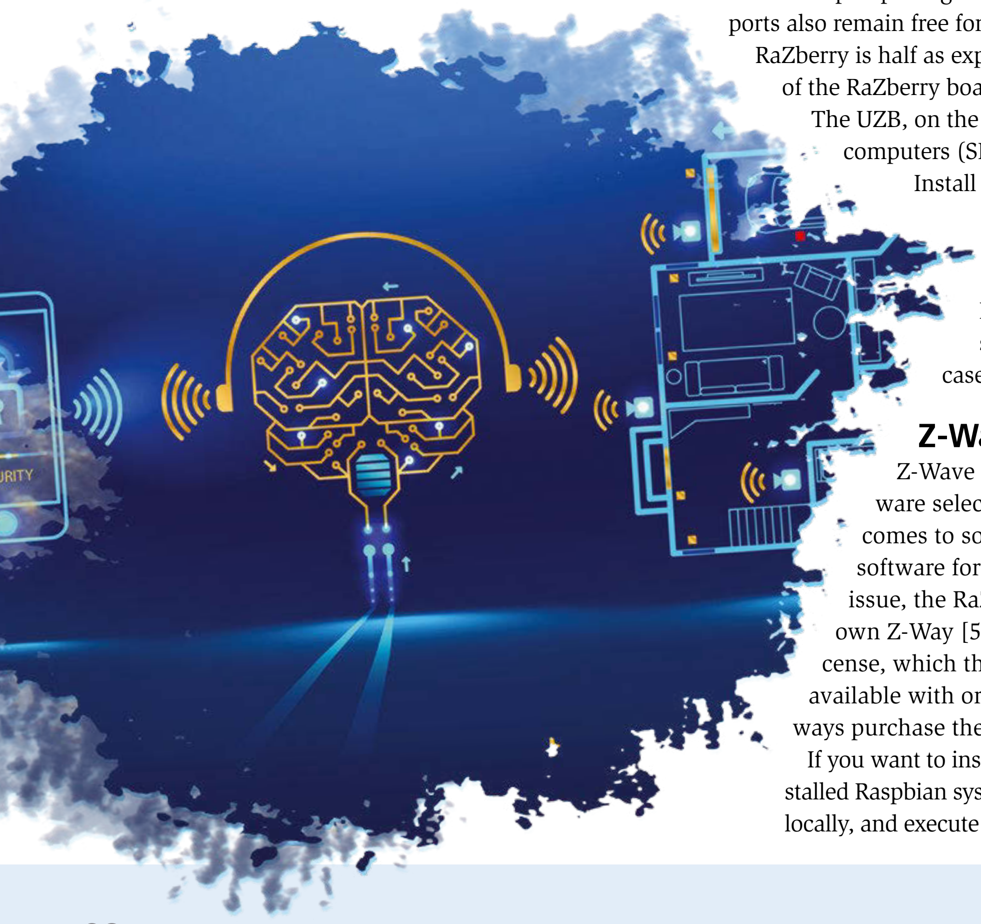
The Z-Wave Alliance [1] takes a different approach: The underlying system's protocol is open, and many manufacturers now offer Z-Wave-compatible devices or services. In addition to Z-Wave founder Sigma Designs, several hundred companies now belong to the consortium [2]. Among them are numerous well-known IT names, such as D-Link, devolo, Logitech, and Zyxel; large electronics groups, such as Bosch, LG, and Panasonic; and many lesser well-known companies. Certification ensures compliance with compatibility standards.

Z-Wave for the Rasp Pi

One of the most attractive features for home users is the fact that a Raspberry Pi is all you need for a Z-Wave control center – all you have to do is teach your Rasp Pi the Z-Wave protocol. You can do this either with the UZB [3], a Z-Wave-ready USB stick suitable for Linux, Mac OS X, and Windows, or with the additional RaZberry board [4], which I will describe in this article.

The RaZberry is a small daughter board that fits over the Rasp Pi's GPIO pins. One advantage of the RaZberry board is that the board is so small that it usually fits into standard Rasp Pi packages without any components protruding. The USB ports also remain free for further expansion. In addition, the EUR50 RaZberry is half as expensive as the EUR100 UZB. One disadvantage of the RaZberry board is that it only works with a Raspberry Pi. The UZB, on the other hand, can connect to other single board computers (SBCs) and even conventional PCs.

Install the RaZberry on the first 10 pins of the Raspberry Pi GPIO. The board is powered directly via the GPIO pins. The board is a bit lower than the USB ports on the Rasp Pi, so the Rasp Pi and RaZberry can be mounted in any standard case. You will want to use a plastic case – metal cases screen off radio signals.



Z-Way on the Rasp Pi

Z-Wave opens many doors when it comes to hardware selection and also offers alternatives when it comes to software. As a counterpart to the open source software for home automation FHEM presented in this issue, the RaZberry manufacturer Z-Wave.Me offers its own Z-Way [5] software. To use Z-Way, you'll need a license, which the RaZberry module already includes. UZB is available with or without license. If necessary, you can always purchase the license later.

If you want to install Z-Way, you have to log in from a freshly installed Raspbian system based on a RPi2 or RPi3, either via SSH or locally, and execute the command shown in Listing 1. If you don't want to install manually, you can alternatively reinstall the memory card of the Rasp Pi with the image provided by Z-Wave.Me.

Before you install, keep in mind that the Raspberry Pi has to learn the correct time zone for its location via:

```
sudo raspi-config 5 Internationalisation Options | ⮐
  2 Change Timezone
```

Otherwise, the system may react incorrectly to time-controlled actions later on, even though the web interface displays the correct time (Figure 1).

In addition, you'll need to reconfigure the serial port of the Raspberry Pi. In the default setting, the serial port operates as a terminal port for operating the Rasp Pi directly. In this case, FHEM will use the serial port to control the RaZberry. You will need to open the Rasp Pi's configuration tool by typing `sudo raspi-config` and answer *No* to the question *Would you like a login shell to be accessible over serial?* below *9 Advanced Options* | *A8 Serial* (Figure 2).

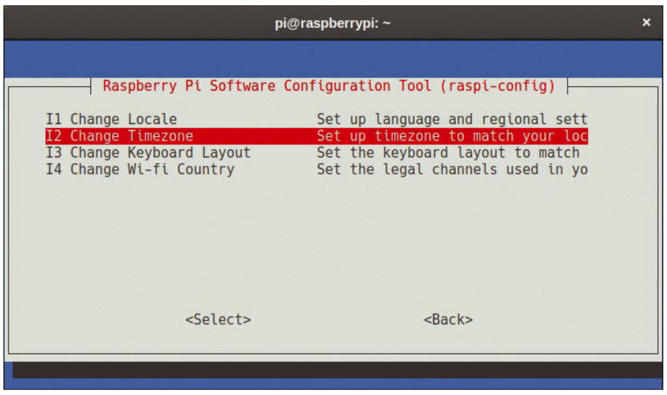
Installing Z-Way

The command from Listing 1 loads an installation script from the Internet and immediately calls it with root privileges. After confirming the terms of use, the installer automatically imports any missing dependencies into the system and creates start-stop scripts so that the service starts together with the computer. Finally, the routine offers to register you with the Z-Wave.Me email distribution list.

The installation routine sets up the program and also updates it. To complete the installation, you need to run the script again with the same command. After setting up the software, you can access the interface in a browser via port 8083. If name resolution on the network works properly and only one active Rasp Pi is on the network, *http://raspberrypi:8083* should also work.

If you work directly on the Rasp Pi system, you need to address localhost instead, i.e., *127.0.0.1:8083*. In case of difficulties finding the Rasp Pi, a help page from Z-Way [6] automatically searches for the Rasp Pi and links directly to the web interface of the Z-Way system discovered on the net.

Since the FHEM developers are continuously developing improvements for various modules of the Smart Home system, you need to be sure to update FHEM by entering the `update` command at the FHEM console.



```
Raspberry Pi Software Configuration Tool (raspi-config)

I1 Change Locale            Set up language and regional sett
I2 Change Timezone          Set up timezone to match your loc
I3 Change Keyboard Layout   Set the keyboard layout to match
I4 Change Wi-fi Country     Set the legal channels used in yo

        <Select>                    <Back>
```

Figure 1: Users have to be careful to set Raspian to the correct time zone with the configuration tool. Otherwise, time-controlled actions will not work correctly.

Listing 1: Installer

```
$ wget -q -O - razberry.z-wave.me/install | sudo bash
Do you accept Z-Wave.Me licence agreement?
Please read it on ZWave.Me web site:
http://razberry.z-wave.me/docs/
ZWAYEULA.pdf
yes/no: yes
z-way-server new installation
Installing additional libraries
OK
http://archive.raspberrypi.org jessie InRelease
OK
http://mirrordirector.raspbian.org jessie InRelease#
[...]
```

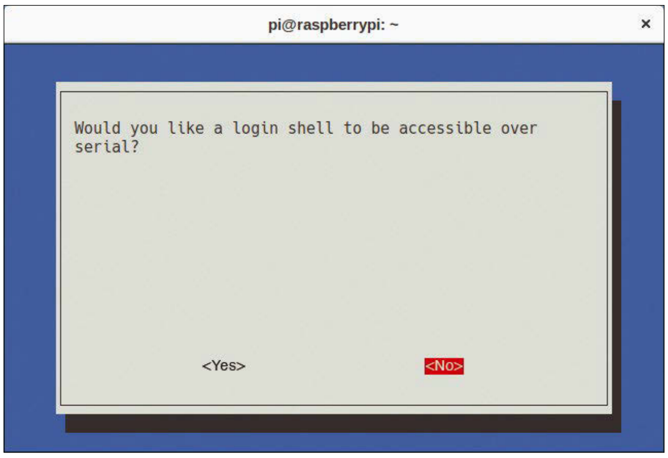
Registering Z-Wave Devices

It is advisable to set up a password for the administrative user *admin* on the Z-Wave interface. The system then automatically forwards the user to the initially empty dashboard. The interface speaks English by default, although you can change the language using *My Settings*.

To teach new devices, you then open the gear menu again and switch to the menu item *Devices* | *Add new*. Z-Way lists a number of manufacturers of Z-Wave compatible devices. It is very likely that your device will not be listed; instead, press *Add and automatically identify new Z-Wave device* at the top of the page.

Now you can put the first Z-Wave device into operation. For example, you can plug a switchable socket into a socket on the wall or insert batteries into a Z-Wave thermostat mounted on a heater. Then switch the device to connection mode. With some products, this happens automatically in the unconfigured state; with others, you have to press a button for a few seconds until a flashing LED signals connection mode. For additional information, see the device's instruction manual of the device.

While the RaZberry module is in teach-in mode, its red LED is lit continuously (Figure 3). The adjacent green LED indicates successful data transmissions to a connected Z-Wave device. As soon as the light indicates success, press *Start teach-in* (inclusion) in the next dialog and wait until the teach-in process



```
Would you like a login shell to be accessible over
serial?

        <Yes>                    <No>
```

Figure 2: To enable FHEM to integrate the RaZberry module, you'll need to deactivate the terminal function of the serial interface.

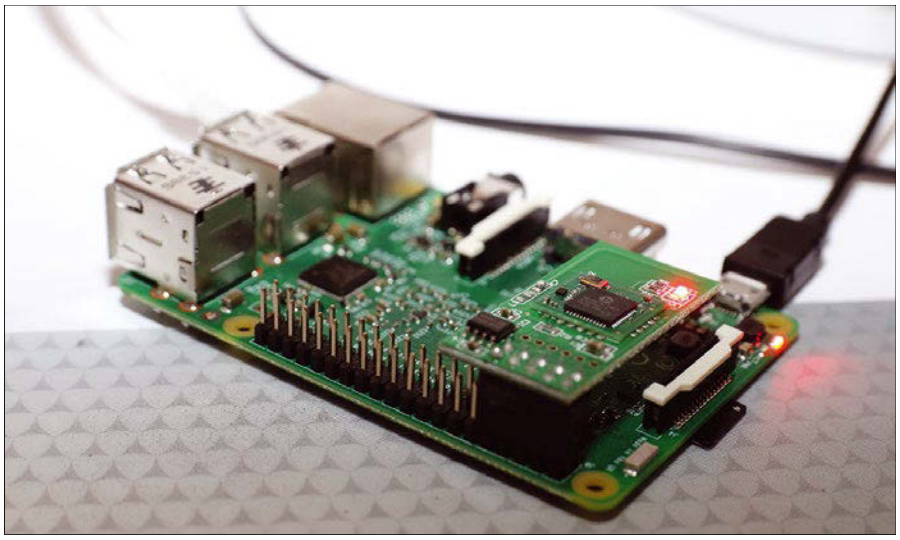


Figure 3: If the red LED on the RaZberry board is continuously lit as shown here, then the module is currently in teach-in mode.

is completed (Figure 4). In Z-Wave jargon, the control center now interviews the device.

In the dialog that follows, you can enter the name of the new device and the designations of the various control variables, sensors, and actuators – with a thermostat, for example, the variables might include the target temperature, the actual temperature, or the brightness of the display lighting (Figure 5). The device can also be assigned to a room if a room has already been defined. However, this assignment can also be defined later.

Dashboard

The Z-Way web interface is divided into four different tabs. Clicking on the house icon takes you to the dashboard, which is still empty in the beginning – more on this later. In the second tab, the Z-Wave devices registered with the system can be sorted by rooms, named as desired, and illustrated with your own photos.

The third tab lists all the actuators and sensors (Figure 6). The gear symbol displayed on each device takes you to the configuration. Here you can hide the device, add it to a room, rename it, or integrate it into the dashboard. This gives you quick access to the most important devices in your home without having to search for a long time. For a better overview, you can sort or filter the list by elements, tags, or chronological order.

The list can be used both to manage the individual devices and to read off their sensor values (such as temperatures or current power consumption) or to control the actuator of the device.

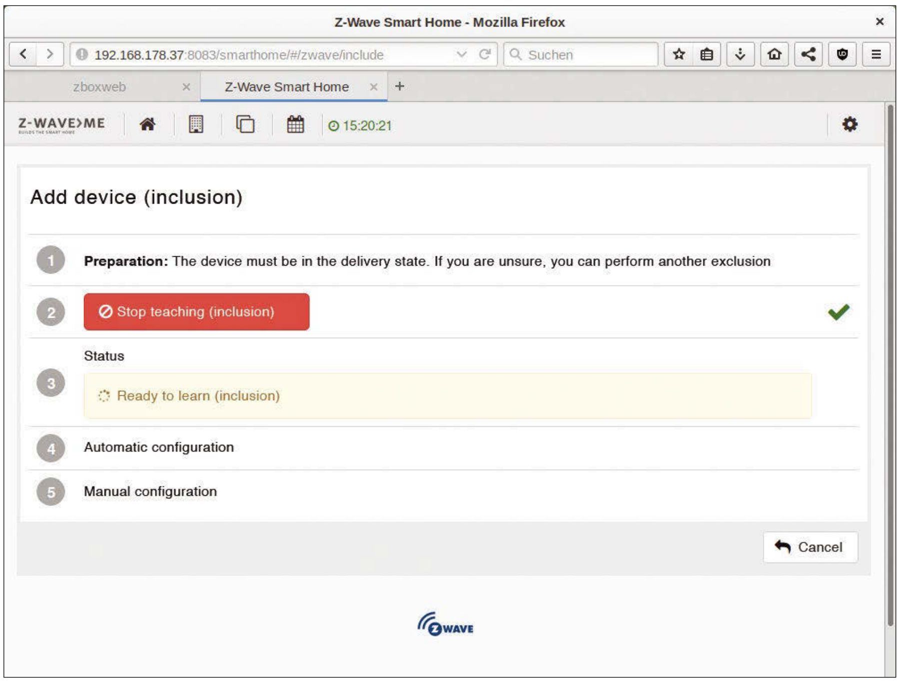
Expert Interface

If the settings options at this point do not meet the requirements, the URL *http://RasPi-IP_address:8083/expert* will call up an extended user interface (Figure 7). In addition to the individual switching options and sensor data, the user interface offers a great deal of additional information on the integrated devices. This includes, for example, details of the firmware version and the SDK protocol. As a rule, however, the standard interface is sufficient.

The dashboard and the device overview now let you control and read out the integrated Z-Wave devices, but still nothing happens automatically in your house. To set up an automation, you first have to define your own rules or scenarios via the *Applications* item in the web interface's gear menu and, if necessary, set up external applications.

Listing 2: Server Status

```
$ sudo service z-way-server status
* z-way-server.service - LSB: RaZberry Z-Wave service
Loaded: loaded (/etc/init.d/z-way-server)
Active: active (running) since Fri 2016-04-08 13:11:03 UTC; 43s ago
Process: 1054 ExecStop=/etc/init.d/z-way-server stop (code=exited,
status=0/SUCCESS)
Process: 1058 ExecStart=/etc/init.d/z-way-server start (code=exited,
status=0/SUCCESS)
CGroup: /system.slice/z-way-server.service
|?1061 z-way-server
```


Z-Wave Smart Home - Mozilla Firefox

192.168.178.37:8083/smarthome/#/zwave/include

Add device (inclusion)

1 **Preparation:** The device must be in the delivery state. If you are unsure, you can perform another exclusion

2 Stop teaching (inclusion) ✓

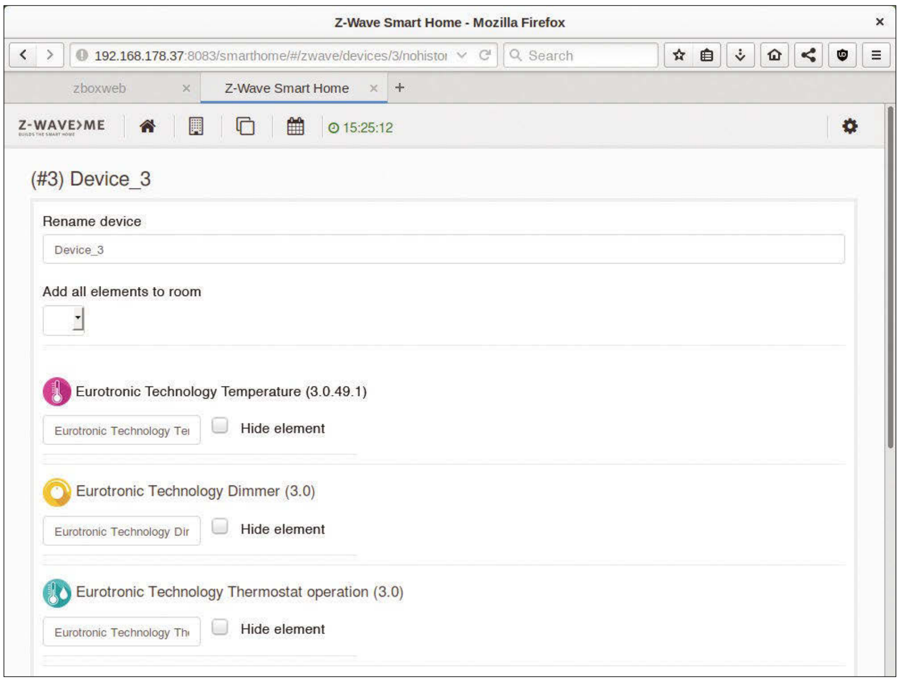
3 Status
Ready to learn (inclusion)

4 Automatic configuration

5 Manual configuration

Cancel

Figure 4: To teach new Z-Wave devices, switch them to connection mode and launch the inclusion function in the Z-Way interface.


Z-Wave Smart Home - Mozilla Firefox

192.168.178.37:8083/smarthome/#/zwave/devices/3/nohistory

(#3) Device_3

Rename device: Device_3

Add all elements to room

Eurotronic Technology Temperature (3.0.49.1) — Eurotronic Technology Te — Hide element

Eurotronic Technology Dimmer (3.0) — Eurotronic Technology Dir — Hide element

Eurotronic Technology Thermostat operation (3.0) — Eurotronic Technology Th — Hide element

Figure 5: After teaching, assign intuitive names to the new device, the sensors, and the actuators.

Z-Way breaks the view down into three tabs: *Local* contains the existing apps. Below this, the program accepts simple IF/THEN rules, schedules, or actions, such as dispatching email or push messages via external services such as Pushover [7] or Pushbullet [8].

The *Server* tab contains a kind of app store, which you can use to install further features, including, say, an astronomy module that lets you discover where the sun is in the sky, and control a blind accordingly, and a camera module for the Rasp Pi's webcam. You can also use this tab to update previously installed modules.

Note that both tabs initially list only a few featured apps. A complete overview can be obtained by changing the filter to the right of the search field to *All Apps*. Clicking on *New App* lets you set up the selected function. You can add extensions that have not yet been installed to the system by clicking *Download*.

The best way to explain how the apps work is to use an example: A schedule is created as a new app for time-controlled switching of an electrical socket outlet. In the next dialog, the app is given a name, the weekdays for the action are selected, and finally a time is entered.

After this, you can define actions, such as setting a dimmer, operating a motor, or setting a thermostat. In the case of a power outlet, you simply set the appropriate switch to *On* or *Off*. Click *Save* to complete the configuration. The app should then appear in the *Active* tab and should switch the device as defined at the appropriate time.

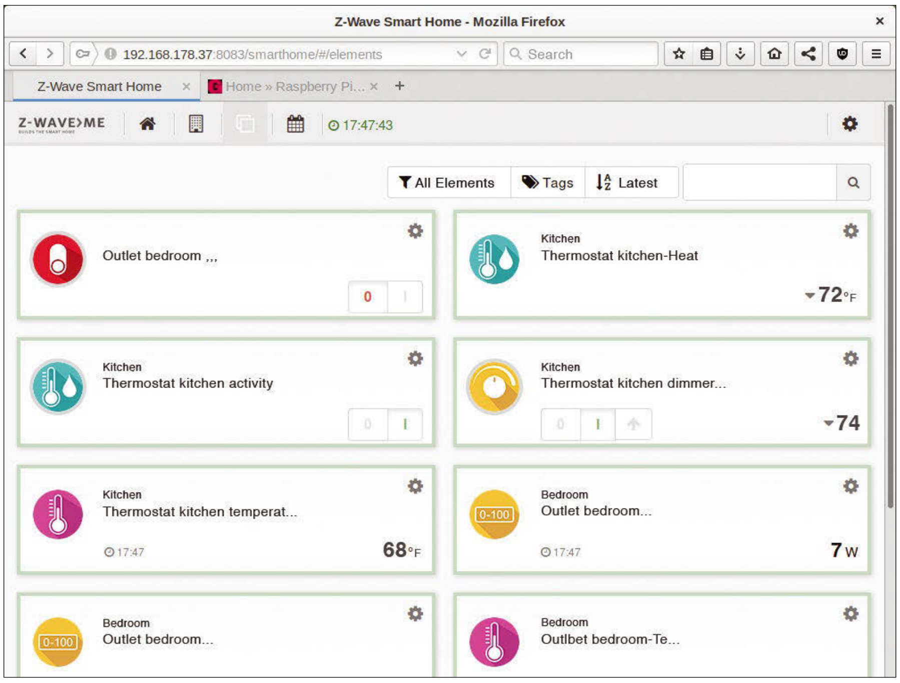
As your home automation system grows, so does the scope of the rules and regulations. Anyone installing many Z-Wave-compatible sensors, such as motion detectors, window and door sensors, brightness and humidity meters, smoke detectors, automated roller shutters, awnings, or even door locks will invest a lot of time in designing the rules. You therefore have the option to save the rules on a computer via the cogwheel menu's *Management* item and reload them if necessary. The same menu also provides the opportunity to create new users, update the RaZberry's firmware, or remotely access the Z-Way configuration interface.

Controlling the Z-Way Server

During the installation of the Z-Way software on the Raspberry Pi, the installer imports a service into the system. If the Z-Way configuration interface cannot be reached or the service reports an error, you can query the status of this service via the terminal in Debian-typical style (Listing 2). If required, the service can also be stopped or started in the same way (Listing 3).

The start-stop script proved to be important during the test. Although the service launched after rebooting the Raspberry Pi (you could log into the web front end), it did not list any devices, even though they were still present before rebooting. As soon as you restart the service while Rasp Pi is running (*restart*), Z-Way lists the lost devices again.

In order to analyze problems that cannot be solved at first glance, it is always worth taking a look at the service logfile


Z-Wave Smart Home - Mozilla Firefox

192.168.178.37:8083/smarthome/#/elements

Outlet bedroom ,,, — 0

Kitchen — Thermostat kitchen-Heat — 72°F

Kitchen — Thermostat kitchen activity

Kitchen — Thermostat kitchen dimmer... — 74

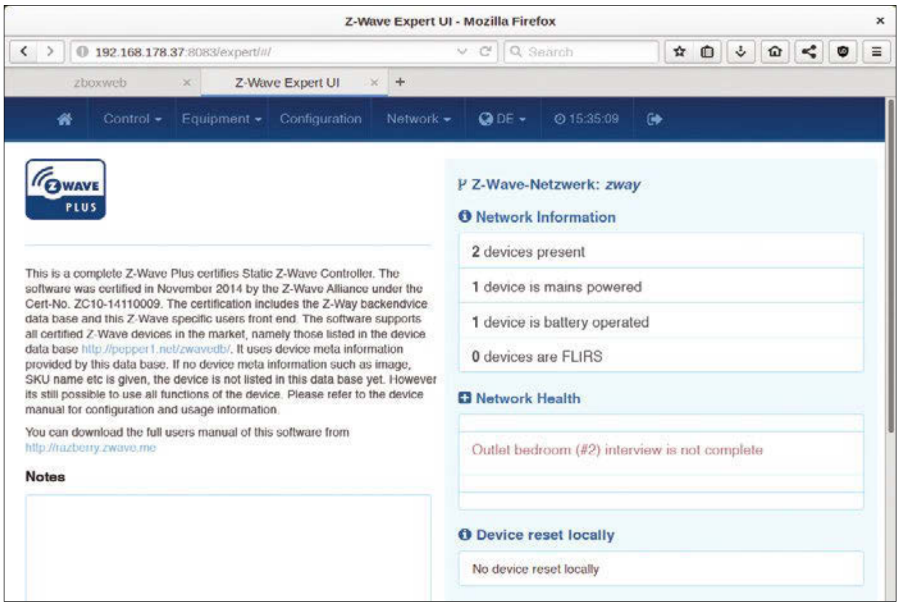
Kitchen — Thermostat kitchen temperat... — 17:47 — 68°F

Bedroom — Outlet bedroom... — 17:47 — 7 W

Bedroom — Outlet bedroom...

Bedroom — Outlbet bedroom-Te...

Figure 6: In the overview, the Z-Way interface lists all devices connected to the Z-Wave network. You can also carry out actions manually.


Z-Wave Expert UI - Mozilla Firefox

192.168.178.37:8083/expert/#/

This is a complete Z-Wave Plus certifies Static Z-Wave Controller. The software was certified in November 2014 by the Z-Wave Alliance under the Cert-No. ZC10-14110009. The certification includes the Z-Way backendvice data base and this Z-Wave specific users front end. The software supports all certified Z-Wave devices in the market, namely those listed in the device data base http://pepper1.net/zwavedb/. It uses device meta information provided by this data base. If no device meta information such as image, SKU name etc is given, the device is not listed in this data base yet. However its still possible to use all functions of the device. Please refer to the device manual for configuration and usage information.

You can download the full users manual of this software from http://razberry.zwave.me

Notes

Z-Wave-Netzwerk: *zway*

Network Information

2 devices present

1 device is mains powered

1 device is battery operated

0 devices are FLIRS

Network Health

Outlet bedroom (#2) interview is not complete

Device reset locally

No device reset locally

Figure 7: The Expert UI provides important information for problem solving. Experienced users will also find more detailed configuration options here.

`z-way-server.log` in the `/var/log/` directory (Listing 4). The log contains numerous bits of information about events during start-up and operation – including incorrect login data for external services, the failure of Z-Wave devices, or bugs in the program itself. In addition, the aforementioned Expert user interface is useful for finding errors.

Z-Way Apps

You don't necessarily want to start the PC for every action in your home, especially considering that you always have a computer at hand in the form of your smartphone or tablet. In the Android and iOS app stores, you will also find various apps that are based on the extensively documented Z-Way API and enable you to control your smart home via your mobile phone. In addition to the two official apps, Z-Way for Android users and ZWay Home Control for iOS users, there are a number of alternative applications, including the Z-Way Control app, for example.

In our test, however, the Android apps in particular did not make a convincing impression. At the time of testing, the official Z-Way app for Android was unable to log on to the server software due to a change in the authentication method. Despite correct access data, the app only reported *Can't connect!*.

The alternative app Z-Way Control made it possible to log on, but this app only displays the web interface used on the PC in a web view. The advantage of the application is that it loads parts of the website locally and saves data volume during access via a mobile Internet connection.

Listing 3: Restart

```
$ sudo service z-way-server start
$ sudo service z-way-server stop
$ sudo service z-way-server
restart
```

Conclusions

If you want to use a Raspberry Pi as your home control center and integrate Z-Wave devices into the installation, you will need the RaZberry board or the UZB stick. The RaZberry is particularly well-suited, because the installation requires hardly any time or effort, the costs are kept within limits, and the USB ports remain free for other applications. The UZB offers the advantage that the GPIO of the Raspberry Pi remains usable for other tasks, and the stick also works well on conventional computers.

The Z-Way server is an easy way to get started, since the software can be installed without a great deal of Linux knowledge and even more complex tasks can be easily configured in the web interface. For this, however, you have to be satisfied with Z-Wave devices only – integrating devices from other Smart Home standards is not possible with Z-Way.

In case of problems, you can get help either from the official support or from the Z-Wave.Me forum. Developers will find good and detailed documentation for the module and the APIs provided by Z-Way on the RaZberry homepage.

In addition to FHEM, there are other open source home automation solutions that support the RaZberry and Z-Way as interfaces. These solutions include OpenRemote and Freedomotic. However, for anyone who values the broadest possible support for a variety of services and protocols, as well as a broad and committed community, it makes sense to integrate the RaZberry with FHEM. ■■■

Info

[1] Z-Wave Alliance: *https://z-wavealliance.org*

[2] Z-Wave member companies: *https://z-wavealliance.org/z-wave_alliance_member_companies*

[3] UZB: *https://z-wave.me/uzb*

[4] RaZberry: *https://z-wave.me/products/razberry*

[5] Z-Way: *https://z-wave.me/z-way*

[6] Find page: *http://find.zwave.me*

[7] Pushover: *https://pushover.net*

[8] Pushbullet: *https://www.pushbullet.com*

Listing 4: z-way-server.log

```
$ tail -f /var/log/z-way-server.log
[2016-04-08 13:11:03.334] [I] [core] Executing script: exit()
[2016-04-08 13:11:03.700] [I] [core] Executing script: /*** Z-Way Home
Automation Engine main executable ***************************** ...
[2016-04-08 13:11:03.702] [I] [core] Executing script: // Comon
utilities and functions ...
[2016-04-08 13:11:03.707] [I] [core] Executing script: // This script
transforms old formats to new ...
```

Choosing a Vim Plugin Manager

Vim Housekeeping

A plugin manager can help you corral your growing collection of Vim plugins. Choosing one depends on your personal preferences. *By Bruce Byfield*

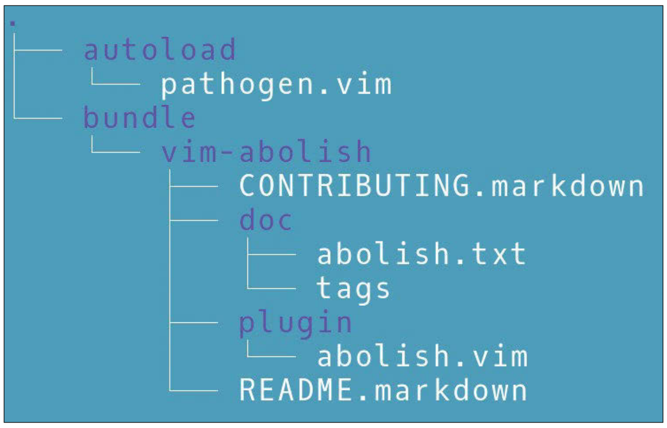
New users of the Vim text editor may be content to use it as installed from their distribution's repositories. However, Vim has hundreds of plugins, and installing one often leads to adding more out of curiosity. However, multiple plugins can quickly become a management nightmare, because unadorned Vim dumps plugins into one directory, which makes file management difficult.

As a result, a variety of Vim plugin managers have been released over the years. Several begin by installing each plugin to a separate directory to simplify file management. Most involve creating and editing a `~.vimrc` file in your home directory. Some are plugins themselves, while a few operate at least partly outside of Vim's structure. Many are housed on GitHub, with instructions specialized for that site.

Listed here are some of the most common Vim plugins that run from the command line. Each makes its own assumptions about its users' knowledge or preferences. Only basic instructions are given, but most of the managers are well-documented online.

Pathogen

Pathogen [1] is the oldest manager that runs from within Vim. Like many that were created after it, Pathogen rearranges the Vim directory structure, providing a separate directory for the files for each plugin, an innovation that makes installation, upgrading, and deletion more efficient (Figure 1). However, unlike some other managers, Pathogen does not automatically upgrade or delete plugins.



```
.
├── autoload
│   └── pathogen.vim
└── bundle
    └── vim-abolish
        ├── CONTRIBUTING.markdown
        ├── doc
        │   ├── abolish.txt
        │   └── tags
        ├── plugin
        │   └── abolish.vim
        └── README.markdown
```

Figure 1: Pathogen's major innovation is to give Vim an orderly directory structure, with a separate directory for each plugin.

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/*.

To install Pathogen, enter the following at the command line:

```
mkdir -p ~/.vim/autoload ~/.vim/bundle ↩
  && curl -LSso ↩
  ~/.vim/autoload/pathogen.vim ↩
  https://tpo.pe/pathogen.vim
```

Users will also need to add lines to their `.vimrc` file to start Pathogen before any other plugin, so that it can manage the rest:

```
execute pathogen#infect()
syntax on
filetype plugin indent on
```

These lines are the minimum required to use Pathogen, but you may also enable automatic updates by adding the lines:

```
call pathogen#runtime_append_all_↩
  bundles()
call pathogen#helptags() "
```

Note, though, that this option can significantly extend Vim's startup time if you have a lot of plugins.

Other plugins can be added from GitHub, by running from within `~/.vim/bundle` the command:

```
git clone git://github.com/[WRITER]/ ↩
  [PLUGIN PATH] ↩
  ~/.vim/bundle/[PLUGIN PATH]
```

GitHub is a useful source, because many Vim plugins are housed there.

Despite Pathogen's age, many users continue to favor Pathogen for its simplicity and order. Since updates can sometimes go wrong, some also appreciate being able to avoid automatic updates.

If you are managing multiple remote plugins, have a look at vim-pandemic [2], which is designed to work alongside Pathogen.

Vundle

Vundle [3] is an enhancement of Pathogen. Originally, Vundle referred to plugins as "bundles," making its name an abbreviation for "Vim bundles." To Pathogen's rationalized directory structure, Vundle adds several utilities (Figure 2).

On Windows, Vundle has a graphical interface. To install Vundle on Linux, run:

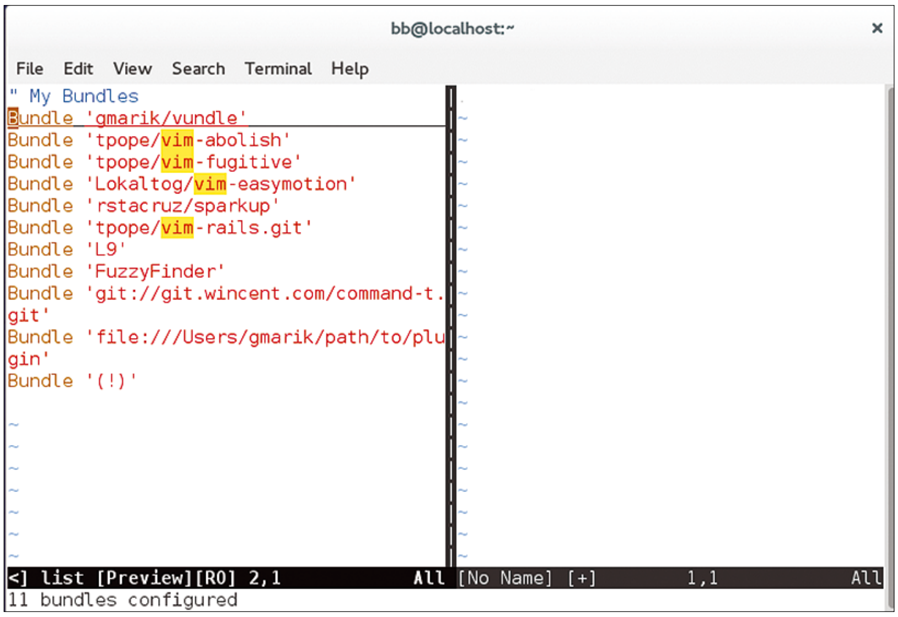
```
git clone https://github.com/↩
  VundleVim/Vundle.vim.git ↩
  ~/.vim/bundle/Vundle.vim
```

In other words, Vundle is installed to a subdirectory of `.vim/bundle`, just like any other plugin.

To update plugins automatically with Vundle, add to the following lines to `.vimrc`:

```
set nocompatible
filetype off

set rtp+=~/.vim/bundle/Vundle.vim
call vundle#begin()

Plugin 'VundleVim/Vundle.vim'
```

These lines set the run-time path to Vundle, initiate it, and set Vundle to manage itself. Below the last line, add the other plugins that Vundle manages,

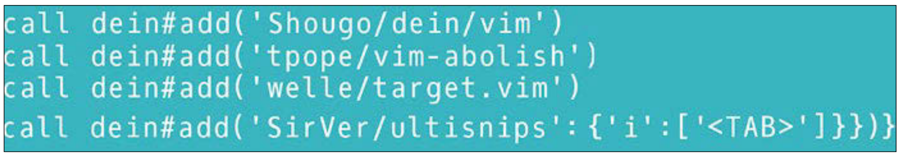


```
" My Bundles
Bundle 'gmarik/vundle'
Bundle 'tpope/vim-abolish'
Bundle 'tpope/vim-fugitive'
Bundle 'Lokaltog/vim-easymotion'
Bundle 'rstacruz/sparkup'
Bundle 'tpope/vim-rails.git'
Bundle 'L9'
Bundle 'FuzzyFinder'
Bundle 'git://git.wincent.com/command-t.
git'
Bundle 'file:///Users/gmarik/path/to/plu
gin'
Bundle '(!)'
<] list [Preview][RO] 2,1        All [No Name] [+]        1,1        All
11 bundles configured
```

Figure 2: Vundle is a popular second-generation plugin manager.

Listing 1: Dein.vim Code for .vimrc

```
if &compatible
  set nocompatible
endif
" Add the dein installation directory into runtimepath
set runtimepath+=~/.cache/dein/repos/github.com/Shougo/dein.vim

if dein#load_state('~/.cache/dein')
  call dein#begin('~/.cache/dein')

  call dein#add('~/.cache/dein/repos/github.com/Shougo/dein.vim')
  call dein#add('Shougo/deoplete.nvim')
  if !has('nvim')
    call dein#add('roxma/nvim-yarp')
    call dein#add('roxma/vim-hug-neovim-rpc')
  endif

  call dein#end()
  call dein#save_state()
endif

filetype plugin indent on
syntax enable
```



```
call dein#add('Shougo/dein/vim')
call dein#add('tpope/vim-abolish')
call dein#add('welle/target.vim')
call dein#add('SirVer/ultisnips':{'i':['<TAB>']}})
```

Figure 3: **Adding a plugin with dein.vim: First, load dein.vim into `.vimrc` followed by two plugins loaded at startup. The final plugin is lazy-loaded, to be started by pressing the Tab key.**

one per line, each starting with `Plugin`. For example:

```
Plugin 'tpope/vim-fugitive'
```

The GitHub Vundle page gives examples of how to list other sources that you might use.

Installing Vundle also installs four utilities:

- `:PluginList`, which lists configured plugins
- `:PluginInstall`, which installs or updates plugins
- `:PluginSearch`, which searches for a plugin
- `:PluginClean`, which removes plugins with a confirmation message

The last three are completed with the name of a plugin.

Vundle is an intermediate Vim manager, kept simple by having utilities that are run manually. It is a solid choice for constant Vim users who are always tinkering with their Vim installations, adding and updating their plugins.

Dein.vim and NeoBundle

A few years ago, a Vundle modification was released called NeoBundle [4]. NeoBundle's creator, Shougo Matsushita, went on to write dein.vim [5] to replace NeoBundle. Much like NeoBundle when first released, dein.vim has minimal English documentation. The documentation that is available assumes a strong knowledge of Vim and the use of plugins.

Like NeoBundle, dein.vim supports the Mercurial and Subversion version control systems, as well as Git. It can be set to use a specific Vim release, a feature that can keep an update from breaking Vim and its plugins. It differs from NeoBundle in that it uses system calls rather than utilities.

In addition, dein.vim itself is optimized for speed – although it is only 100 milliseconds or so faster than NeoBundle. Much of this speed is obtained by concurrent loading, which can make pinpointing a misbehaving plugin difficult at times.

To install dein.vim, enter:

```
mkdir ~/.vim/bundle
curl https://raw.githubusercontent.com/⮐
  Shougo/dein.vim/master/bin/⮐
  installer.sh > installer.sh
```

Then add the code in Listing 1 to `.vimrc`. Install `dein.vim` with:

```
:call dein('PLUGIN-PATH')
```

Plus add similar lines for other plugins to run at startup. To decrease startup time, other plugins can be set to be "lazy-loaded" – that is, loaded after startup by pressing a designated keystroke (Figure 3).

Dein.vim is an advanced manager, ideal for those with a dozen or more plugins, who want to get every last bit of speed from their installation. For those who want a better-documented, advanced plugin-manager, NeoBundle is still available.

vim-plug

Vim-plug [6] is a minimalist plugin manager, designed to be quickly installed and edited (Figure 4). To install, run the command:

```
curl -fLo ~/.vim/autoload/plug.vim ⮐
  --create-dirs
  https://raw.githubusercontent.com/⮐
  junegunn/vim-plug/master/plug.vim
```

To automatically update `plug.vim`, place Listing 2 in `.vimrc` before the line that starts with `call plug#begin()`.

To install new plugins, below the line that starts with `call plug#begin`, add one plugin per line in the format `Plug <PATH>`, much like in dein.vim. To borrow the example from the vim-plug tutorial, if your plugin source is GitHub, enter:

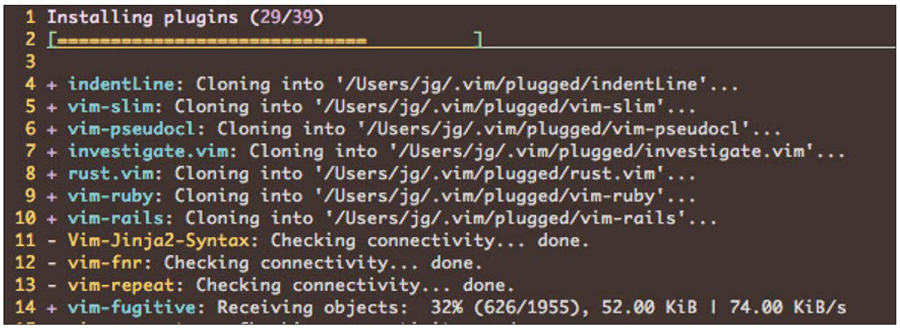
```
call plug#begin('~/.vim/plugged')

" Declare the list of plugins.
Plug 'tpope/vim-sensible'
Plug 'junegunn/seoul256.vim'
"
```

Note the double quotation marks around the list.

When the list is complete, restart Vim and run `:PlugUpdate` to complete the process. You can either review the changes by pressing *D* in the window or by running `:PlugDiff`. Should another plugin no longer work or interfere with another, you can see the latest changes with `:PlugDiff`. Scroll to the plugin's line and

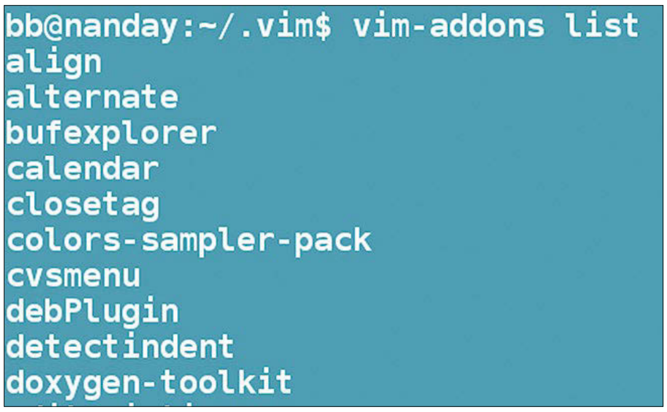
Listing 2: Code to Auto Update plug.vim

```
if empty(glob('~/.vim/autoload/plug.vim'))
  silent !curl -fLo ~/.vim/autoload/plug.vim --create-dirs
    \ https://raw.githubusercontent.com/junegunn/vim-plug/master/plug.vim
  autocmd VimEnter * PlugInstall --sync | source $MYVIMRC
endif
```



```
Installing plugins (29/39)
[==============================          ]

+ indentLine: Cloning into '/Users/jg/.vim/plugged/indentLine'...
+ vim-slim: Cloning into '/Users/jg/.vim/plugged/vim-slim'...
+ vim-pseudocl: Cloning into '/Users/jg/.vim/plugged/vim-pseudocl'...
+ investigate.vim: Cloning into '/Users/jg/.vim/plugged/investigate.vim'...
+ rust.vim: Cloning into '/Users/jg/.vim/plugged/rust.vim'...
+ vim-ruby: Cloning into '/Users/jg/.vim/plugged/vim-ruby'...
+ vim-rails: Cloning into '/Users/jg/.vim/plugged/vim-rails'...
- Vim-Jinja2-Syntax: Checking connectivity... done.
- vim-fnr: Checking connectivity... done.
- vim-repeat: Checking connectivity... done.
+ vim-fugitive: Receiving objects:  32% (626/1955), 52.00 KiB | 74.00 KiB/s
```

Figure 4: **A shot from the vim-plug homepage, showing one of the manager's advantages: Installing multiple plugins at the same time.**



```
bb@nanday:~/.vim$ vim-addons list
align
alternate
bufexplorer
calendar
closetag
colors-sampler-pack
cvsmenu
debPlugin
detectindent
doxygen-toolkit
```

Figure 5: **Vim-addon-manager edits Vim plugins from outside of Vim.**

roll back the changes by scrolling to its entry and pressing the *X* key. To remove a plugin, delete its `.vimrc` entry and run `:PlugClean`.

Vim-plug is a popular manager, because it offers many of the same features as more complex managers, but with an elegant simplicity. If I was recommending a single manager for all user levels, it would be vim-plug.

vim-addon-manager

Properly speaking, vim-addon-manager [7] is not a Vim plugin. Instead, it is an ordinary Linux package that is installed with a second package that contains over 20 classic Vim add-ons. Each of these add-ons has a status of installed, removed, disabled, broken, or unavailable. Using the command structure

```
vim-addon-manager install ADD-ON
```

users can install multiple add-ons, using a space-separated list (Figure 5).

Vim-addon-manager is useful as an introduction to Vim extensions. In addition, it places extension management into general package management, simplifying system administration. Its main drawback is the limited selection. However, the selection can be extended by placing other add-ons in `/usr/share/vim/addons` – a matter of file management rather than editing configuration files.

Volt

First released in 2015, Volt [8] uses the *go-git* library in its operations, a basic implementation of `git`'s "porcelain" or high-level, user-friendly options, operating partly outside of Vim. It can be installed with:

```
go get github.com/vim-volt/volt
```

Then, to make Volt functional, set the environment variables by adding to `.vimrc`:

```
$HOME/volt/rc/default/vimrc.vim ↩
  (installed to: $HOME/.vim/vimrc)
```

Another profile can replace `default`, using the online instructions [9]. If necessary, change the path to match your preference (Figure 6).

Plugins can be installed using the command:

```
volt get ↩
https://github.com/CREATOR/PLUGIN
```

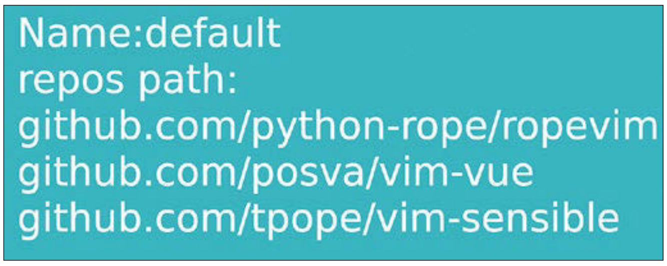
The command can be shortened to start with GitHub or even omit it, so long as GitHub is the source. The full command has the advantage of using what you copy and paste.

All plugins can be updated with:

```
volt get -l -u
```

Individual plugins can be updated with:

```
volt get -u /CREATOR/PLUGIN
```



```
Name:default
repos path:
github.com/python-rope/ropevim
github.com/posva/vim-vue
github.com/tpope/vim-sensible
```

Figure 6: **Volt supports multiple profiles. All profiles begin with a listing of repositories.**

Similarly, Volt itself can be updated with the command `volt self-upgrade`.

Volt is ideal for users familiar with `git`, or those who, while familiar with the command line, are only moderately comfortable with Vim and only occasionally use it.

Making a Choice

These are not the only choices for Vim plugins. Another whole subset are clones of Janus, which describes itself as a distribution with its own limited selection of available plugins. Increasingly, there are also managers for GUIS like GVim.

However, you don't have to go beyond the command line to find choices for several different backgrounds and levels of expertise. I would recommend vim-plug to practically anyone, but I would also suggest stating with Pathogen and then experimenting with its descendants Vundle, NeoBundle, and dein.vim until you find the level of complexity that best suits you. However you experiment, be careful to back up your `.vimrc` files, labelling the backups so you can revert to one if disaster strikes.

Like much of Vim, the plugin managers can appear complex at first. However, once you understand each one's basic premise, most of them become easier to use. Each can also help to reduce complexity as your collection of other plugins grows – as it almost certainly will. ■■■

Info

[1] Pathogen: *https://github.com/tpope/vim-pathogen*

[2] vim-pandemic: *https://github.com/jwcxz/vim-pandemic*

[3] Vundle: *https://github.com/VundleVim/Vundle.vim*

[4] NeoBundle: *https://github.com/Shougo/neobundle.vim*

[5] dein.vim: *https://github.com/Shougo/dein.vim*

[6] vim-plug: *https://github.com/junegunn/vim-plug*

[7] vim-addon-manager: *https://github.com/MarcWeber/vim-addon-manager*

[8] Volt: *https://github.com/vim-volt/volt*

[9] Volt profile: *https://github.com/vim-volt/volt#switch-set-of-plugins-profile-feature*

Breathe new life into an old computer with PXE boot and TinyCore Linux

Boot Story

Implementing PXE boot with TinyCore Linux lets you boot a computer over the network – a great solution for revitalizing old computing hardware. *By Andy Carlson*

One of my passions is enabling the reuse of older computer hardware that others consider obsolete. I picked up a used laptop at a local university's surplus equipment sale for $15. It was a pretty decent little thing – an older 32-bit dual core CPU with 1-2GB of memory – but my thought was that it would be a great laptop for my kids. The problem: It had no hard drive. My initial thought was to put Linux on a USB drive and boot to that. This worked wonderfully for the first few days – until my kids failed to remember to shut it down properly, which hosed the USB filesystem. To their sadness and my frustration, the computer sat for a few months. Then one day an idea popped into my head: PXE boot the laptop. Preboot eXecution Environment (PXE) is a standard for booting computers across the network. I knew it used DHCP, but that was the extent of my knowledge. This is the story of what I learned about PXE boot, how I implemented it using TinyCore Linux [1], and how I found the coolest, geekiest solution for my kids.

Laying the Groundwork

Almost all of us have seen the PXE or Network boot option on our BIOS screen. PXE uses the network interface to load the operating system and necessary configuration files over a network. This is accomplished using two particular network protocols: Dynamic Host Configuration Protocol (DHCP) and Trivial File Transfer Protocol (TFTP). DHCP is used to request client configuration data before any operating system is loaded. Once the client has network access, files are downloaded via TFTP and are loaded into memory on the client machine. Once the files are loaded, the operating system will boot on the client completely in memory without requiring any local storage. Knowing a little about these two components will improve the ability to properly implement a PXE environment.

One of the key components of a PXE boot environment is a DHCP server. DHCP allows a client configuration to be requested and delivered over a network. When a client device connects to a network, a series of requests and responses must take place to complete a DHCP session using the following steps:

1. The client device issues a broadcast request for available DHCP servers.
2. If there is an available DHCP server, it will respond to the client indicating that it is available.
3. The client sends a request to the DHCP server.
4. The server responds with the configuration data for the client device.

The data returned to the client in Step 4 is where DHCP's power lives. Although the host configuration portion of the response contains the client IP address and gateway address (along with a few other pieces of data), the response also includes a section containing other configuration options. Depending on the DHCP server configuration, this step can be used to provide many other pieces of client configuration data, including DHCP lease information, router addresses, DNS servers, the time server, the TFTP server address, and many other parameters. The PXE hardware on most computers has the ability to perform the preceding steps to obtain configuration data from a DHCP server. At this point, the client can begin loading the operating system and prepare the boot environment.

To complete the loading of the boot environment, PXE will request the necessary files from a TFTP server. TFTP allows for discrete transferring of files over a network without the authentication or filesystem browsing overhead of other protocols, such as FTP. This minimal functionality means a small memory footprint, making it ideal for a PXE environment. In order to create the boot environment, the PXE firmware for the operating system must be loaded onto the client device. The firmware, along with other boot files and the TFTP path to download additional files, is specified in the DHCP response, along with other client configuration options.

Back to the project involving my kids' laptop: While the network boot environment provided the resiliency of being able to reboot and attain a clean running operating system, I also wanted a limited environment allowing my kids to do specific tasks like send email or browse the web. Having played with TinyCore Linux in the past, I looked at it first. Its small memory footprint made it incredibly desirable for the PXE environment. For a bit of context, the TinyCore ISO image for the command-line and GUI desktop environments are 11MB and 16MB respectively. While it does not measure up to a mature distribution like Debian or CentOS, it does provide a repository of popular software packages. All of the software I wanted to run (such as Thunderbird, Firefox, Gimp, and AbiWord) are all available through the TinyCore repository.

I have identified all of the high-level moving parts of this custom PXE environment. The implementation will require the installation and configuration of a DHCP server and a TFTP server. It will also involve customizing TinyCore's out-of-the-box distribution, which will involve installing the desired applications and reconfiguring the GUI. This approach will not only provide a stable and functional environment to do various tasks, but it will provide a platform to easily deploy new applications that can be delivered via PXE boot.

Preparing the Base Files

To boot my PXE environment, all of the required files will come from two places: the Syslinux Project [2] and the TinyCore ISO image [1]. The Syslinux Project contains bootloaders for Linux, including ones designed for network booting (PXELINUX) and CD-ROM booting (ISOLINUX). For now, I will focus on PXELINUX as the bootloader; ISOLINUX will be referenced later in this article. The PXE bootloader file is named `pxelinux.0`. While it is technically possible to build this file from the Syslinux source, an older version is available on the Syslinux website that contains a pre-built binary version of `pxelinux.0`, which will work nicely. To download the Syslinux package and extract the necessary file, run the following command:

```
curl -s https://mirrors.⮐
  edge.kernel.org/⮐
  pub/linux/utils/boot/syslinux/⮐
  syslinux-4.04.tar.gz | tar -zxvf ⮐
  - syslinux-4.04/core/pxelinux.0
```

This will create the `syslinux-4.04` folder in the current working directory, which in turn contains a folder named `core` and within that folder is the `pxelinux.0` firmware file. To make this file available to a PXE client, it will need to reside within the TFTP root folder, which you create with the following command:

```
mkdir -p /opt/lib/tftp/boot
```

For this example, I have set the TFTP root directory to `/opt/lib/tftp`, and the PXE firmware should be placed in the boot folder within the TFTP root.

Next, I will complete the file configuration with the TinyCore ISO image. For this project, I am opting to go with the 16MB ISO containing the GUI environment. Since the process of preparing the TinyCore specific files is quite involved, it will be divided into two steps: preparing the base operating system and adding new applications.

The initial setup of the base system requires the `/boot` folder's contents to be extracted for further customization. This can be accomplished using the following template script:

```
sudo mount ⮐
  -o loop TINYCORE.ISO /mnt/cdrom
cp -R /mnt/cdrom/boot/* ⮐
  /opt/lib/tftp/boot
```

In this script, `TINYCORE.ISO` should be replaced with the path to the TinyCore ISO image. All of the files mentioned in this section (PXE firmware, kernel, init ramdisk, and boot folder contents) will need to be located within the TFTP root. Leave the ISO image mounted as more files will be needed later in the process. Once extracted, the target folder will contain three items: `vmlinuz` (the kernel), `core.gz` (the init ramdisk), and `isolinux` (the folder containing the boot menu configuration). The kernel file will not be modified and will remain in the boot folder in the TFTP root. The init ramdisk file, `core.gz`, will be modified to contain all of the files necessary to run the GUI environment. To modify the contents of `core.gz`, it must be extracted to a folder. This can be done using the following template script:

```
mkdir /tmp/initrd-contents
cd /tmp/initrd-contents
gzip -cd /opt/lib/tftp/boot/core.gz | ⮐
  cpio -i
```

In this example, the contents of the init ramdisk are located in `/tmp/initrd-contents`. Note that the `cpio` command is available for all popular distributions, but it may need to be installed. As mentioned earlier, the whole desktop environment will be copied onto the init ramdisk. To accomplish this, the file will need to be copied from the ISO image to a folder on the init ramdisk with the following template script:

```
mkdir -p /tmp/initrd-contents/usr/local/⮐
  share/install-base
cp /mnt/cdrom/cde/optional/*tcz ⮐
  /tmp/initrd-contents/usr/local/share/⮐
  install-base
```

The files that were copied with the TCZ file extension will be installed at boot time. Since TinyCore is a single-user distribution, this can be accomplished by editing the script located in `/etc/profile` to install the files and start X windows. The following lines will be added to the end of the `/etc/profile` file in the init ramdisk:

```
tce-load ⮐
  -i /usr/local/share/install-base/*tcz
startx
```

At this point, the init ramdisk content is configured to boot into the default out-of-the-box GUI configuration. This is a great time to repackage the init ramdisk as a base boot image. This can be done using the following script template:

```
cd /tmp/initrd-contents
find . | cpio -H newc -o > ↩
  /opt/lib/tftp/boot/core
gzip -f /opt/lib/tftp/boot/core
```

Note that running this script will overwrite the init ramdisk in the boot folder under the TFTP root folder. This finalizes the creation of the init ramdisk image. To automate the process of customizing the init ramdisk the `inity.sh` scrip can be used [3]. Now that a base image has been created, the Firefox application will be added and saved as a new init ramdisk image. This will be accomplished by modifying the above commands that were added to the `/etc/profile` script to the following:

```
tce-load ↩
  -i /usr/local/share/install-base/*tcz
tce-load -w -i firefox-esr.tcz
startx
```

This modification will allow the Firefox package to be installed after the base system has been installed. It should be noted that the larger an application is the longer it will take to install and therefore boot the environment. This is now ready to be repackaged into a new init ramdisk image, which can be accomplished with the following template script:

```
cd /tmp/initrd-contents
find . | cpio -H newc -o > ↩
  /opt/lib/tftp/boot/core-firefox
gzip -f /opt/lib/tftp/boot/core-firefox
```

Note that the name of the new init ramdisk will be `core-firefox.gz`. The last step of our process is to modify the boot menu to display two boot options: the base image and the new Firefox image. The boot menu is contained in the `/opt/lib/tftp/boot/isolinux/isolinux.cfg` file. This file has three main sections separated by empty lines. The first section contains general configuration items, such as `DEFAULT`, `UI`, `PROMPT`, and so on. The second section contains configurations beginning with `MENU`, which will be used to configure the boot menu's look and feel. The third section contains multiple line-separated groups of configurations that correspond to menu items. The first line in each group of configurations should begin with a `LABEL` statement. All of the entries in the third section may be removed for this project. After removing the existing menu entries, the file will look similar to Listing 1.

There might be minor differences depending on the TinyCore version being used. Now the new PXE environments will be added to the menu. The environment with the base init ramdisk image should look like this:

```
LABEL tc
MENU LABEL TinyCore Base
KERNEL /boot/vmlinuz
INITRD /boot/core.gz
```

The Firefox environment will look the same as this base configuration with the exception of the `INITRD` configuration, which will have a value of `/boot/core-firefox.gz`. This completes the configuration of the available PXE environments into which you can boot. The remaining step involves configuring the network services that power the PXE boot environments: DHCP and TFTP.

Configuring DHCP and TFTP

Not wanting to set up a whole new server for DCHP and TFTP, I opted to use a Raspberry Pi I had on my network to host them. Luckily for this project there is a single application that will run both of these services: dnsmasq [4]. Dnsmasq is a server application that provides, among other services, DHCP, DNS, and TFTP. Since services are being delivered over the network, the server OS distribution is not important. Luckily dnsmasq is available for most popular Linux distributions. The first step is configuring the DHCP server portion of dnsmasq. Enter a basic DHCP configuration into the `/etc/dnsmasq.conf` file as shown in Listing 2.

Line 1 of Listing 2 uses the `interface` directive to set the network interface on which the DHCP service will be listening (in this case, `eth0`). Line 2 uses the `dhcp-range` directive to specify the range of IP addresses that will be available for client use and the DHCP lease length. In

Listing 1: Modified Boot Menu

```
DEFAULT tc
UI menu.c32
PROMPT 0
TIMEOUT 600
ONTIMEOUT tc
F1 f1
F2 f2
F3 f3
F4 f4

MENU TITLE TinyCore
MENU MARGIN 10
MENU VSHIFT 5
MENU ROWS 5
MENU TABMSGROW 14
MENU TABMSG Press ENTER to boot, TAB to edit, or press F1
            for more information.
MENU HELPMSGROW 15
MENU HELPMSGENDROW -3
MENU AUTOBOOT BIOS default device boot in # second{,s}...
```

Listing 2: A Basic DHCP Configuration

```
01 interface=eth0
02 dhcp-range=192.168.1.100,192.168.1.200,12h
03 dhcp-option=3,192.168.1.1
```

Listing 3: Configuring Custom DHCP Parameters

```
01 dhcp-boot=/opt/lib/tftp/boot/pxelinux.0,192.168.1.10
02 dhcp-option-force=209,/boot/isolinux/isolinux.cfg
03 dhcp-option-force=210,/
04 dhcp-option-force=66,192,168.1.10
05 enable-tftp
06 tftp-root=/opt/lib/tftp
```

this case, the IP addresses between 192.168.1.100 and 192.168.1.200 will be available and will be leased for 12 hours. Line 3 uses the `dhcp-option` directive to set a DHCP option 3, which corresponds to the default gateway address (192.168.1.1). Once this has been completed, you can configure the custom DHCP parameters to enable PXE boot. This will involve adding Listing 3 to the `/etc/dnsmasq.conf` file.

Line 1 of Listing 3 uses the `dhcp-boot` directive to specify the full path of the PXE firmware on the TFTP server along with the TFTP server IP address. The next three lines are setting specific DHCP options. Option `209` on line 2 is the relative path (within the TFTP root) to the PXELINUX configuration file. Since the configuration file was copied from the TinyCore ISO image, it is named `isolinux.cfg`, which corresponds to the Syslinux type that was used to boot the CD. Since both types are based on Syslinux, this will not be an issue for this project. Option `210` on line 3 sets the directory to look for PXELINUX boot files (the kernel, the init ramdisk, etc.), which like option `209` is relative to the TFTP root path. Option `66` on line 4 sets the TFTP server's IP address. All of these options will be returned in the DHCP response packet. Lines 5 and 6 are configuring the TFTP server. Line 5 uses the `enable-tftp` directive to enable listening for TFTP connections. Line 6 uses the `tftp-root` directive to specify the full path where files that will be available over TFTP will be located.

The PXE environment configuration is now complete and is ready to boot clients over a LAN. If your network already uses a DHCP server, this could cause problems as there is no assurance which DHCP server will respond to the client first. Consequently, this configuration should be deployed either on a network segment where DHCP is currently not running or configured to work with the existing DHCP service. When I implemented this solution, I was running DD-WRT on my router, which uses dnsmasq as its DNS and DHCP. So, I added the advanced DHCP options above to my DD-WRT instance and used my Raspberry Pi as the TFTP server only.

Retro PXE Functionality

Another really cool feature that can be implemented with my PXE environment is the ability to boot legacy raw disk images over PXE. This is accomplished using another component available within the Syslinux package: MEMDISK. Within the Syslinux archive that was downloaded earlier, this file is located in the `/syslinux-4.04/memdisk` folder. To enable booting a floppy disk over the network, there are three steps. First, the MEMDISK file must be placed in the TFTP root (preferably in the boot folder). Second, the floppy disk image to be booted must be placed in the TFTP server root. A folder off of the TFTP root named `floppy` could be created, and the disk image placed there. I would encourage playing with MenuetOS [5], a floppy disk OS with a GUI and many fun applications. Finally, a menu item must be configured to enable booting of the floppy disk. The menu item should look like this:

```
LABEL menuet
MENU LABEL MenuetOS
KERNEL /boot/memdisk
INITRD /floppy/menuet.img
```

Note that the MEMDISK file is referenced as the kernel to be used and the floppy disk image is referenced as the init ramdisk. This could also be configured to boot a larger disk image; however, this could impact load time and possibly performance.

Conclusion

This project was a success. My kids enjoy being able to use the Internet, email grandma and grandpa, and play a few old legacy games that I've put on floppy disks. I enjoy not having to re-image the laptop when it gets shut down incorrectly. The system's resilience is a great asset, plus it's an excellent solution for breathing life into old computing hardware. ■■■

Info

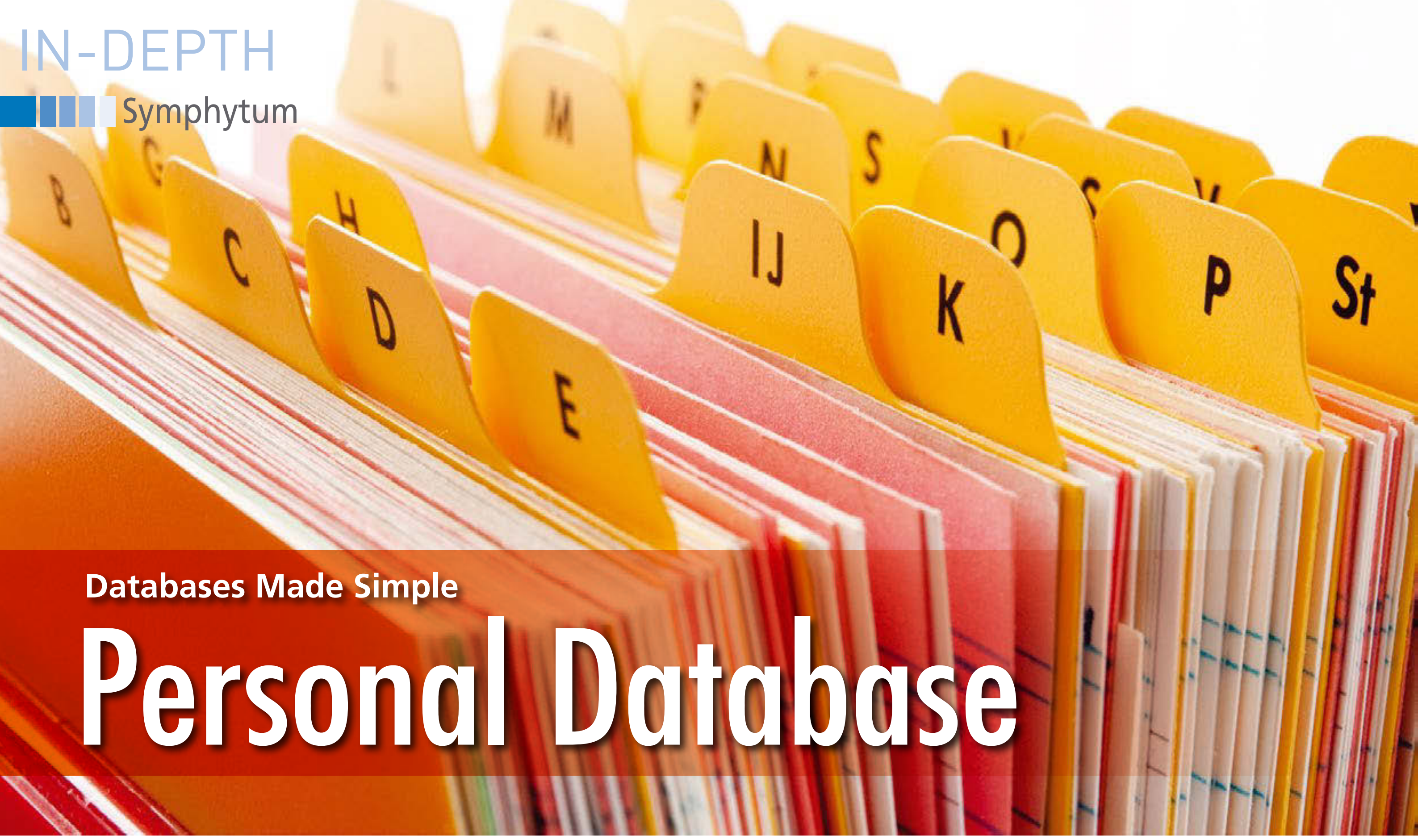
[1] TinyCore Linux: *http://www.tinycorelinux.net/*

[2] Syslinux: *https://wiki.syslinux.org/*

[3] inity.sh (manipulate init ramdisks): *http://git.andydoestech.com/git/scripts/.git/tree/shell/inity.sh*

[4] dnsmasq: *http://www.thekelleys.org.uk/dnsmasq/doc.html*

[5] MenuetOS: *http://www.menuetos.net*



Databases Made Simple

Personal Database

From a simple task list to a collection that keeps tabs on your books, Symphytum lets you quickly and easily build databases for storing and working with any type of data imaginable. *By Dmitri Popov*

You rarely need a full-blown relational database if you only want to store recipes, notes, tasks, and other simple pieces of data. What you need is a tool that makes it possible to quickly create simple, easy-to-use, database-powered applications with a minimum of effort and technical knowledge. Enter Symphytum [1]. You'll be hard-pressed to find a more straightforward and user-friendly application for creating personal databases. With absolutely no knowledge of database theory and design, you can build databases for pretty much any purpose. More importantly, the resulting databases feature a polished interface that doesn't require a degree in computer science to use.

Getting Started

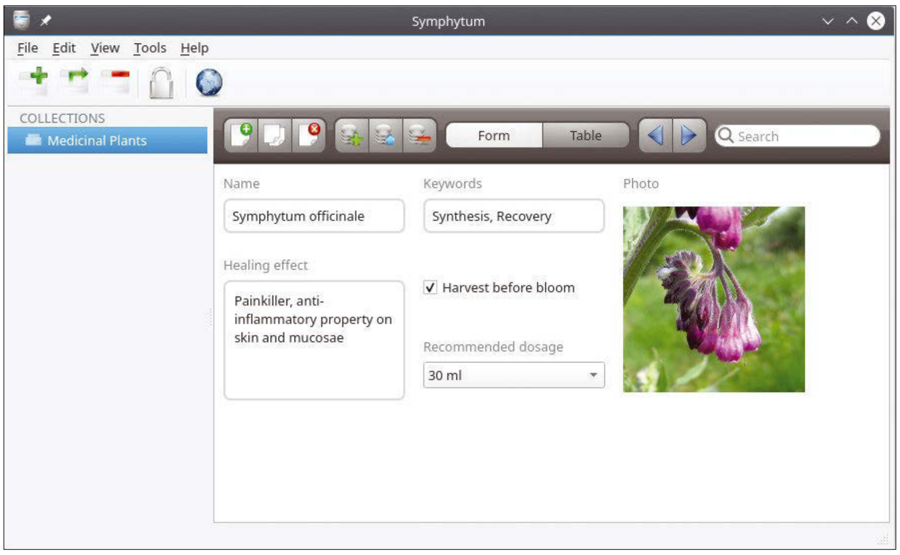
Symphytum uses SQLite as its database engine. Being lightweight, robust, and mature, SQLite is a popular choice for powering database-driven applications. However, Symphytum hides all the technical intricacies behind a user-friendly interface, so you are never exposed to the scary database underbelly.

The project's GitHub site offers packaged versions for popular Linux distributions. You'll also find an AppImage self-contained executable. It's larger than the packages for specific distributions, and it might run slightly slower on your machine. But it requires no installation and comes with all required dependencies. As such, it offers the easiest and fastest way to run Symphytum on practically any Linux system. Grab the latest AppImage file from the project's Releases page, make the downloaded file executable using the command

```
chmod +x Symphytum-x86_64.AppImage
```

and then double-click on the file to launch Symphytum.

Before you create your first database, you might want to tweak a few settings. Unlike any other regular desktop application, Symphytum doesn't create a lot of separate files. Instead, each application in Symphytum is just a separate table of the `data.db` database. This file, along with the files you attach to indi-



Name: Symphytum officinale
Keywords: Synthesis, Recovery
Healing effect: Painkiller, anti-inflammatory property on skin and mucosae
✓ Harvest before bloom
Recommended dosage: 30 ml

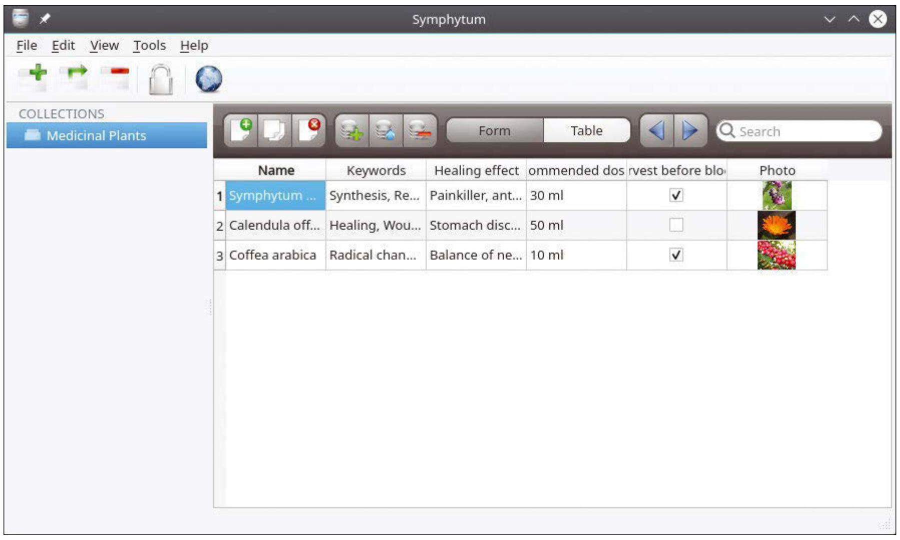
Figure 1: Symphytum in all its beautiful simplicity.

vidual records, is stored in a separate folder. By default, this folder is hidden in your home directory, and you might want to choose another location for it to make backup easier. To do this, choose *Tools* | *Preferences*, switch to the *General* section, and specify the desired path and directory name. While you are at it, you might want to change the default font and font size. Switch to the *Appearance* section and specify the font and size you want in the appropriate fields. Finally, switch to the *Advanced* section, and enable the option that improves performance by caching images. This option requires more RAM, but as it significantly improves performance, it's worth enabling in most situations.

Although Symphytum is a single-user application, it does support synchronization. With this functionality enabled, you can access your data from any other machine running Symphytum. To activate and configure the synchronization feature, switch to the *Cloud Sync* section, select *Enabled* from the *Status* drop-down list, and press *Close*. This launches a simple wizard that guides you through the process of setting up syncing. Symphytum supports the Dropbox and MEGA synchronization services. Before you can use the latter, you need to install and configure the `MEGAcmd` [2] command-line tool. If you prefer to enable and set up synchronization later, you can do it by choosing *Tools* | *Cloud synchronization* in the main toolbar. You can use any other synchronization service, too. You just have to configure your preferred cloud syncing tool to synchronize Symphytum's working directory.

Working with Collections

Databases in Symphytum are called collections, and the application comes with an example collection for you to explore. Each collection can contain a number of fields of different types and two views: *Form* and *Table*. You can use the appropriate buttons in the main toolbar to browse through the existing collection entries, as well as create, duplicate, and remove records. The *Table* view (Figure 2) allows you to sort records in ascending or descending order by clicking on the desired field heading. For example, to sort all records in the example database alphabetically, click on the *Name* field heading. Click on it again to reverse the order.



	Name	Keywords	Healing effect	ommended dos	rvest before blo	Photo
1	Symphytum ...	Synthesis, Re...	Painkiller, ant...	30 ml	✓	
2	Calendula off...	Healing, Wou...	Stomach disc...	50 ml	☐	
3	Coffea arabica	Radical chan...	Balance of ne...	10 ml	✓	

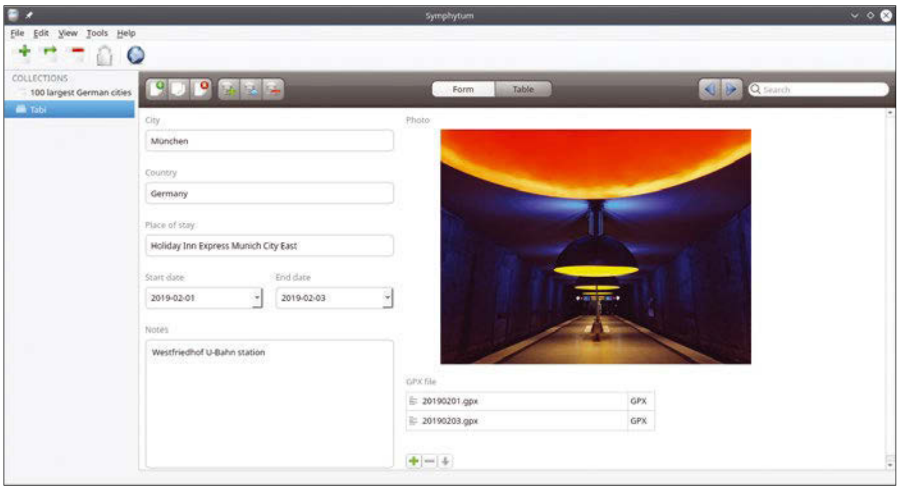
Figure 2: The *Table* view lets you view records as a list and sort them by column name.

Need to find a specific record? Start typing the desired search criteria in the *Search* field to see the list of matching records. The search feature is a one-trick pony: It searches through all available fields, and that's all it can do.

Building a collection from scratch in Symphytum is straightforward. If you travel a lot, you might want to set up a collection to keep track of the cities you have visited (Figure 3). Press the *New Collection* button in the main toolbar, or choose *File* | *New* | *New Collection* to create a new collection. Replace the default collection name with a more descriptive title, and the blank collection is ready.

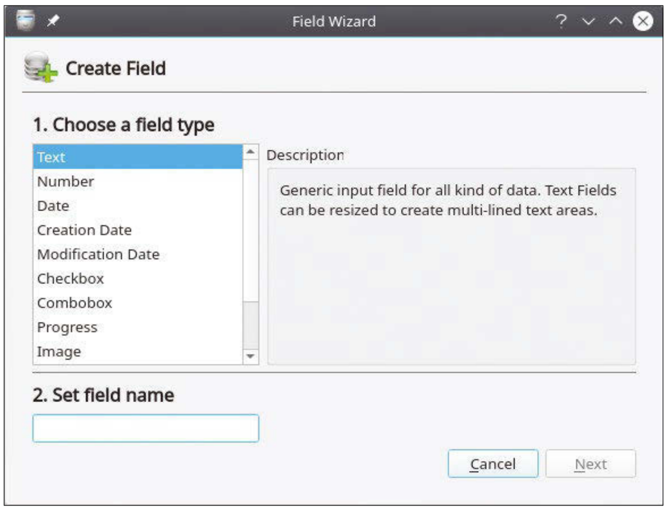
The first order of business is to add at least one field to the collection. In this case, you can start by creating a text field to store the name of the city. Press the appropriate button in the main working area, select *Text* from the list of available field types, give the field a name (it will appear as the field's label in the *Form* view), and press *Next* (Figure 4). You can enable the *Required field* option if you want to make the field mandatory (i.e., it must not be empty). Press *Finish* to add the field. Switch then to the *Form* view using the appropriate button. To adjust the field's size and position, click on the field's label to select the field. Adjust the field's width and height by dragging the selection points with the mouse. To move the field, click and hold the field's label and drag the field to the desired position. Using the *New Field* button in the main toolbar (or the Ctrl + Shift + N keyboard shortcut), you can add as many fields as you like.

With Symphytum, you are not limited to text fields. If you want to add



City: München
Country: Germany
Place of stay: Holiday Inn Express Munich City East
Start date: 2019-02-01
End date: 2019-02-03
Notes: Westfriedhof U-Bahn station
Photo
GPX file:
20190201.gpx GPX
20190203.gpx GPX

Figure 3: Despite its simplicity, Symphytum offers everything you need to build databases that accommodate practically any type of data.


Field Wizard

Create Field

1. Choose a field type

- Text
- Number
- Date
- Creation Date
- Modification Date
- Checkbox
- Combobox
- Progress
- Image

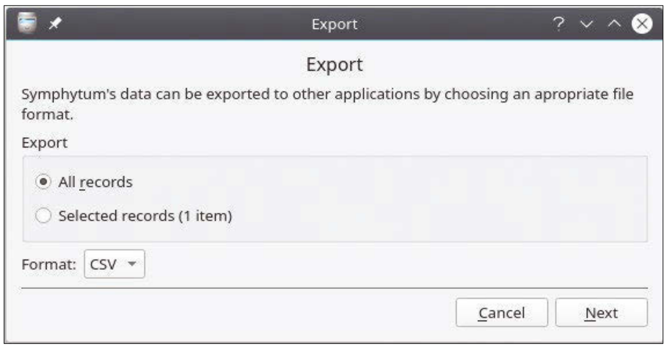
Description

Generic input field for all kind of data. Text Fields can be resized to create multi-lined text areas.

2. Set field name

Cancel | Next

Figure 4: Symphytum supports a wide range of field types.


Export

Export

Symphytum's data can be exported to other applications by choosing an apropriate file format.

Export

(●) All records
( ) Selected records (1 item)

Format: CSV

Cancel | Next

Figure 5: Symphytum can export existing data to other applications.

the dates on which you visited the city, create two date fields: one for the arrival date and another for the departure date. When adding a date field, Symphytum allows you to choose a date format. Symphytum also lets you enable the reminder option for the date field, so the application will display a reminder when the date in the date field is reached. This simple option can come in useful when working with collections containing deadlines (e.g., a simple task list). Symphytum even has a dedicated view for listing all upcoming reminders (*View* | *Date reminder list*).

Symphytum supports other field types, too. The *Image* field type makes it possible to insert images into records. Symphytum can handle most popular image formats, including PNG, JPEG, GIF, TIFF, and even vector-based SVG. Keep in mind that when you add an image to a record, Symphytum doesn't insert the image file into the record. Instead, the application saves the image file in the dedicated *files* directory inside Symphytum's working folder and then creates a reference to this file in the record. Why is this important? Because if you want to keep images in your collection safe, you have to make sure that you back up the entire content on Symphytum's working folder, including the `files` directory.

The *File List* field type allows you to add multiple files to a record. To add a file or multiple files, you can either drop them onto the *File List* field in the *Form* view or use the dedicated button below the field. The *Remove* button lets you delete individual files, while the *Export* button can come in handy when you need to save a file in the record to a local directory.

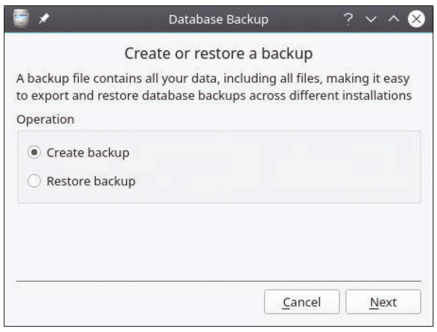
If you want to add a drop-down list containing a list of predefined entries, the *Combobox* field type is what you need. Finally, the *Checkbox* field type allows you to add options that can be toggled, while the *Progress* field type makes it possible to add an adjustable progress bar to the record. Combined, these three field types are ideal for creating collections that can help you to manage tasks and keep tabs on your projects. Add to this a *Date* field with the reminder option enabled, and you have a simple way to keep track of your deadlines.

Symphytum doesn't distinguish between the design mode (where you add fields and design forms) and the work mode (where you manage records and populate them with data). While this is convenient, this also means that you may sometimes select a field instead of focusing the cursor on it and even inadvertently resize or move the field. To prevent this from happening, Symphytum lets you lock the *Form* view using the *Lock* button in the main toolbar. There is another creature comfort: the *Tools* | *Database* | *Free unused space* command can clean up and optimize the underlying database to reduce its size.

If you already have data in the comma-separated (CSV) format, Symphytum allows you to import them and create a new collection on-the-fly. To import existing data, choose *File* | *Import* and follow the import guide. You can also export data (Figure 5) from the existing collection using *File* | *Export*. The import and export functionality in Symphytum does have a serious limitation: It cannot import and export files and images (since they are stored as references and not inserted directly into records). If you need to move data from one Symphytum instance to another, you can use the backup and restore feature instead (Figure 6). Choose *File* | *Backup* and follow the guide to export Symphytum data into a single `.syb` file. You can then import it into Symphytum running on another machine via *File* | *Backup* and choosing the restore option.

Conclusion

Symphytum won't rival advanced database applications like LibreOffice Base and Kexi. But what it lacks in functionality, it makes up for in ease of use. And honestly, unless you are building complex relational databases, you'll hardly ever need more than what Symphytum has to offer. ■■■


Database Backup

Create or restore a backup

A backup file contains all your data, including all files, making it easy to export and restore database backups across different installations

Operation

(●) Create backup
( ) Restore backup

Cancel | Next

Figure 6: You can use Symphytum's backup and restore functionality to keep your data safe.

Info

[1] Symphytum: *https://github.com/giowck/symphytum*

[2] MEGAcmd: *https://mega.nz/cmd*

The sys admin's daily grind: urlwatch

What's New Pussycat?

Experienced system administrators attach great importance to always being up to date when it comes to information technology. Urlwatch is a command-line tool that presents the latest news from websites that do not offer RSS feeds by email. *By Charly Kühnast*

Some years ago, I reported on the Miniflux RSS feed aggregator [1] in this column, and I still use it. Miniflux is lean, fast, and easy to use. The media have been predicting the death of RSS feeds for what feels like an eternity, but it still has not happened. However, some websites simply don't offer feeds. I have to do something different here. What I need is a tool that alerts me when a particular website changes.

That's far more complicated than it sounds at first. Using a web service for this purpose (there are countless numbers of them) is out of the question for reasons of data economy. However, the biggest problem is something else. Things on websites that I don't find relevant are constantly changing. For example, a daily newspaper that I regularly read displays job ads that change with every reload. If I was notified every time, it would drive me crazy. There has to be a better way – and there is: urlwatch.

Written in Python 3, urlwatch is included in most popular distributions. If you want to know exactly if and which version of urlwatch is available in your favorite distro, you can find out on Repology.org [2]; DIY enthusiasts can use GitHub [3]. For example, I instructed urlwatch to keep an eye on the online news page for *Linux Magazine* (Listing 1) – pointlessly, because it actually still offers RSS.

Listing 1: Monitoring Websites

```
$ urlwatch --add url=http://www.linux-magazine.com/Online/News,name=LinMag
```

In my home directory below `.config/urlwatch/`, urlwatch has now created a configuration file named `urls.yaml` (Listing 2). So far, so good – but I want to be sure that I am only notified when a new news ticker post is published. I want urlwatch to ignore any other changes to the website.

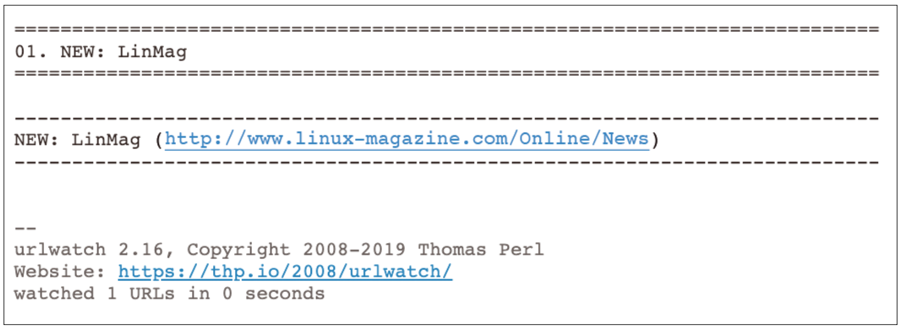
Listing 2: urls.yaml

```
kind: url
name: LinMag
url: http://www.linux-magazine.com/Online/News
```

To do this, I briefly immersed myself in the website's HTML source code and discovered that every news ticker entry is introduced by a `div` tag that contains the specification `"td_module_9 td_module_wrap"` as its `class` specification. This prompted me to append the following line to the configuration in `urls.yaml`:

```
filter: element-by-id:td_module_9
```

Now urlwatch only alerts me when a new ticker entry appears on the website. There is a rich selection of alerting types; I opted for plain old email. All I have to do is enter the appropriate mail server, as well as the sender and recipient addresses, below `.config/urlwatch/` in the `urlwatch.yaml` configuration file's `report` section. And, lo and behold, the electronic mailman was soon ringing the bell (Figure 1). ■■■



```
===========================================================================
01. NEW: LinMag
===========================================================================

---------------------------------------------------------------------------
NEW: LinMag (http://www.linux-magazine.com/Online/News)
---------------------------------------------------------------------------


--
urlwatch 2.16, Copyright 2008-2019 Thomas Perl
Website: https://thp.io/2008/urlwatch/
watched 1 URLs in 0 seconds
```

Figure 1: Urlwatch immediately notifies Charly by email when news is posted on his favorite websites.

Info

[1] "The Sys Admin's Daily Grind: Miniflux" by Charly Kühnast, *Linux Magazine*, issue 164, July 2014, *http://www.linux-magazine.com/Issues/2014/164/Charly-s-Column-Miniflux/(language)/eng-US*

[2] urlwatch packages: *https://repology.org/project/urlwatch/versions*

[3] urlwatch on GitHub: *https://github.com/thp/urlwatch*

Author

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

Flatpak integration with desktop systems

The Future of Flatpak

Flatpak's development may have been prompted by container development, but its future depends on the desktop. *By Ferdinand Thommes*

Alternative systems for distributing software are on the rise. In 2018, everyone was talking about Flatpak and Snap. In addition, AppImage, which does not require any basic installation, offers completely self-sufficient packages.

Flatpak [1], developed by Fedora under the leadership of Alexander Larsson, is compatible with desktop applications. It lets you package software so that the same package works on all distributions. The only requirement is a matching Flatpak runtime environment.

Origins

Flatpak, which reached version 1.0 in mid-2018, is now considered mature enough for production use. The current version is 1.50. The origin of the format dates back to 2007, when Red Hat employee Larsson was experimenting with klik [2], AppImage's predecessor. However, he did not like some of the technical details. In the same year, he released Glick, which was based on FUSE [3], due to the lack of container APIs, which had not yet been invented.

Glick 2 relied on the newly introduced kernel namespaces in 2011, which again aroused Larsson's interest in alternative packaging. Larsson published a long article on his blog explaining why he did not consider the existing packaging systems to be ideal and why he preferred bundling software. This early article already outlined the basics of OSTree [4], Atomic Host [5], and Silverblue [6].

Around 2013, kernel support for containers evolved, and Docker was launched. Larsson's task was to get Red Hat Enterprise Linux (RHEL) ready for Docker. At a Gnome hackfest in the same year, more concrete ideas on runtime, sandboxing, and the modules known as portals for controlled access to the actual system's resources were developed.

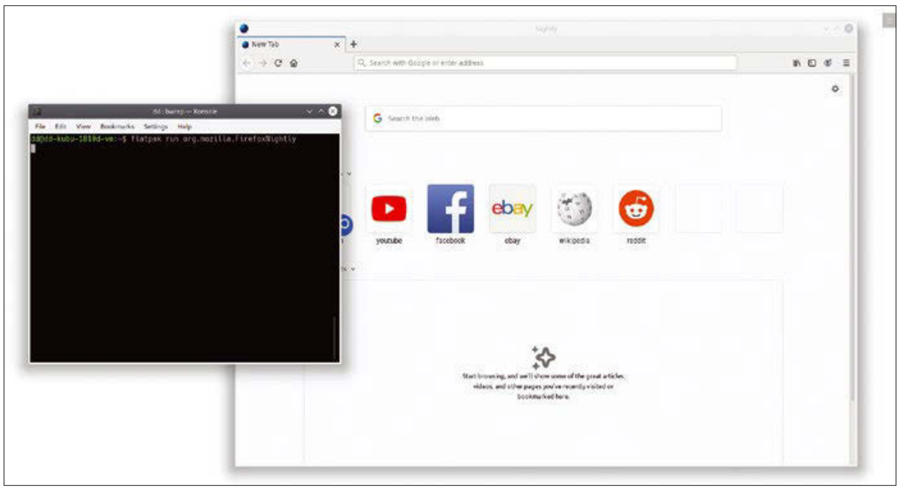
Lennart Poettering and Greg Kroah-Hartman joined Larsson in the discussion, which led to a manifesto [7], resulting in the birth of Flatpak (whose name derives from IKEA's method of flattening DIY furniture).

The project initially was dubbed xdg-app before being renamed Flatpak. This forerunner already used OSTree to download, store, and deduplicate applications. Kernel namespaces were also used to execute unprivileged containers.

Container DNA

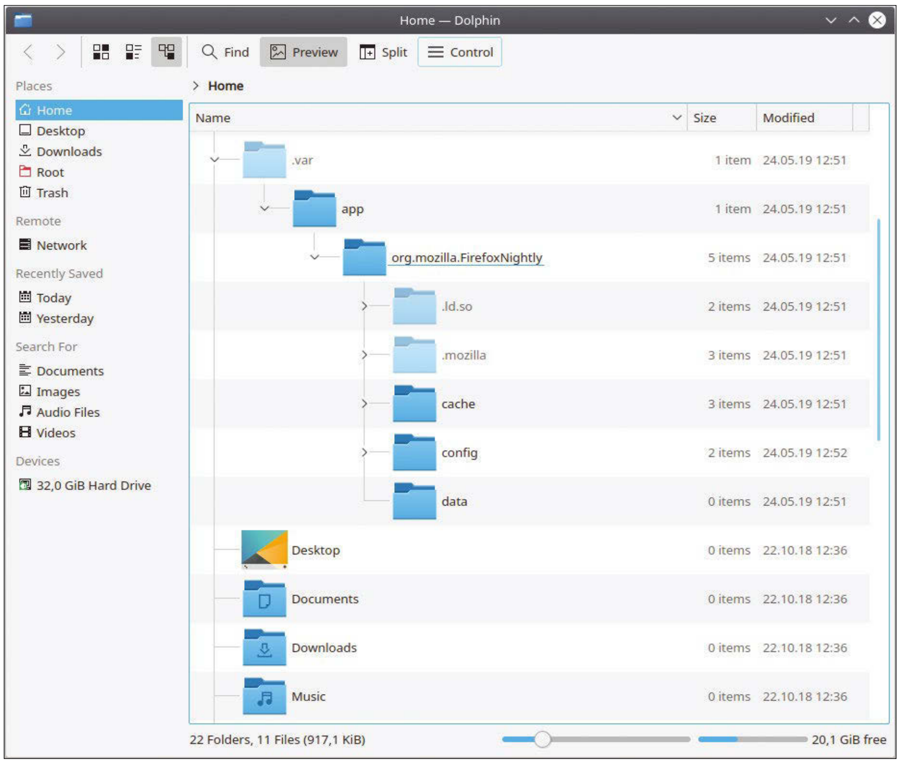
Flatpak uses the same building blocks and mechanisms that are used in container systems such as Docker or LXC. This includes the previously mentioned function of kernel namespaces. However, the basis for Flatpaks is one or more runtime environments that provide basic functions for the Flatpaks via libraries and interpreters. In contrast to the container formats mentioned earlier, Flatpak sandboxes are unprivileged; they do not need root.

Bubblewrap [8] is used to run Flatpak applications in the user context. In principle, the software works like a `chroot` [9],



```
dd@dd-kubu-1810d-vm:~$ flatpak run org.mozilla.FirefoxNightly
```

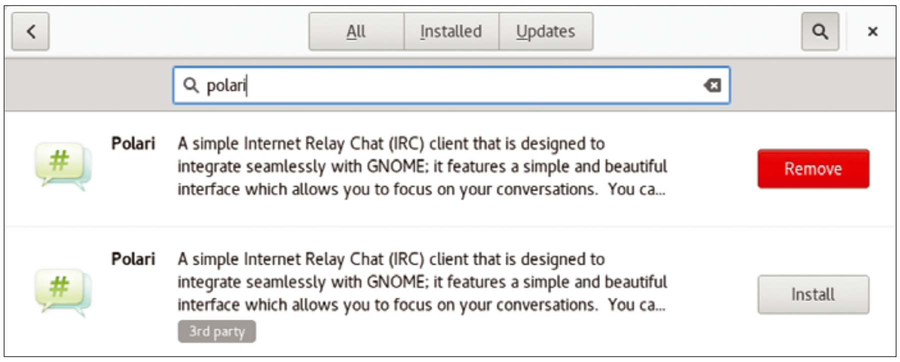
Figure 1: In addition to Firefox installed via the filesystem, you can install a nightly build of the browser that uses its own profile. It is installed in the user context here.



Name	Size	Modified
.var	1 item	24.05.19 12:51
app	1 item	24.05.19 12:51
org.mozilla.FirefoxNightly	5 items	24.05.19 12:51
.ld.so	2 items	24.05.19 12:51
.mozilla	3 items	24.05.19 12:51
cache	3 items	24.05.19 12:51
config	2 items	24.05.19 12:52
data	0 items	24.05.19 12:51
Desktop	0 items	22.10.18 12:36
Documents	0 items	22.10.18 12:36
Downloads	0 items	22.10.18 12:36
Music	0 items	22.10.18 12:36

22 Folders, 11 Files (917,1 KiB) 20,1 GiB free

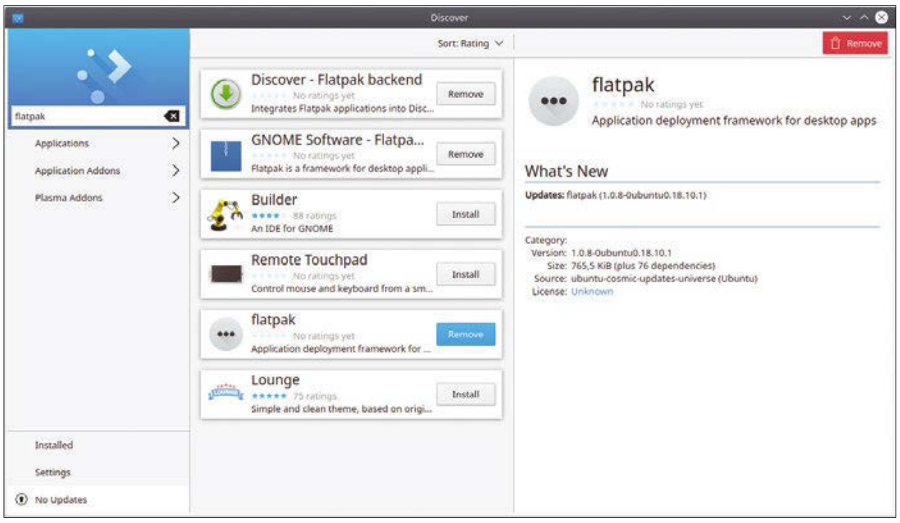
Figure 2: Apps installed in the user context are connected to the system via the hidden `.var/app/` folder in the home directory.



Polari A simple Internet Relay Chat (IRC) client that is designed to integrate seamlessly with GNOME; it features a simple and beautiful interface which allows you to focus on your conversations. You ca... Remove

Polari A simple Internet Relay Chat (IRC) client that is designed to integrate seamlessly with GNOME; it features a simple and beautiful interface which allows you to focus on your conversations. You ca... 3rd party Install

Figure 3: In Gnome Software, Fedora offers several apps in native format and as Flatpaks. For example, Gnome tags the Polari Flatpak as *3rd party*.



Discover - Flatpak backend — No ratings yet — Integrates Flatpak applications into Disc... — Remove
GNOME Software - Flatpa... — No ratings yet — Flatpak is a framework for desktop appli... — Remove
Builder — 88 ratings — An IDE for GNOME — Install
Remote Touchpad — No ratings yet — Control mouse and keyboard from a sm... — Install
flatpak — No ratings yet — Application deployment framework for ... — Remove
Lounge — 75 ratings — Simple and clean theme, based on origi... — Install

flatpak
No ratings yet
Application deployment framework for desktop apps

What's New
Updates: flatpak (1.0.8-0ubuntu0.18.10.1)

Category:
Version: 1.0.8-0ubuntu0.18.10.1
Size: 765,5 KiB (plus 76 dependencies)
Source: ubuntu-cosmic-updates-universe (Ubuntu)
License: Unknown

Figure 4: In addition to Flatpak apps, Plasma Discover offers the basic Flatpak components for installation.

but relies on unprivileged user namespaces [10]. A process flag also prevents the software from being granted new privileges that might allow it to break out of the sandbox.

On top of this, the developers secure the sandbox with seccomp [11]. In this way, they try to prevent potentially risky system calls from reaching the outside world. By default, the application in the sandbox is only able to write to some of the home directory's subdirectories.

Installing Software

Flatpak is now preinstalled on many distributions, with the exception of Ubuntu, which uses Snap to propagate its own variant of an alternative package system. Flatpaks are well integrated from the user's point of view: In addition to tools for the command line (Figure 1), there are now management applications for desktop environments. Software can be installed in this way in the user or system context (Figure 2).

This is at least true of Gnome (Figure 3) and KDE (Figure 4). Applications for package management such as Gnome Software or Plasma Discover integrate Flatpaks and automatically display existing updates. However, there is still room for improvement here.

Flathub [12] is a central repository with currently more than 400 packages. Developers can post their apps here, and users can install them with a single click (Figure 5).

Since the principle of repositories is already anchored in Flatpak's source code, creating and offering your own archives in this format requires very little overhead [13].

Developer Criticism

The desire for a uniform package delivery system under Linux is not new. For a long time, criticism (mainly from developers) has been levied at the distributions' conventional approaches. A package format for all distributions would result in faster delivery of developments to users and thus accelerate the work on many programs in general.

The army of critics even includes Linus Torvalds, who took a stand on the subject at a Debian conference [14]. Despite the desire for a uniform package management system, there is criticism of Flatpak from several sides, which also

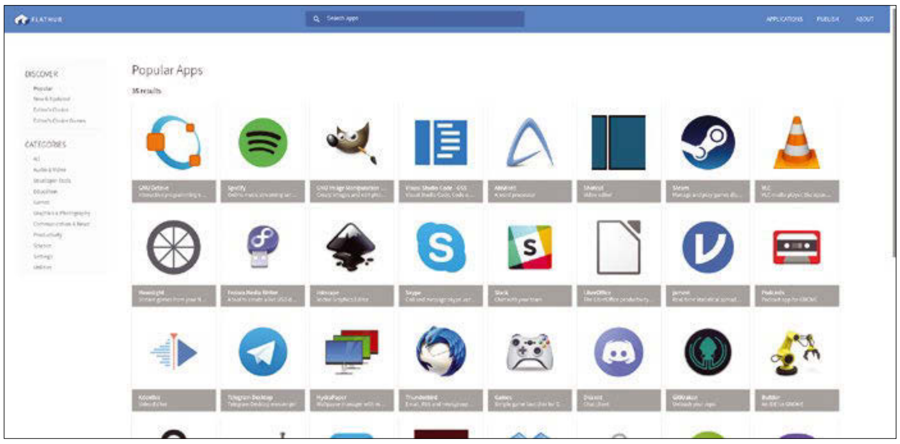


Figure 5: Flathub now offers over 400 applications.

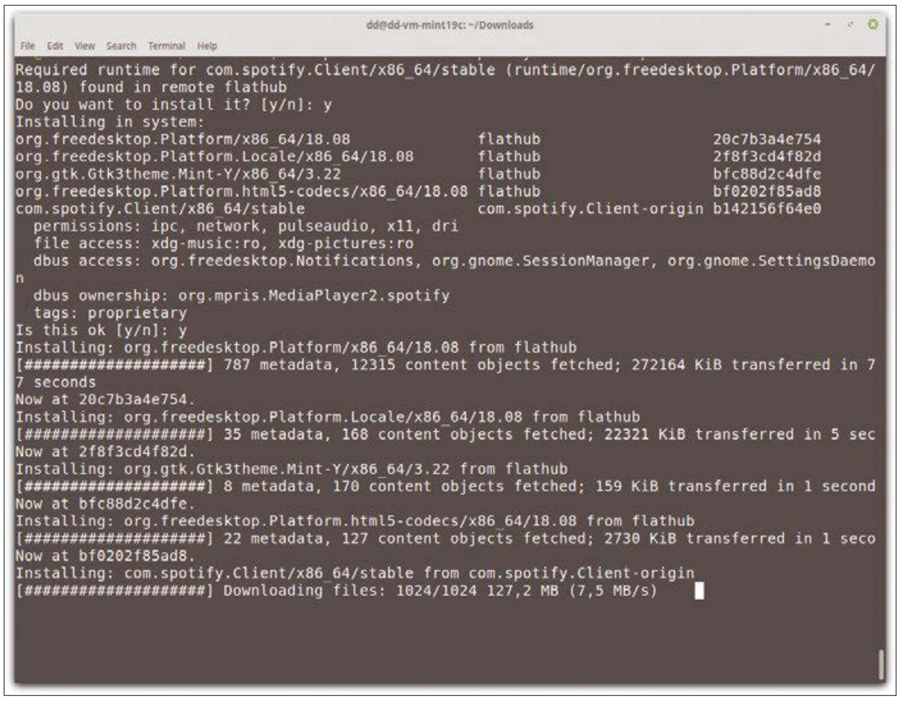
generally applies to other alternative package management systems.

Deduplication

From the beginning, Flatpak and similar approaches were thought to waste too much space on hard disks by duplicating runtime environments and libraries. This accusation is not easy to deny, because setting up a package as a Flatpak, Snap, or AppImage often requires a download of several hundred megabytes, whereas the distribution maintainer's version may just be a few kilobytes or megabytes (Figure 6).

This is due to its universal applicability: Flatpak bundles all necessary additional programs and installs them with the actual software. This is particularly noticeable if a bug occurs in a library that is used in many programs: If not every Flatpak maintainer exchanges this version, a faulty version remains on the system. The problem does not arise with the maintainer version, since the distributor replaces the library.

Recently, the situation has improved slightly, because the *libostree* library now lets you deduplicate. As for hard disk space, it is cheaper today than ever before. If you adhere to a minimalist approach, Flatpak is hardly the right choice for you anyway.


```
dd@dd-vm-mint19c: ~/Downloads
File  Edit  View  Search  Terminal  Help
Required runtime for com.spotify.Client/x86_64/stable (runtime/org.freedesktop.Platform/x86_64/
18.08) found in remote flathub
Do you want to install it? [y/n]: y
Installing in system:
org.freedesktop.Platform/x86_64/18.08              flathub                      20c7b3a4e754
org.freedesktop.Platform.Locale/x86_64/18.08       flathub                      2f8f3cd4f82d
org.gtk.Gtk3theme.Mint-Y/x86_64/3.22               flathub                      bfc88d2c4dfe
org.freedesktop.Platform.html5-codecs/x86_64/18.08 flathub                      bf0202f85ad8
com.spotify.Client/x86_64/stable                   com.spotify.Client-origin b142156f64e0
  permissions: ipc, network, pulseaudio, x11, dri
  file access: xdg-music:ro, xdg-pictures:ro
  dbus access: org.freedesktop.Notifications, org.gnome.SessionManager, org.gnome.SettingsDaemo
n
  dbus ownership: org.mpris.MediaPlayer2.spotify
  tags: proprietary
Is this ok [y/n]: y
Installing: org.freedesktop.Platform/x86_64/18.08 from flathub
[####################] 787 metadata, 12315 content objects fetched; 272164 KiB transferred in 7
7 seconds
Now at 20c7b3a4e754.
Installing: org.freedesktop.Platform.Locale/x86_64/18.08 from flathub
[####################] 35 metadata, 168 content objects fetched; 22321 KiB transferred in 5 sec
Now at 2f8f3cd4f82d.
Installing: org.gtk.Gtk3theme.Mint-Y/x86_64/3.22 from flathub
[####################] 8 metadata, 170 content objects fetched; 159 KiB transferred in 1 second
Now at bfc88d2c4dfe.
Installing: org.freedesktop.Platform.html5-codecs/x86_64/18.08 from flathub
[####################] 22 metadata, 127 content objects fetched; 2730 KiB transferred in 1 seco
Now at bf0202f85ad8.
Installing: com.spotify.Client/x86_64/stable from com.spotify.Client-origin
[####################] Downloading files: 1024/1024 127,2 MB (7,5 MB/s)
```
Figure 6: Installing the Spotify client as a Flatpak means downloading more than 300MB including a runtime. The package size in DEB format is around 40MB.

Package Maintainers at Risk

Another point of criticism is that application developers should not decide alone what is delivered with their package. This task is currently handled by the respective distribution's package maintainer, who adapts the application to the system's needs and guidelines. The maintainer also serves as a contact and intermediary for both developers and users.

If there were only alternative package systems like Flatpak, the role of maintainers would be obsolete. Kyle Keen, who works as a maintainer at Arch Linux, described this dilemma in his much-acclaimed article "Maintainers Matter" [15] in 2016. Basically, distributions are already barely able to take on staff; letting a maintainer handle packaging would save resources.

Sandbox Security

Last year, *Flatkill.org* [16] caused a sensation for a short time. It tried to demonstrate, in the style of the well-known systemd criticism, that Flatpak is a nightmare when it comes to sandbox security. The critics attacked the way Flatpak handles permissions. However, with a few exceptions, the accusations no longer applied by the time the criticism was published.

The criticism was directed against apps based on GTK2. Applications based on GTK3 and Qt 5 use the previously mentioned portals [17] for D-Bus-based access to the filesystem and other resources, such as printers, from inside the sandbox (Figure 7).

Since the stable version 1.0, Flatpak has seen additional improvements. Noteworthy are support for multiple Nvidia devices, the introduction of the username `flatpak` and of a custom fuse filesystem to enhance security in the home context. Flatpak can also handle webcams through the new Screencast portal, which makes use of Pipewire. Overall, Flatpak offers better control over the lifecycle of individual versions and an improved platform for regression testing.

Moving forward, major versions will appear every three months, supplemented by snapshots in between releases. For a deeper understanding of Flatpak's technical background, see Larsson's presentation from the All Systems Go conference in Berlin in September 2018 [18].

The Middle Ground

Flatpak and other alternative systems have found their way into the Linux infrastructure and are not likely to quickly disappear. Each of the approaches received both praise and criticism: Some critics see the demise of Linux coming; some proponents wish that distributions could predominantly consist of Flatpaks or Snaps. Fedora is currently implementing this in the Silverblue project, for example. As is so often the case, the middle ground makes the most sense.

Flatpaks offer advantages – and, depending upon your point of view, also cause disadvantages. The advantages are especially important for stable distributions and LTS versions. While, for reasons of stability, only older versions of software are available on LTS, Flatpak offers users the option of installing current software in a way that does not clash with the version provided by the actual package manager.

If you run several distributions in parallel, you only need to download a Flatpak once to use the software everywhere. Flatpaks are therefore completely independent of the distribution update cycle.

Conclusions and Outlook

Flatpak has arrived on the desktop with varying responses. Developers use Flatpak to serve all distributions with a single package. Fedora is enthusiastic about the new format and sees it as the future of distribution.

An informal survey of friends and family shows that Flatpaks are used moderately by some advocates, with the number of applications rarely exceeding a dozen.

As for the future, Flatpak will only live as long as the desktop does. If the influence of web apps continues to increase, the desktop's function may at some point be mainly to launch the browser. And that would probably be the end of Flatpak. ■■■

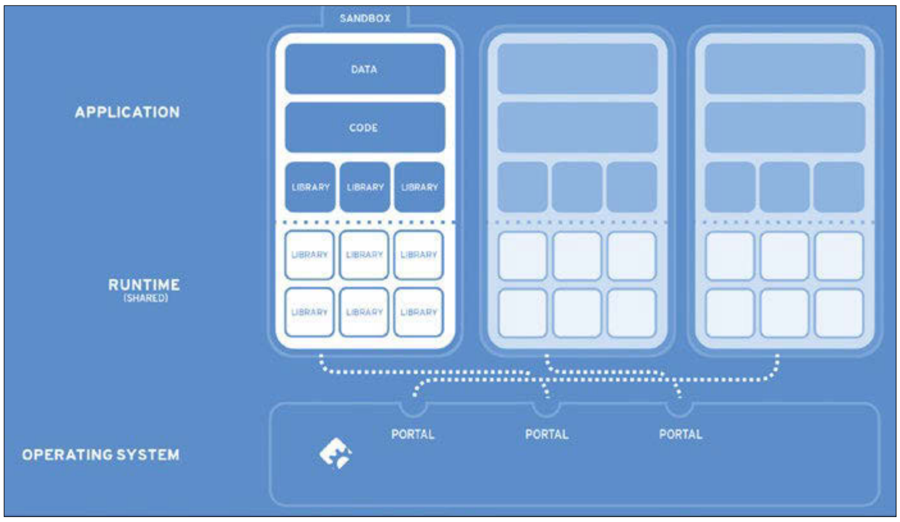


Figure 7: Portals form the interface between the app in the sandbox and the underlying system. They ensure two-way communication via D-Bus and regulate access to resources.

Info

[1] Flatpak: *https://github.com/flatpak/flatpak/releases*

[2] Klik: *https://en.wikipedia.org/wiki/AppImage#klik*

[3] Glick: *https://people.gnome.org/~alexl/glick/*

[4] OSTree: *https://ostree.readthedocs.io/en/latest/manual/introduction/*

[5] Atomic: *https://www.projectatomic.io*

[6] Silverblue: *https://silverblue.fedoraproject.org*

[7] Manifesto: *https://docs.google.com/document/d/1QTgxakyUVFMkvr-xFY2Xg9lYjcJLd6kPTI3lj5_dL7Q/edit*

[8] Bubblewrap: *https://github.com/projectatomic/bubblewrap*

[9] chroot: *https://en.wikipedia.org/wiki/Chroot*

[10] User namespaces: *https://lwn.net/Articles/532593/*

[11] Seccomp: *https://en.wikipedia.org/wiki/Seccomp*

[12] Flathub: *https://flathub.org/home*

[13] Hosting a repository: *http://docs.flatpak.org/en/latest/hosting-a-repository.html*

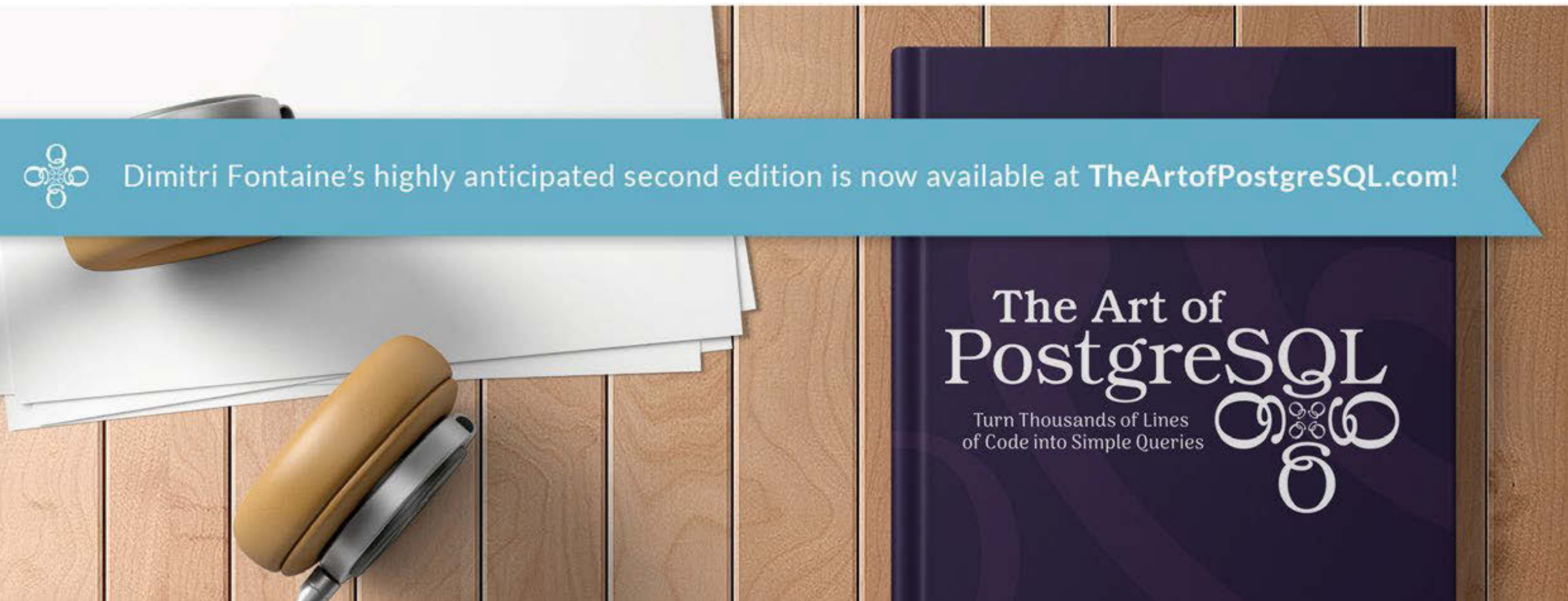
[14] Torvalds' criticism: *https://www.reddit.com/r/programming/comments/47z3kx/linus_torvalds_on_linux_application_packaging/*

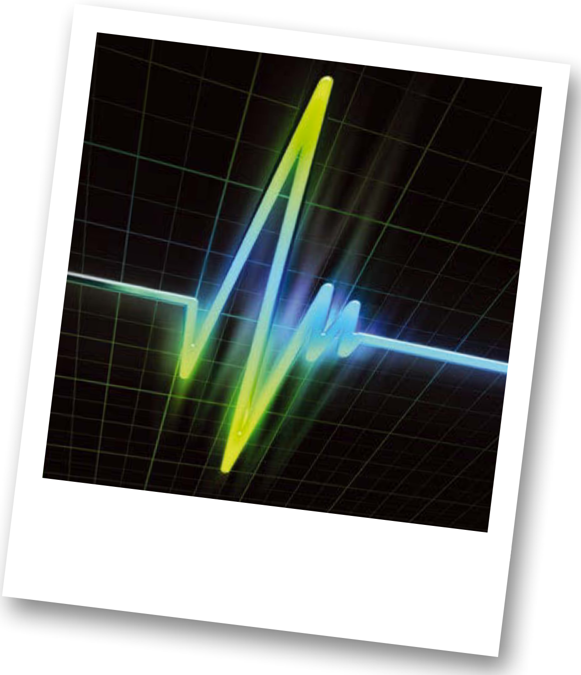
[15] "Maintainers Matter": *http://kmkeen.com/maintainers-matter/2016-06-15-11-51-16-472.html*

[16] Flatkill: *http://flatkill.org*

[17] Portals: *https://github.com/flatpak/flatpak/wiki/Portals*

[18] Presentation from All Systems Go: *https://www.youtube.com/watch?v=K0bkapSpzzk*


Dimitri Fontaine's highly anticipated second edition is now available at TheArtofPostgreSQL.com!
The Art of PostgreSQL
Turn Thousands of Lines of Code into Simple Queries



Watching activity in the kernel with the bpftrace tool

Open Heart Surgery

Who is constantly creating the new processes that are paralyzing the system? Which process opens the most files and how many bytes is it reading or writing? Mike Schilli pokes inside the kernel to answer these questions with bpftrace and its code probes. *By Mike Schilli*

If you are tasked with discovering the cause of a performance problem on a Linux system that has slowed down to a crawl, you will typically turn to tools such as `iostat`, `top`, or `mpstat` to see exactly what is throwing a spanner in the works [1]. Not enough RAM? Lame hard disk? CPU overloaded? Or is network throughput the bottleneck?

Although a tool like `top` shows you the running processes, it cannot detect short-lived instances that start and end again immediately. Periodically querying the process list only makes sense to visualize long-running processes.

Fortunately, the Linux kernel already contains thousands of test probes known as Kprobes and tracepoints. Users can inject code, log events, or create statistics there. One totally hot tool for doing this is bpftrace. With simple one-liners, it injects into the kernel scripts that determine in real time metrics like bytes heading off into the network or onto the hard drive, or lists which processes open or close which 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.

BPF stands for Berkeley Package Filter and testifies to the origin of the corresponding tool from the BSD world as a tracing tool for network packets. The practice of scattering probes throughout the code that are usually tacit, but run small snippets of code when triggered, proved so practical that it soon entered the Linux world as eBPF.

Once there, it lost its ties to network packets and conquered wide areas of the kernel code as a generic tracing concept. Good naming is hard work that engineers often shy away from, so the author of eBPF changed the name of his now popular work back to BPF. Of course, that complicates things for authors writing tutorials like this one, who are hard pressed to find an explanation as to why the BPF name has lost all meaning with respect to the product as it is today.

The approach of distributing dynamically deployable probes in kernel code came from the Sun world. For a long time, Solaris was the only operating system that allowed administrators to use DTrace to activate small pieces of D language code at strategic points, such as the system call entry point, and fire off counters or timers for performance analysis.

BPF on newer Linux kernels works in a similar way to DTrace, but has been rewritten (also for patent reasons). It executes instructions assigned to the probes in the BPF language, either in an interpreter or via a JIT compiler in native code, directly inside the kernel.

Status: Improving

The bpftrace programming language is very reminiscent of scripting with Unix veteran Awk, but it's still incomplete, and programmers sometimes struggle to complete even the simplest of tasks.

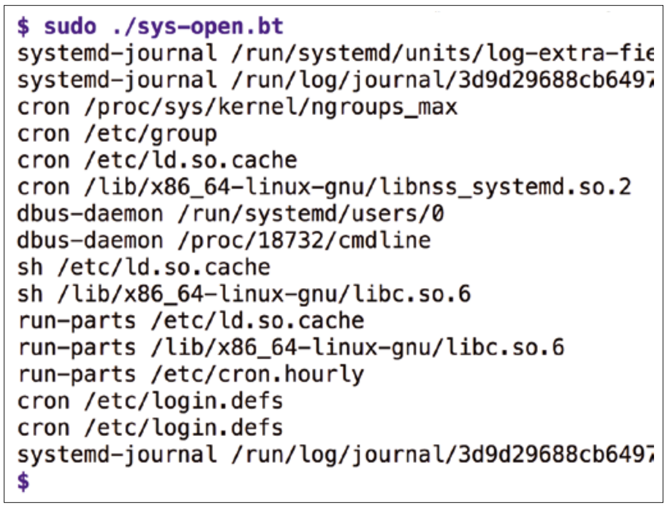
The bpftrace parser (implemented via the Unix veterans Lex and Yacc) is in a sorry state that doesn't even come close to the functionality of Awk – but maybe it will at some point. Netflix engineer Brendan Gregg and some open source friends are working on fixing it. Brendan's book on BPF [2] will be published in December 2019 (a preview is already available).

Back to the task at hand: How do you enable a probe in the kernel that outputs a message each time any userspace program calls the `open()` function to open a file? With this function, you'll be able to monitor in real time processes of active files. Turns out this is really easy to do. Listing 1 [3] shows the program code; Figure 1 shows the program output.

Compact Code

The actual work starts in line 8 with the definition of the `kprobe:do_sys_open` probe; the following block contains instructions to be executed when the probe triggers. When triggering it, the kernel tells the probe which file the `open()` system call wants to open. In the

Lead Image © Zoya Fedorova, 123RF.com



```
$ sudo ./sys-open.bt
systemd-journal /run/systemd/units/log-extra-fie
systemd-journal /run/log/journal/3d9d29688cb6497
cron /proc/sys/kernel/ngroups_max
cron /etc/group
cron /etc/ld.so.cache
cron /lib/x86_64-linux-gnu/libnss_systemd.so.2
dbus-daemon /run/systemd/users/0
dbus-daemon /proc/18732/cmdline
sh /etc/ld.so.cache
sh /lib/x86_64-linux-gnu/libc.so.6
run-parts /etc/ld.so.cache
run-parts /lib/x86_64-linux-gnu/libc.so.6
run-parts /etc/cron.hourly
cron /etc/login.defs
cron /etc/login.defs
systemd-journal /run/log/journal/3d9d29688cb6497
$
```

Figure 1: The code from Listing 1 reports in real time which processes are trying to open which files.

block, the `printf()` instruction outputs the Unix command of the triggering Unix process stored in the `comm` variable along with the first argument `arg1`, which carries the name of the file to be opened. Because `printf()` expects a string, but BPF saves `arg1` as a character pointer, the standard `str()` function converts the pointer appropriately.

The code for the `interval:s:5` event starting in line 3 is just some optional feature that cancels the program after five seconds. The event defines an interval of five seconds at which bpftrace jumps into the code block. The call to `exit()`, which shuts down the program, occurs here as soon as the block has been accessed for the first time. Tracing tools often use intervals like this to output consolidated statistics every few seconds. Once bpftrace has been installed on a Ubuntu system like this:

```
$ sudo apt-get update
$ sudo apt-get install bpftrace
```

all you need to do is run Listing 1 with `sudo`. It launches in the blink of an eye and keeps showing you which processes on the system are currently attempting to open which files. Before you get too excited, however, please note that bpftrace only works on relatively new kernels. Its creators recommend at least version 4.9, and preferably a series 5 kernel.

It is a very powerful tool. Astonished users will rub their eyes in amazement thinking about what just happened behind the scenes during the inconspicuous call: Bpftrace activated the `do_sys_open` kprobe in the kernel and translated the `printf()` statement into an internal format. It then installed the compiled code on the probe, causing it to display a message every time the kernel passes the probe. When the bpftrace call terminates, it deactivates the probe in the kernel and removes the injected code.

Full Tilt While Idle

How does this work inside the kernel? It would obviously be devastating for kernel performance if it had to check whether each probe is currently active and then carry on normally in the program in almost 100 percent of the cases when the probe is inactive. There are always very few, if any, probes active from thousands of possible ones.

Instead, the BPF technology, just like DTrace under Solaris, uses a trick: Normally, when the probe is inactive, it inserts a 5 byte no-op instruction into the code, which the processor skips with practically no impact at run time. If the user activates the probe, for example by calling bpftrace, BPF replaces the no-op instruction in the kernel with a jump address to the interpreter that executes the desired code.

No doubt, the CPU will consume time when executing the BPF instructions, which will slow down the kernel a bit. But since the processor stays in kernel mode and doesn't have to switch to user space every time, the probe can quickly refresh the desired statistics – then the flow continues with the actual kernel code.

However, if the infiltrated code were to block the kernel, the result would be devastating: The entire system would stop, which is tantamount to a computer crash. That's why BPF verifies the code before it is introduced and only inserts it if the analysis shows that it will terminate relatively quickly. This is why the bpftrace language does not offer `for` loops or similar constructs for which it cannot predict with certainty whether they will stop running in the foreseeable future.

Huge Selection

There's plenty of choice of probes in the kernel. From `vfs_read` (the function that reads bytes from disk and can pass a count to a probe), through `do_exe_cve` (for monitoring newly created Unix processes), to `trace_pagefault_reg` (which is triggered when a memory page is reloaded), users can inspect the kernel's workings at will and discover in real time what's going on and where the bottlenecks are.

Figure 2 lists the probes that bpftrace prints when called with the `-l` switch. BPF distinguishes between `kprobes`, which track important kernel functions by name, and `tracepoint` probes, which the kernel maintainers manually maintain at a slightly higher logical level and which are thus more resilient to changes in the kernel. In contrast to userspace-facing kernel APIs, the kernel's internal functions are by no means guaranteed to be stable.

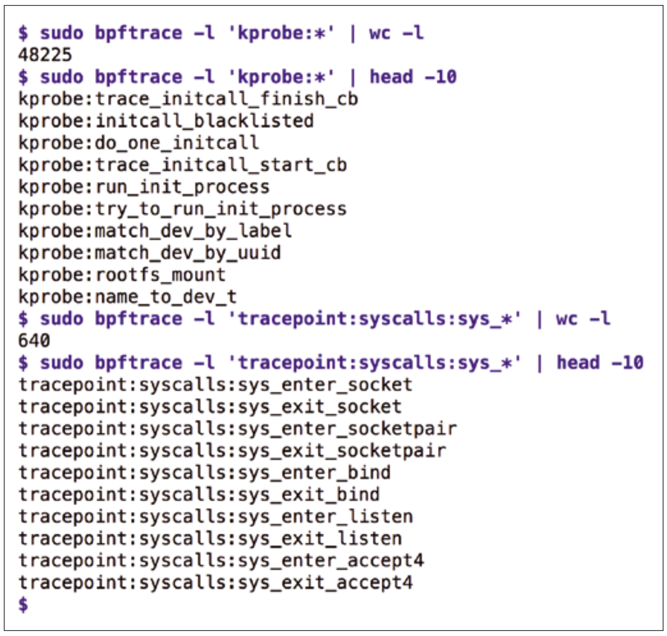
Potential for More

How about a script that outputs all newly created processes on the system in real time, including the command that was used to start them and their parameters? Listing 2 shows a one-liner that activates the `sys_enter_execve` tracepoint and prints its argument list `argv` in the `args` structure.

Here you can see that the range of functions in bpftrace still has potential for more. For example, there is the `join()` function, which uses spaces to join and

Listing 1: sys-open.bt

```
01 #!/usr/bin/bpftrace
02
03 interval:s:5
04 {
05   exit();
06 }
07
08 kprobe:do_sys_open
09 {
10   printf("%s %s\n", comm, str(arg1));
11 }
```



```
$ sudo bpftrace -l 'kprobe:*' | wc -l
48225
$ sudo bpftrace -l 'kprobe:*' | head -10
kprobe:trace_initcall_finish_cb
kprobe:initcall_blacklisted
kprobe:do_one_initcall
kprobe:trace_initcall_start_cb
kprobe:run_init_process
kprobe:try_to_run_init_process
kprobe:match_dev_by_label
kprobe:match_dev_by_uuid
kprobe:rootfs_mount
kprobe:name_to_dev_t
$ sudo bpftrace -l 'tracepoint:syscalls:sys_*' | wc -l
640
$ sudo bpftrace -l 'tracepoint:syscalls:sys_*' | head -10
tracepoint:syscalls:sys_enter_socket
tracepoint:syscalls:sys_exit_socket
tracepoint:syscalls:sys_enter_socketpair
tracepoint:syscalls:sys_exit_socketpair
tracepoint:syscalls:sys_enter_bind
tracepoint:syscalls:sys_exit_bind
tracepoint:syscalls:sys_enter_listen
tracepoint:syscalls:sys_exit_listen
tracepoint:syscalls:sys_enter_accept4
tracepoint:syscalls:sys_exit_accept4
$
```

Figure 2: Bpftrace can tap into a selection of tracepoints and Kprobes.

Listing 2: procs-new.bt

```
01 #!/usr/bin/bpftrace
02
03 BEGIN
04 {
05   printf("New processes with arguments\n");
06 }
07
08 tracepoint:syscalls:sys_enter_execve
09 {
10   join(args->argv);
11 }
```

output elements of a command line in `args->argv`. It cannot return the result as a string, however, so you could format the output with `printf()`. Hopefully, upcoming versions will resolve this issue.

The `BEGIN` block from line 3 simply provides entertainment for the user. If you want the script to display a message or initialize a variable right at startup, this happens in the `BEGIN` block as shown in Listing 2, based on the Awk programming model.

In the Thick of It

However, things become more complicated if a probe that detects a problem cannot output the desired data because it is located somewhere else. For example, to look at processes that try to open files that do not exist (or to which they have no access), Listing 3 taps into the `sys_exit_openat` tracepoint, which the kernel runs through when the `open()` system call returns.

Using the condition `args->ret < 0`, Bpftrace checks whether the return code from the system call was negative, which indicates that the desired file could not be opened. If so, we want the code to output the name of the process in question and the file name at this point. However, the `exit` tracepoint does not have access to the file name, which was only present when the kernel previously ran the `open()` function, tied to the `sys_enter_openat` tracepoint (notice the subtle difference between `enter` versus `exit`).

The solution in this case is to have bpftrace create a data structure during the `open()` call and somehow carry it over to `exit`, which then extracts the filename from it and reports the error with the desired context. For this to happen, the script stores all names of opened files in a Map type data structure when entering `open()` (i.e., in the `sys_enter_openat` tracepoint), under the key of the current kernel thread ID, which is present in the predefined `tid` variable. If the file fails to open later on, the `sys_exit_openat` tracepoint can look up the name of the file in question in the map and notify the user of this and even tell it the command of the process in `comm` that experienced the error.

The filter set in line 9 of Listing 3 is `/ @filename[tid] /`, and it ensures that the probe executes the following code if the kernel thread has previously set a file name in the map. If the call came from elsewhere than the `sys_enter_openat` tracepoint defined above, the map entry won't exist, and the filter lets bpftrace ignore the event.

After reporting the incident, the code proceeds to line 14, which calls `delete` to remove the map entry. If it forgot to do that, the map would grow indefinitely and eventually consume too much memory if the bpftrace script were to run for a longer period of time.

Listing 3: opens-failed.bt

```
01 #!/usr/bin/bpftrace
02
03 tracepoint:syscalls:sys_enter_openat
04 {
05   @filename[tid] = args->filename
06 }
07
08 tracepoint:syscalls:sys_exit_openat
09   / @filename[tid] /
10 {
11   if ( args->ret < 0 ) {
12     printf("%s %s\n", comm, str(@filename[tid]));
13   };
14   delete(@filename[tid]);
15 }
```

Who Is Writing What?

To find out which processes are writing the most data bytes to disk, Listing 4 attaches to the `sys_exit_write` tracepoint and uses the `/args->ret > 0/` filter to limit probe activity to successful writes only.

The code construct in line 6 grabs the number of written bytes from `args->ret`, saves it to the unnamed map `@` under the key for the process name, and uses `sum()` to process new values by adding them to previously existing ones.

By default, at the end of the program, bpftrace displays the content of all non-empty maps – in this case, after the user has pressed the Ctrl+C keyboard shortcut. This is why Figure 3 shows a map of all actively writing processes, as well as the number of bytes written. Alternatively, you could use an `END` block with `print(@)` to output the nameless map.

Slice by Slice

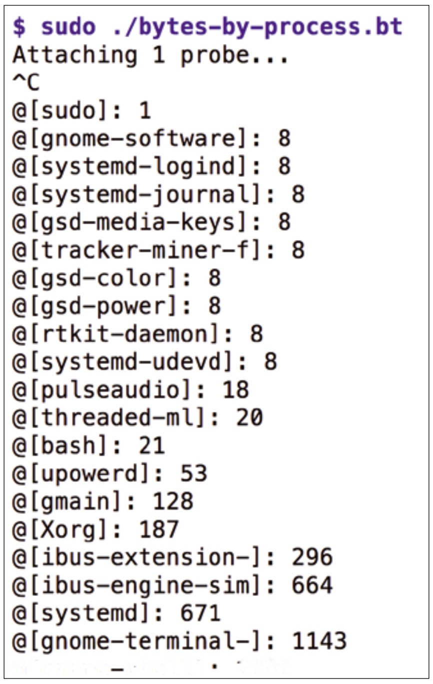
What about the average length of the data blocks being written? If you instructed bpftrace to log the length of each write operation in bytes, the result would be unreadable due to the sheer

Listing 4: bytes-by-process.bt

```
#!/usr/bin/bpftrace

tracepoint:syscalls:sys_exit_write
  /args->ret > 0/
{
  @[comm] = sum(args->ret)
}
```

volume of individual data. In addition, each time bpftrace encounters a `printf()` statement, it has to wait until the slow terminal has absorbed the written text. This takes an eternity by kernel standards. To avoid slowing down the kernel too much, in this thrashing mode, the probes occasionally let an event pass without processing, notifying the user by a `Skipped <xxx> events` message. Fortunately, BPF provides statistical processing routines at the kernel level, such as `hist()` in Listing 5.

The `hist()` function assigns the individual byte sizes of tracked write events to buckets of increasing size and prints the Map type storing variable as a histogram at the end of the program. Figure 4 shows that the length of the written data blocks varies from 1 byte to 256KB, and that the most common block lengths are between 8 and 16 bytes and 128 and 256KB. On the far left, the ASCII diagram lists the size windows, from minimum to maximum size of the bucket. In the center, it shows the number of entries in the bucket; on the right, it shows the graphical animation of the counter.



```
$ sudo ./bytes-by-process.bt
Attaching 1 probe...
^C
@[sudo]: 1
@[gnome-software]: 8
@[systemd-logind]: 8
@[systemd-journal]: 8
@[gsd-media-keys]: 8
@[tracker-miner-f]: 8
@[gsd-color]: 8
@[gsd-power]: 8
@[rtkit-daemon]: 8
@[systemd-udevd]: 8
@[pulseaudio]: 18
@[threaded-ml]: 20
@[bash]: 21
@[upowerd]: 53
@[gmain]: 128
@[Xorg]: 187
@[ibus-extension-]: 296
@[ibus-engine-sim]: 664
@[systemd]: 671
@[gnome-terminal-]: 1143
```

Figure 3: Listing 4 counts the number of bytes written for each process.

Histograms of this kind are also suitable for displaying mean values and any outliers in time measurements, you could just as well track package run times over the network, bpftrace could instead determine the time delta between an `enter-exit` tracepoint pair by buffering the current nanosecond value in the `nsecs` variable when it occurs and by calculating the difference from the current `nsecs` value at the `exit` tracepoint.

The graphical representation can be used to determine whether Service Level Agreements were met or how often the set targets were missed. If you'd like to explore this some more, all variables and commands for bpftrace can be found in its documentation [4] and in a practical cheat sheet [5].

Listing 5: writes-by-size.bt

```
#!/usr/bin/bpftrace

tracepoint:syscalls:sys_exit_write
  /args->ret > 0/
{
  @ = hist(args->ret)
}
```

With bpftrace, there's a powerful new tool available for those who are running a fairly recent kernel and are not afraid of the project's lack of polish. Even as it is right now, it pretty much outshines everything that has been done before when it comes to quickly getting to the bottom of system-related bottlenecks on Linux servers. And it'll just keep getting better! ■■■

Info

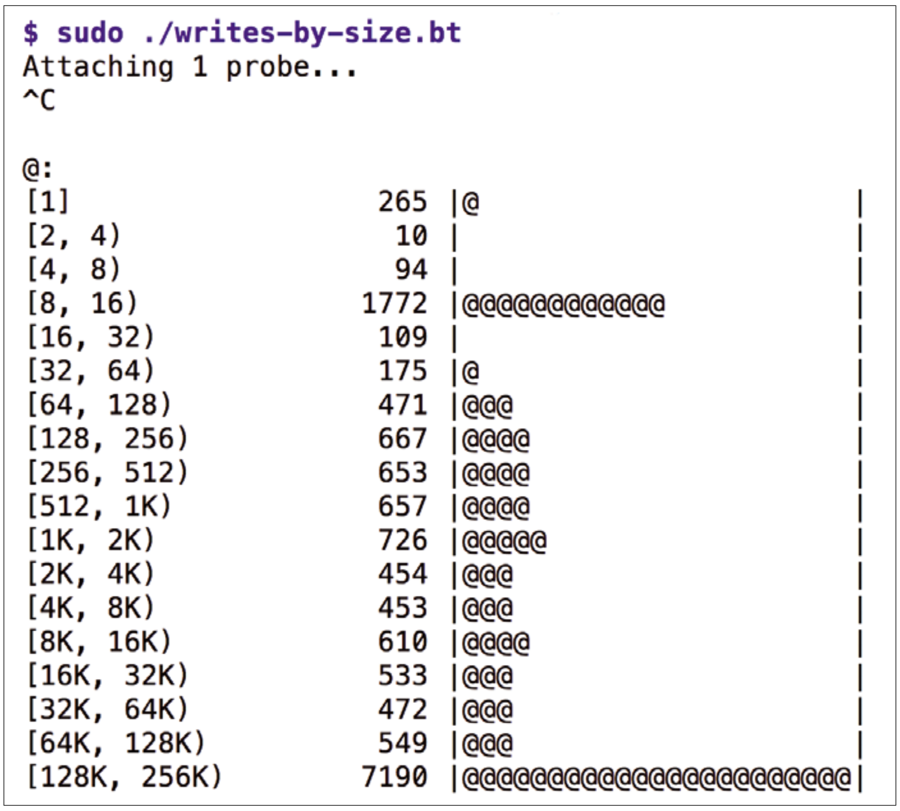
[1] "Measuring performance with the perf kernel tool" by Paul Menzel, *Linux Magazine*, issue 221, April 2019, p. 20-23, *http://www.linux-magazine.com/Issues/2019/221/perf/*

[2] Gregg, Brendan. *BPF Performance Tools*. Addison-Wesley, 2019, *https://www.amazon.com/BPF-Performance-Tools-Brendan-Gregg/dp/0136554822*

[3] Listings for the article: *ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/230/*

[4] bpftrace Reference Guide: *https://github.com/iovisor/bpftrace/blob/master/docs/reference_guide.md*

[5] BPF Cheat Sheet: *http://brendangregg.com/BPF/bpftrace-cheat-sheet.html*



```
$ sudo ./writes-by-size.bt
Attaching 1 probe...
^C

@:
[1]                  265 |@                          |
[2, 4)                10 |                           |
[4, 8)                94 |                           |
[8, 16)             1772 |@@@@@@@@@@@@               |
[16, 32)             109 |                           |
[32, 64)             175 |@                          |
[64, 128)            471 |@@@                        |
[128, 256)           667 |@@@@                       |
[256, 512)           653 |@@@@                       |
[512, 1K)            657 |@@@@                       |
[1K, 2K)             726 |@@@@@                      |
[2K, 4K)             454 |@@@                        |
[4K, 8K)             453 |@@@                        |
[8K, 16K)            610 |@@@@                       |
[16K, 32K)           533 |@@@                        |
[32K, 64K)           472 |@@@                        |
[64K, 128K)          549 |@@@                        |
[128K, 256K)        7190 |@@@@@@@@@@@@@@@@@@@@@@@@@@@|
```

Figure 4: Listing 5 sorts the writes by block size.

MakerSpace

Pyro – Networking made simple

One for All

Pyro allows multiple hardware devices to interact as if they are all on a local machine by hiding the networking. *By Scott Sumner*

Several of my projects have required multiple Raspberry Pis working in tandem to accomplish an ultimate goal, such as driving multiple independent displays or integrating a device with a dedicated controlling computer. Sometimes the setup had unique hardware (e.g., sensors); other times distance made it easier to use a remote system and WiFi rather than a lot of cabling. Although you can choose from many approaches to distributed technology, here, I'll focus on the Python remote objects (Pyro) library.

My most recent project that fell in this category was a set of scoreboards for my church's Vacation Bible School. I integrated four large LCD TVs into the set design (Figure 1) and dedicated a Raspberry Pi to each one. Also, I wanted the ability to update each team's score in real time from a centralized console. To accomplish this, I wrote the code for the scoreboards and their controller in Python and communicated between the different screens with Pyro [1].

Each Pi runs a Raspbian Lite distro and a custom Python script. After flashing the SD card with Raspbian, the only special configuration was to connect to the WiFi network and install the Pyro library. Python is installed by default with Raspbian, and the PyGame library, also a default, provided the graphics. PyGame can drive the Pi framebuffer directly, so the graphical desktop isn't needed.

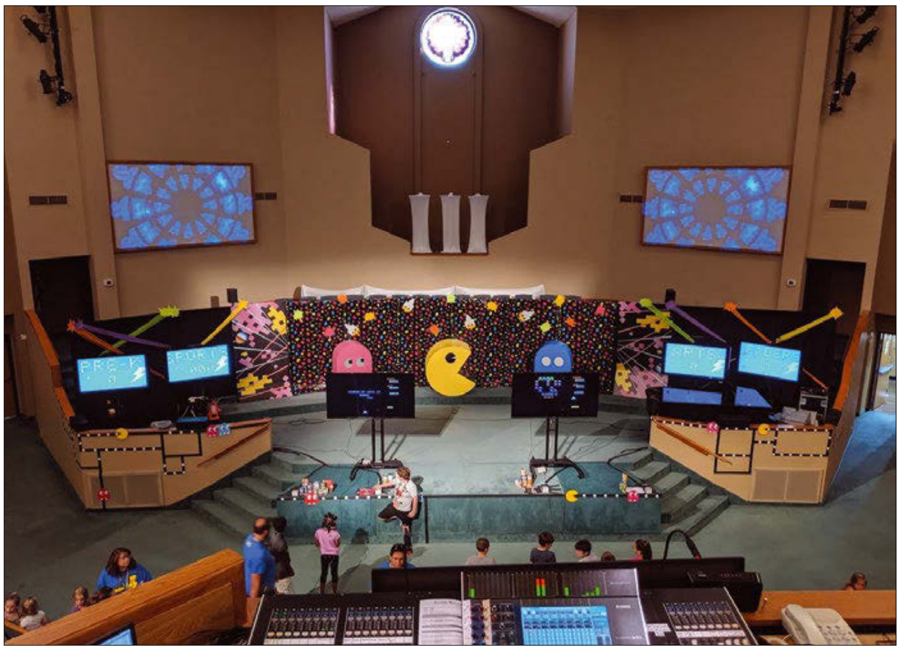


Figure 1: The four monitors as seen from the audio console. Also shown are two screens for Nintendo Entertainment Systems (center) and worship center projector screens (top). The computer barely visible in the bottom left corner accesses the web manager.

Elements of a Pyro System

To begin, I'll look at the three parts of a Pyro system and the network individually. Usually, all parts run on separate hardware, but that's not a requirement; they will happily coexist on a single computer for testing, or if it best fits your application. Figure 2 shows how a Pyro network is laid out.

Network

For Pyro to work, physical devices must be able to talk across the network. As long as everything is on the same router (either wired or WiFi), you should be in good shape. To check for basic connectivity between devices, use `ip a` and `ping` (see the "Checking Network Connectivity" box for more details). If everything is running on a single computer, you're also in good shape.

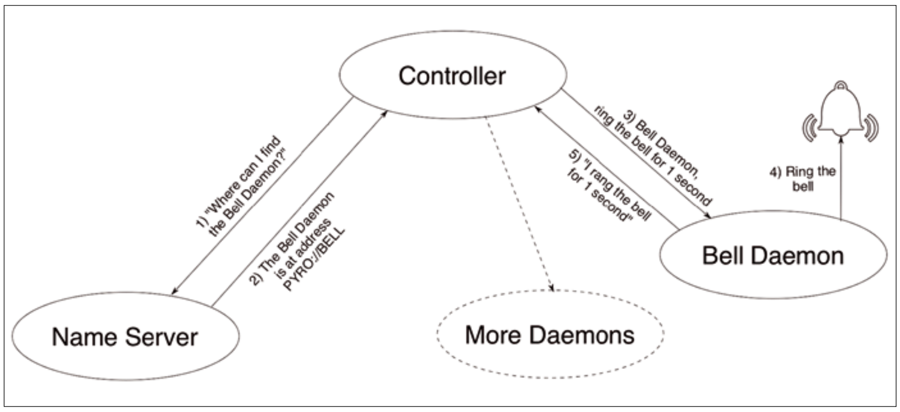

Controller
1) "Where can I find the Bell Daemon?"
2) The Bell Daemon is at address PYRO://BELL
3) Bell Daemon, ring the bell for 1 second
4) Ring the bell
5) "I rang the bell for 1 second"
Bell Daemon
Name Server
More Daemons

Figure 2: A typical Pyro network and the flow of a request through the system.

Daemons

In Linux, daemons are processes that run in the background to take care of services like printing, checking email, serving web pages, and myriad other tasks. Pyro uses the term "daemon" to describe its workers. A Pyro daemon provides data (class properties) and things it can do (class methods) on the Pyro network. The Python that you write in each daemon makes the provided task happen. Pseudocode looks something like:

Checking Network Connectivity

First you'll need to know the IP address of the device on which your daemon will run. You might already know this information if you've logged in over SSH, but, if not, open a terminal on the "remote" system and type `ip a`. Then look through the output for *inet* followed by an IP address. Note that entry 0 is usually *localhost*, so you're looking for an address that doesn't start with

127.0 ....

Once you have the IP address, go back to your "controller" computer and open a terminal there. You should be able to type

```
ping <IP address from above>
```

and start seeing replies. If not, make sure that both computers are on the same network or router. For larger networks, you might also need to adjust your subnet mask.

A mask of 255.255.255.0 requires the first three numbers of the address to be the same. You can change the mask to 255.255.0.0 and require only the first two numbers to be the same. Make this change, reboot both systems, and try your ping again.

```
@Pyro4.expose
def ringBell():
  bellPin = 17
  GPIO.out ( bellPin , True )
  time.wait ( 1 )
  GPIO.out ( bellPin , False )
  Return "Bell rang for 1 second"
```

If this looks familiar, you're right! The code to implement your task works just like any other Python code you might write to accomplish a task. The only difference is the decorator on the first line, which lets Python know that it is a Pyro function accessible to the outside world. More on that later.

Controllers

Now that you have a set of daemons out on the network ready to do your bidding, you need a controller (or two or 10) to direct those tasks. Here is where Pyro starts to shine through. After a few lines of configuration, you call a daemon as if it's a local function. All of the networking, routing, connections, packets, and data transfer happens behind the scenes. The code to ring the bell would be:

```
message = remoteBell.ringBell()
```

If the remote function returns any data, it is passed back over the network and returns normally. A Pyro network can have multiple controllers, and each controller can connect to multiple daemons. You can start to see how this arrangement can simplify an otherwise complex system.

Name Server

The name server, similar to its web counterpart, is the "phone operator" of the system. Daemons connect to the nameserver and say something like, "My service is called 'bell' and I can ring a bell." Controllers connect to the name server and ask, "Where can I find a service named 'bell'?" The name server consults its phone book and returns a URI for the bell daemon. After that, the two are free to communicate directly, even if the name server goes offline. Just like a telephone system, if you know the URI of your desired daemon, you can connect to it directly without a name server.

Starter Example

The first example is a complete Pyro system with a Python daemon that draws a colored square on a screen with the PyGame library. The controller asks for user input and then draws the square as requested. The two scripts talk via Pyro and a name server. Everything is set up to run on a single computer, and you should run each script in its own terminal.

Before you begin, install Pyro (see the "Installing Pyro" box for instructions and testing) and open three terminals. In terminal 1, start a name server (Figure 3, top terminal):

```
python -m Pyro4.naming
```

The `-m` tells Python to run this module as a script. In this case, Pyro will start a name server. The name server doesn't say much, but it is very efficient at its job and lets all of the scripts communicate with each other seamlessly.

In terminal 2, start the receiver (Figure 3, bottom left):

```
python receive.py
```

Nothing will happen until the transmitter initiates a connection, so in terminal 3, start the transmitter (Figure 3, bottom right):

```
python transmit.py
```

Once the transmitter starts, it initiates the Pyro connection, and the receiver, in this example, draws a square on the screen; then, the transmitter asks for input. It understands the commands `RED`, `GREEN`, and `BLUE` or three numbers (0-255) separated by commas. If Python can split the three

Installing Pyro

The Pyro library isn't installed by default, so you need to open a terminal and type

```
sudo pip3 install Pyro4
```

to download and install Pyro. You can confirm that it is installed with:

```
python3
>>> import Pyro4
```

If you get another Python prompt and no error messages, you are good to go!

numbers successfully, it treats them as an RGB value and sets the square color appropriately (e.g., `200,200,200` colors the box gray). After I've shown you how to set up the code with Pyro later in this article, you can try setting the box color a few times and then use `TIME` to see how long you've been playing with the system (Figure 3, bottom right).

Converting Python to Pyro

In many cases, Python classes you've already written can be converted directly to Pyro objects just by adding a few decorators. Decorators are Python's way of extending the functionality of a class or function without explicitly modifying it.

The following are some of the most common Pyro decorators on the class:

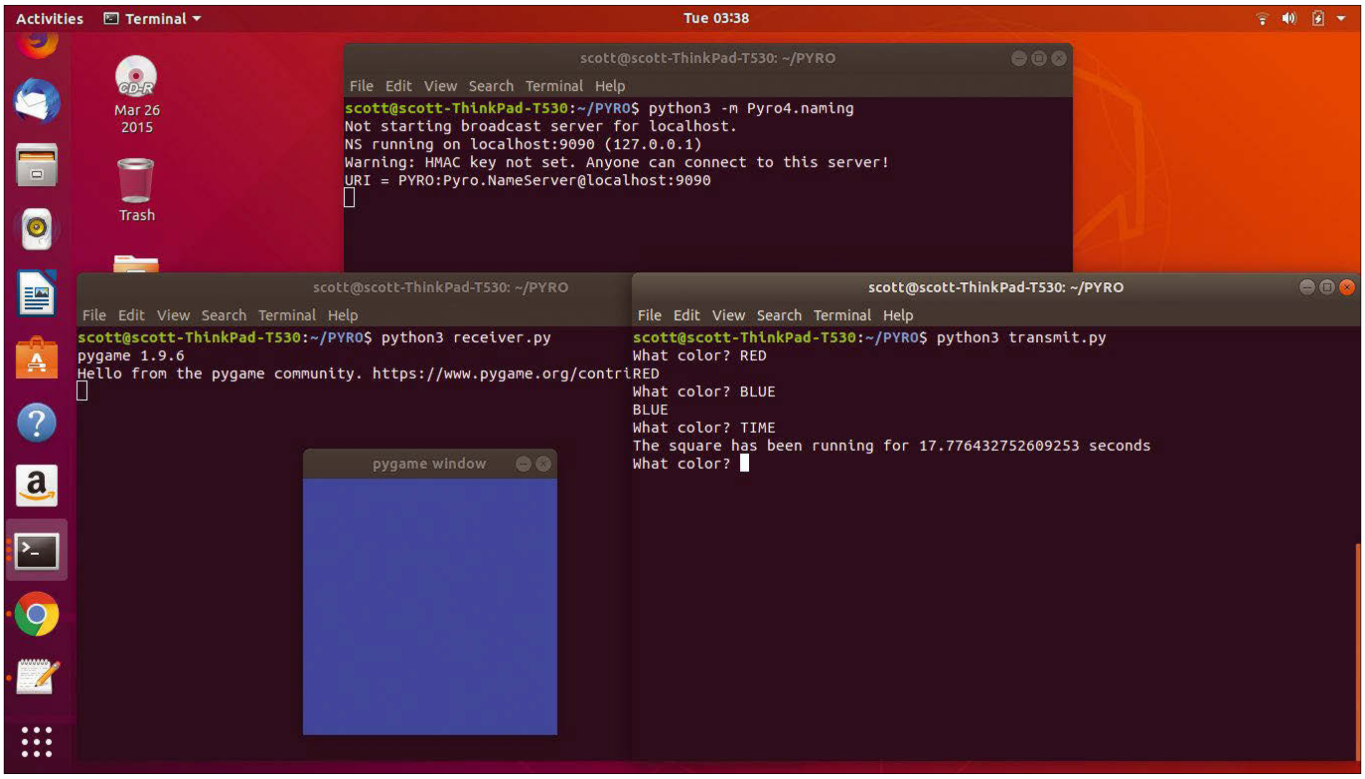
- `@Pyro4.behavior(instance_mode="single")`: Tells Pyro to use a single instance of the class regardless of how many controllers connect to the daemon. When you're dealing with a finite resource (displays, bells, pixels), this mode helps prevent resource conflicts. You can also specify `percall` for the instance mode, which creates a new version of the daemon for every call, regardless of where it originates. Once the call is finished, the instance is discarded. If you don't specify an instance mode, Pyro uses `session`. A new instance of the daemon will be created for each unique controller that connects. Each controller can see changes it has made to its specific daemon, but multiple controllers will not share data.
- `@Pyro4.expose`: Exposes an existing class to the Pyro network. When put above a class definition, every class method and property is then available to any controller that connects, which is handy if you can't modify the class itself; however, it can also introduce some security holes.

Common Pyro decorators for functions include:

- `@Pyro4.expose`: Exposing specific methods inside classes is the preferred method of making class methods available to the Pyro network. Pyro handles the movement of all of the data across the network, and you call the method as if it were running locally.
- `@Pyro4.oneway`: Tells Pyro to return immediately rather than wait for a response, which is preferred if you're going to do something that takes a long time. The downside is that you can't return a value to the controller. Any calls to this method return immediately to the controller, and the process is handled in its own thread on the daemon.
- `@property`: Tells Pyro to treat the method as a read-only property. In this case, it is accessible through `remoteObject.timeValue`:

```
@property
def timeValue ( self ):
  return self.value
```

- `attr.setter`: Tells Pyro to treat the method as a writable property:


Activities
Terminal
Tue 03:38
Mar 26 2015
Trash
scott@scott-ThinkPad-T530: ~/PYRO
File Edit View Search Terminal Help
pygame window

```
scott@scott-ThinkPad-T530:~/PYRO$ python3 -m Pyro4.naming
Not starting broadcast server for localhost.
NS running on localhost:9090 (127.0.0.1)
Warning: HMAC key not set. Anyone can connect to this server!
URI = PYRO:Pyro.NameServer@localhost:9090
```

```
scott@scott-ThinkPad-T530:~/PYRO$ python3 receiver.py
pygame 1.9.6
Hello from the pygame community. https://www.pygame.org/contri
```

```
scott@scott-ThinkPad-T530:~/PYRO$ python3 transmit.py
What color? RED
RED
What color? BLUE
BLUE
What color? TIME
The square has been running for 17.776432752609253 seconds
What color?
```

Figure 3: Calling three Pyro scripts for the name server (top), the receiver (bottom left), and the transmitter (bottom right). The colored square is the "remote" object being drawn by `receive.py` but controlled by `transmit.py`.

```
@attr.setter
def editableTime ( self , newValue ):
   self.value = newValue
```

In this case, `remoteObject.editableTime = 25`.

Colorsquare Example

The Colorsquare example has two parts: a transmitter and a receiver. I'll look at `receive.py` (Listing 1) first. Pyro is imported just like any other library, and I import `PyGame` and `time`, as well (lines 3-5).

Lines 8-16 initialize the Pyro class. The first Pyro decorator in line 8 tells Pyro to create a `single` instance of the class and use it for all requests. If you were to leave this out, each controller that connected to the class would spawn a separate instance of the class.

Line 11 defines the `__init__` function, which works as expected, with one caveat – it is only called when something connects to the Pyro object. If nothing ever connects, then `__init__` never runs. Lines 13 and 14 initialize PyGame, which draws the square onscreen, and line 16 stores the start time of the process, so run time can be calculated later.

The Pyro function appears in lines 22-28. `@Pyro4.expose` in line 22 tells Pyro that this function can be called by a connected controller. Class methods without this decorator won't be accessible with Pyro calls.

Lines 24 and 25 change the color of the square on the screen and then redraw the display so the new color is visible. As with most functions, `return` is found at the end (line 28). Pyro takes care of getting the return value back across the network to wherever the call initiated.

The other functions in lines 30-34, 36-40, and 42-46 all work exactly the same way. The only difference is a hard-coded color value. The `getTime` in lines 48-50 doesn't change the display; it simply returns how long the daemon has been running.

Now the code sets up Pyro and goes from just a class to a remote object (lines 52-61). To start, line 53 creates a daemon; then, the class is registered with the daemon (line 55) and returns its own URI. You can then save the URI in a database, write it to a temp file, or (as done

Listing 1: receive.py

```
01 # Nothing different here, just import the Pyro4 library
     along with whatever else you need
02
03 import Pyro4
04 import pygame
05 import time
06
07 # Tell Pyro to use a single instance of the class
     regardless of where the request originates
08 @Pyro4.behavior(instance_mode="single")
09 class colorSquare:
10    # __init__ runs like normal, the only difference is it
        won't get called until something requests the Pyro
        object
11    def __init__ ( self ):
12       # Standard PyGame init
13       pygame.display.init()
14       self.window = pygame.display.set_mode ( ( 256 ,
           256 ) )
15       # Bookmark the start time
16       self.start = time.time()
17
18    # Pyro4.expose makes this function accessible to the
        outside world
19    # Anything without expose is private
20    # You can also expose the entire class if you want
21    # Or declare @Pyro4.oneway which will make the call
        return None immediately and allow the remote process
        to run for as long as it needs to
22    @Pyro4.expose
23    def setColor ( self , color ):
24       self.window.fill ( color )
25       pygame.display.flip()
26       # Whatever you return will get passed back to the
           remote caller
27       # All python types and simple classes are supported
28       return "CUSTOM"
29
30    @Pyro4.expose
31    def setRed ( self ):
32       self.window.fill ( ( 255 , 0 ,0 ) )
33       pygame.display.flip()
34       return "RED"
35
36    @Pyro4.expose
37    def setGreen ( self ):
38       self.window.fill ( ( 0 , 255 , 0 ) )
39       pygame.display.flip()
40       return "GREEN"
41
42    @Pyro4.expose
43    def setBlue ( self ):
44       self.window.fill ( ( 0 , 0 , 255 ) )
45       pygame.display.flip()
46       return "BLUE"
47
48    @Pyro4.expose
49    def getTime ( self ):
50       return time.time() - self.start
51
52 # Create a daemon -- Pyro's worker class to serve an
     object
53 daemon = Pyro4.Daemon()
54 # Get the URI of the daemon
55 uri = daemon.register ( colorSquare )
56 # Find the nameserver on the network
57 ns = Pyro4.locateNS()
58 # Let the nameserver know what we answer to and where to
     find us
59 ns.register ( "square" , uri )
60 # Listen for and handle requests
61 daemon.requestLoop()
```

here) use it to register with the name server. Line 57 finds the name server and line 59 registers it and lets it know how the service should be known. Now you can start listening and respond to calls as they arrive (line 61).

Setting up the controller is even easier (Listing 2). To begin, import Pyro (line 1) and find the name server (line 4), as in the receiver code. Next, search for the daemon to which you want to connect (line 6). Finally, create a proxy to the remote object (line 8).

These steps can be repeated as needed to connect to as many daemons as you like. Afterward, you can treat everything as if it is local.

The rest of the script asks for the user input (line 13) described earlier (Figure 3, bottom right) and then processes it. If it is a color known by name, the script calls that color's function (lines 16-18). If `TIME` has been requested, it is calculated and returned (line 19); otherwise, the program tries to parse the input as three comma-separated numbers (lines 22-26). The `try/except` exception handler (lines 22-25) will display a message if it can't find three numbers separated by commas.

Scoreboards

Each scoreboard runs identical code, with the exception of the Pyro name registered with the name server. To start the program on each Rasp Pi, I logged in via SSH from my desktop and started the Python script directly from the terminal. Each Rasp Pi had a separate tab, so any problems with connectivity or other errors could be checked from the console to see what's happening. My desktop was running the Python name server and controller, as well.

Each scoreboard had three major functions: updating the team name, updating the score, and displaying a timer. Once started, the timer mode was exited by pushing a stop button. Each button was wired to the Rasp Pi's GPIO via twisted pair wire. A simple header was connected to GPIO14 (physical pin 8, transmit) [2], which was selected because of the adjacent ground connection.

In the code walkthrough, I'll focus on the Pyro functions, although the majority of the code is actually for the graphics. As with any Python program, you have to start by importing libraries (Listing 3). In this case I import Pyro, `pygame`, `time`, and `RPi.GPIO`. Next, I set the mode of the GPIO pin numbering to the Broadcom SoC channel number, not the pin number on the board. Although I'm only using one GPIO pin, this is the pin numbering I'm used to.

The `@Pyro4.behavior` decorator (line 8) sets the `instance_mode` to `single`, so the class will be instantiated once, and all calls will use the same instance.

The `__init__` method is standard, but remember that Pyro will not instantiate a class until it is actually called from a controller, which can make your first call seem a little sluggish as everything gets set up. Response times will improve afterward. As mentioned before, if nothing ever connects to the associated Python object, then the `__init__` method will not be called.

Lines 11, 14, 19, 23, and 40 print status messages that I can monitor from the terminal from which I launch each daemon (Figure 4). Lines 12 and 13 initialize PyGame and set up a window in which to draw. The `pygame.FULLSCREEN` removes all window decoration and maximizes the window to take over the entire desktop.

Next, I set up the fonts (lines 16-18); The `init()` starts the PyGame font module, and then `pygame.font.Font` loads a font from disk. The second argument specifies the font size in pixels. Each instance of a font object has the font itself and the size it should be rendered. If I want to use the same font in different sizes, I have to initialize it multiple times.

To set up the background image (lines 21-23), I load it from disk then add `.convert()` to the end to store it internally in the same format as my initialized display; `self.screen.blit` then draws it on the display.

The next group of lines that begin with `self` set default values for scores, team names, drawing color, and timers, with the timer status. I also set up some variables for blitting the background back onto itself to erase portions of the display.

The next lines (37-38) set up the GPIO – in this case, one input pin. The `pull_up_down` entry gives the input pin the electronic equivalent of a default value. In the absence of a signal, the pull up will make

Listing 2: transmit.py

```
01 import Pyro4
02
03 # Find the nameserver
04 ns = Pyro4.locateNS()
05 # Ask the nameserver for the URI to the remote object we
     want
06 uri = ns.lookup ( "square" )
07 # Create a connection to the remote object
08 remoteSquare = Pyro4.Proxy ( uri )
09
10 # Repeat as necessary for multiple remote objects
11
12 while 1:
13    request = raw_input ( "What color? " )
14    # Treat the remote object above as if it is local
15    # The return value (just printed here) is whatever the
        remote function wants to send back (if anything)
16    if request == "RED": print ( remoteSquare.setRed() )
17    elif request == "GREEN": print (
        remoteSquare.setGreen() )
18    elif request == "BLUE": print ( remoteSquare.setBlue() )
19    elif request == "TIME": print ( "The square has been
        running for " + str ( remoteSquare.getTime() ) + "
        seconds" )
20    else:
21       # fail gracefully if we can't decode 3 numbers
           separated by commas
22       try:
23          RGB = request.split ( "," )
24          print ( remoteSquare.setColor ( ( int ( RGB [ 0
                ] ) , int ( RGB [ 1 ] ) , int ( RGB [ 2
                ] ) ) ) )
25       except:
26          print ( "Couldn't parse an R,G,B argument" )
```

Listing 3: display.py

```
import Pyro4
import pygame
import time
import RPi.GPIO as GPIO

GPIO.setmode ( GPIO.BCM )

@Pyro4.behavior(instance_mode="single")
class scoreboard:
    def __init__ ( self ):
        print "Starting Scoreboard"
        pygame.display.init()
        self.screen = pygame.display.set_mode
          ( ( 1280 ,720 ) , pygame.FULLSCREEN )
        print "Setting full screen 1280x720"

        pygame.font.init()
        self.scoreFont = pygame.font.Font
          ( "OOTT.TTF" , 300 )
        self.titleFont = pygame.font.Font
          ( "OOTT.TTF" , 400 )
        print "Fonts initialized"

        self.background = pygame.image.load
          ( "pu1280x720.png" ).convert()
        self.screen.blit ( self.background , ( 0 , 0 ) )
        print "Background loaded"

        self.scoreGoal = 0
        self.currentScore = 0

        self.teamName = "ARTS"
        self.titleColor = ( 250 , 218 , 94 )

        self.oldRect = None

        self.startTime = None
        self.endTime = None
        self.running = False

        GPIO.setup
          ( 14 , GPIO.IN , pull_up_down=GPIO.PUD_UP )
        GPIO.add_event_detect
          ( 14 , GPIO.FALLING , callback=self.bumpPoints )

        print "Done with init"

    def bumpPoints ( self , channel ):
        self.updateScore ( 10 )

    @Pyro4.oneway
    def updateTitle ( self , name ):
        self.teamName = name
        self.screen.blit ( self.background , ( 0 , 0 ) )

        self.drawTitle()
        self.drawScoreLocal()
        pygame.display.flip()

    @Pyro4.expose
    def timerRunning ( self ):
        return self.running

    @Pyro4.oneway
    def drawTitle ( self ):
        title = self.titleFont.render
          ( self.teamName , True , self.titleColor )
        X = self.screen.get_width() / 2 - title.get_
            width() / 2
        self.screen.blit ( title , ( X , 10 ) )
        print "Draw Title"

    @Pyro4.oneway
    def drawScore ( self ):
        self.drawScoreLocal()
        print "Draw Score"

    def drawScoreLocal ( self ):
        score = self.scoreFont.render ( str
          ( self.currentScore ) , True , ( 255 , 255 ,
          255 ) )
        shadow = self.scoreFont.render ( str
          ( self.currentScore ) , True , ( 0 , 0 , 0 ) )

        X = self.screen.get_width() / 2 - score.get_
            width() / 2

        blitArea = pygame.rect.Rect
                   ( 0 , 400 , 1280 , 300 )
        self.screen.blit ( self.background , blitArea ,
                           blitArea )

        self.screen.blit ( shadow , ( X + 5 , 405 ) )
        self.screen.blit ( score , ( X , 400 ) )
        pygame.display.update ( [ blitArea ] )
        print "Draw score local"

    @Pyro4.oneway
    def updateScore ( self , scoreDelta ):
        self.scoreGoal += scoreDelta
        nextUpdate = time.time() + .01

        while self.scoreGoal != self.currentScore:
            if time.time() < nextUpdate: continue
            nextUpdate = time.time() + .01

            print self.scoreGoal , self.currentScore

            if self.scoreGoal < self.currentScore:
                self.currentScore -= 5
                self.drawScoreLocal()
            elif self.scoreGoal > self.currentScore:
                self.currentScore += 5
                self.drawScoreLocal()
        print "Update score"

    @Pyro4.oneway
    def startTimer ( self ):
        self.startTime = time.time()
        self.running = True
        self.updateTimer()

    def updateTimer ( self ):
        nextUpdate = time.time() + .01
        while self.running == True:
            if time.time() < nextUpdate: continue
            nextUpdate = time.time() + .01

            elapsed = time.time() - self.startTime
            score = self.scoreFont.render
                    ( "{0:03.3f}".format ( elapsed ) ,
```

Listing 3: display.py (continued)

```
                    True , ( 255 , 255 , 255 ) )
117         shadow = self.scoreFont.render
                     ( "{0:03.3f}".format ( elapsed ) ,
                     True , ( 0 , 0 , 0 ) )
118
119         X = self.screen.get_width() / 2 - score.get_
                width() / 2
120
121         blitArea = pygame.rect.Rect
              ( 0 , 400 , 1280 , 300 )
122         self.screen.blit
              ( self.background , blitArea , blitArea )
123
124         self.screen.blit ( shadow , ( X + 5 ,
              405 ) )
125         self.screen.blit ( score , ( X , 400 ) )
126         pygame.display.update ( [ blitArea ] )
127
128             if GPIO.input ( 14 ) == 0:
                  self.running = False
129         print "update Timer"
130
131
132     @Pyro4.oneway
133     def update ( self ):
134         pygame.display.flip()
135         print "update"
136
137
138 Pyro4.config.HOST = "172.16.71.227"
139 Pyro4.Daemon.serveSimple ( {
140     scoreboard: "scoreboard2"
141     } , ns=True
142     )
```

the pin read 1, or HIGH. The `add_event_detect` sets up a trigger so that when the GPIO pin falls (i.e., a button is pressed), a class method is called (`self.bumpPoints`). The timer functions will read the GPIO directly, but this allows calling the method in the background.

On the last day of camp, we introduced an Easter Egg into the scoreboard program. The buttons along the front of the stage allowed campers to add points to their team total. Once discovered, campers staged elaborate plans to keep pressing their buttons to run the scores up. Internally, these button presses call `self.updateScore` (line 43), awarding the team 10 points with each press.

Updating the Screens

The `@Pyro4.oneway` decorator (line 45) tells Pyro not to wait for a response, but to run this method in its own thread so the main program doesn't block. In this way, the controller isn't waiting for each display to update before returning to monitor the GUI.

The title is the team name displayed at the top of the screen. After storing the new team name in the class (line 47), line 48 blits the background image onto the entire display to erase it. Next, I call `drawTitle` and `drawScoreLocal` to redraw all of the information for the scoreboard. Finally, `pygame.display.flip` draws everything to the screen to make the changes visible.

The sole purpose of the utility function `timerRunning` (line 55) is to allow the controller to check whether the timer is still being displayed by returning `self.running`. I could have done this with a property, but this method works just as well.

The `drawTitle` function under the `oneway` decorator renders the title (line 60), calculates a horizontal center point (line 61), and then draws it onscreen (line 62). Because Python was having locking problems when updating the score, drawing locally instead of in another thread spun off by Pyro solved the problem. Therefore, the `drawScore` function exposed to Pyro (line 66) calls the internal function `drawScoreLocal`, which does the work. This local function renders the score twice: once in black (line 71) and once in white (line 72).

The next task is to calculate the horizontal center (line 74), define the bounding box of the score display in a PyGame `Rect` (line 76), and then blit only that portion of the background image to clear the score (line 77). The shadow color is blitted first (line 79) followed by the score rendered in white (line 80). Finally, I call the `update` function and pass `blitArea` as a list, which tells PyGame to update only that region of the screen, rather than the entire thing. Updating the screen in this way aids in performance, because `drawScoreLocal` will be called quite often.

To update the score (lines 84-101), the `updateScore` function works a little differently from `drawScore`, in that it increments the score five points at a time. Line 86 updates `self.scoreGoal`, which is the score to be displayed, and line 87 adds 1/100th of a second to the current time for the next display update. The `while` loop (line 89) first checks the current time and, if it is not time to update, simply checks again until it is time. This loop continues until the currently displayed score and the goal score match. Lines 95-100 increment or decrement the

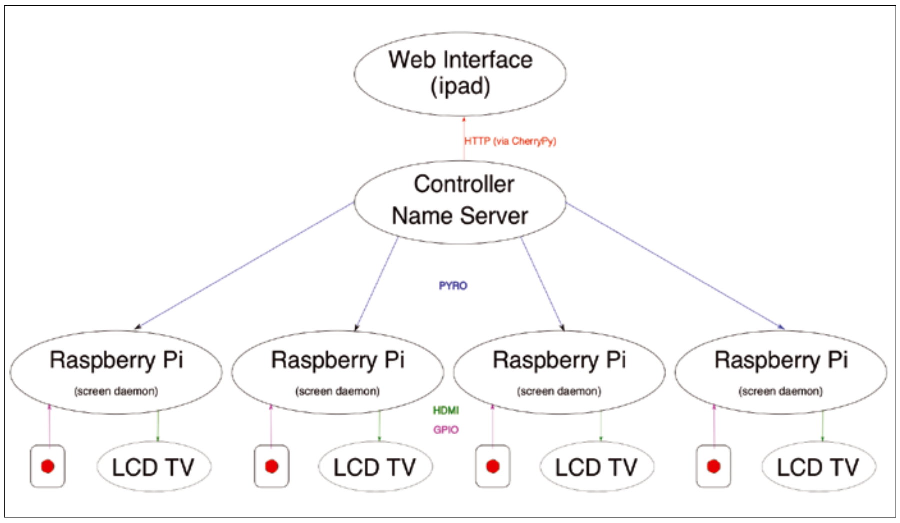

Web Interface (ipad)
HTTP (via CherryPy)
Controller Name Server
PYRO
Raspberry Pi (screen daemon)
Raspberry Pi (screen daemon)
Raspberry Pi (screen daemon)
Raspberry Pi (screen daemon)
HDMI
GPIO
LCD TV
LCD TV
LCD TV
LCD TV

Figure 4: Displays, Raspberry Pis, controllers, and daemons all talk to each other across various formats and protocols.

score and then call `drawScoreLocal` to show it onscreen.

Timers

The `startTimer` function block switches the display into timer mode by recording the current time (line 105), setting the `self.running` flag (line 106), and calling `self.updateTimer` (line 107).

The `updateTimer` function block draws the timer onscreen until the button is pressed (GPIO goes LOW). Line 110 calculates the time 1/100th of a second in the future, and the `while` loop (line 111) runs for as long as the `self.running` flag is `True`. If it is not, it continues checking until it is (line 112). Line 113 updates the time for the next pass through the loop, and line 115 calculates the elapsed time.

The next three lines render the current elapsed time in both a shadow and foreground color and calculate the horizontal center. Lines 121 and 122 blit a small portion of the background over the previous timer entry to erase it, and then lines 124 and 125 blit the shadow and foreground surfaces before line 126 draws the update to the screen. Finally, line 128 checks the GPIO input, and if it is `0` (LOW), the `self.running` flag is set to `False` to exit the loop. The `update` method allows the controller to force a screen refresh.

For the daemon to receive commands across the network, line 138 defines its IP address. Without it, the daemon will only listen on localhost. Line 139 uses Pyro's `serveSimple` utility function to start a daemon. I provide a dictionary of the class I want to serve (line 140, left of the colon) and what it should be called (`scoreboard2`, right of the colon). The final argument (line 141) asks Pyro to register this daemon with the name server.

Web Manager

The Web Manager controls everything. At any given time, it was usually loaded on my cell phone, an iPad, and a computer at the media station. The web manager (Figure 5) uses CherryPy [3] to serve a web page with JavaScript, which returns user requests via Ajax. When the Ajax calls are received, the manager script calls remote Pyro functions to control the scoreboard.

For the manager (Listing 4), I needed two libraries: `cherrypy` to act as a web server and `Pyro4` to talk to the Pyro objects already discussed.

CherryPy operates by taking a Python class and exposing certain methods as web addresses. For instance, *http://webManager.local:8080/updateTitle* calls the associated `self.updateTitle` class method. Any POST or GET variables show up as named arguments. Just as in Pyro, only methods with the `@cherrypy.expose` decorator are accessible via HTTP, so the `__init__` class that sets up all of the Pyro objects can't be accessed from a web server.

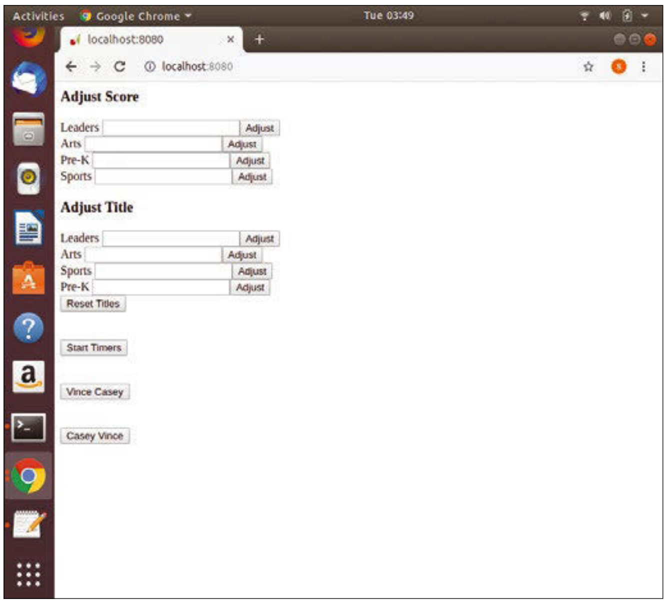


Figure 5: The web manager is a bare-bones interface, because it is only accessed by the crew. The "Vince Casey" and "Casey Vince" buttons changed the displays to show the names of the emcees as they came on stage. If they switched places, the names would follow.

Unlike Pyro, CherryPy calls the `__init__` method regardless of whether anything has connected or not, so line 6 creates a list of screens, and lines 7-10 ask Pyro to go find the addresses of the named objects from the name server and append them to the screen list; the initial graphics then are displayed on all four screens (lines 12-14).

Just like `index.html` in Apache, the `index` function here is the default destination if no specific address is specified. In this case, lines 18-166 define the manager web page. Although the HTML and JavaScript are beyond the scope of this article, the following is a brief summary.

Each of the JavaScript functions (e.g., `adjustScore`, lines 22-31) defines a function that can be called by a button or JavaScript event on the manager web page. The `obj` created in line 24 contains the POST variables (i.e., `screen`, the screen to update, and `delta`, how much to change the score).

jQuery then posts the request to `adjustScore` (line 28), which is defined in the next line. The inline `function` is called when the Ajax call completes and receives `data`, which is anything that the Python function that follows returns.

The `updateTitle` function (lines 169-170) is exposed by CherryPy and called by jQuery, as described above. The argument `screen` is used as an index for `self.screens`, and `updateTitle` is called again on the remote object via Pyro. All of that happens behind the scenes, though, so the Python code here doesn't look any different. The same thing happens with `adjustScore` (lines 173-174).

The `startTimers` function loops over the `self.screens` list (line 178) and starts the timer on all screens (line 179). This remote Pyro call runs on each Raspberry Pi. The function then sits in a loop (lines 182-186) calling `screen.timerRunning` on each Pi until one returns `False` (line 184). That screen's title is updated to show *WINNER* and a flag is set that first place has been found (line 186).

Setting the `socket_host` to 0.0.0.0 in line 189 tells CherryPy to respond to all requests regardless to which network adapter or address they arrive. The final line uses CherryPy's `quickstart` function to start serving the application from `web()` < at address "`/`".

Conclusion

As I've walked through the code, you've probably noticed that the majority of it is not related to Pyro at all; rather, the code does whatever your application would normally do. Pyro is just an overlay that allows multiple hardware devices to interact as if they are all on a local machine by hiding all of the networking, so you don't have to worry about it.

Another application I set up was connected to a sound system, so any calls for sound effects were directed to the Pyro server. Other Pyro daemons controlled four scoreboard screens similar to the application described here. Another one ran a high score display on a fifth monitor. Pyro hid all of the networking code, so all I had to do was call things like `sound.playBuzzer()` and `highScore.addTeam()`. ■■■

Info

[1] Pyro: *https://pythonhosted.org/Pyro4/*

[2] GPIO pin numbering: *https://www.raspberrypi.org/forums/viewtopic.php?p=1239056*

[3] CherryPy: *https://cherrypy.org/*

Listing 4: webManager.py

```
import cherrypy
import Pyro4

class web:
    def __init__ ( self ):
        self.screens = list()
        self.screens.append ( Pyro4.Proxy ( "PYRONAME:scoreboard1" ) )
        self.screens.append ( Pyro4.Proxy ( "PYRONAME:scoreboard2" ) )
        self.screens.append ( Pyro4.Proxy ( "PYRONAME:scoreboard3" ) )
        self.screens.append ( Pyro4.Proxy ( "PYRONAME:scoreboard4" ) )

        for screen in self.screens:
            screen.drawTitle()
            screen.update()

    @cherrypy.expose
    def index ( self ):
        html = """
           <head>
              <script src="https://ajax.googleapis.com/ajax/libs/jquery/3.4.1/jquery.js"></script>
              <script>
                 function adjustScore ( screen )
                 {
                    obj = new Object();
                    obj.screen = screen;
                    obj.delta = $ ( "#scoreInput" + screen ).val();

                    $.post ( "adjustScore" , obj , function ( data ) {
                       $ ( "#scoreInput" + screen ).val( "" );
                    } );
                 }

                 function adjustTitle ( screen )
                 {
                    obj = new Object();
                    obj.screen = screen;
                    obj.name = $ ( "#titleInput" + screen ).val();

                    $.post ( "updateTitle" , obj , function ( data ) {
                       $ ( "#titleInput" + screen ).val( "" );
                    } );
                 }

                 function resetTitles()
                 {
                    obj = new Object();
                    obj.screen = 1;
                    obj.name = "LEADERS";

                    $.post ( "updateTitle" , obj , function ( data ) {
                    } );

                    obj = new Object();
                    obj.screen = 2;
                    obj.name = "ARTS";

                    $.post ( "updateTitle" , obj , function ( data ) {
                    } );

                    obj = new Object();
                    obj.screen = 4;
                    obj.name = "SPORTS";

                    $.post ( "updateTitle" , obj , function ( data ) {
                    } );

                    obj = new Object();
                    obj.screen = 3;
                    obj.name = "PRE-K";

                    $.post ( "updateTitle" , obj , function ( data ) {
                    } );

                 }

                 function caseyVince()
                 {
                    obj = new Object();
                    obj.screen = 1;
                    obj.name = "";

                    $.post ( "updateTitle" , obj , function ( data ) {
                    } );

                    obj = new Object();
                    obj.screen = 2;
                    obj.name = "VINCE";

                    $.post ( "updateTitle" , obj , function ( data ) {
                    } );

                    obj = new Object();
                    obj.screen = 4;
                    obj.name = "CASEY";

```

Listing 4: webManager.py (continued)

```
            $.post ( "updateTitle" , obj ,
              function ( data ) {
            } );

            obj = new Object();
            obj.screen = 3;
            obj.name = "";

            $.post ( "updateTitle" , obj ,
              function ( data ) {
            } );

          }

          function vinceCasey()
          {
            obj = new Object();
            obj.screen = 1;
            obj.name = "";

            $.post ( "updateTitle" , obj ,
              function ( data ) {
            } );

            obj = new Object();
            obj.screen = 2;
            obj.name = "CASEY";

            $.post ( "updateTitle" , obj ,
              function ( data ) {
            } );

            obj = new Object();
            obj.screen = 4;
            obj.name = "VINCE";

            $.post ( "updateTitle" , obj ,
              function ( data ) {
            } );

            obj = new Object();
            obj.screen = 3;
            obj.name = "";

            $.post ( "updateTitle" , obj ,
              function ( data ) {
            } );

          }

          function startTimers()
          {
            $.post ( "startTimers" );
          }
        </script>
      </head>

      <h3>Adjust Score</h3>
      <div>Leaders <input type='text'
        id='scoreInput1'><input type='button'
        onclick='adjustScore ( 1 )'
        value='Adjust'></div>
      <div>Arts <input type='text'
        id='scoreInput2'><input type='button'
        onclick='adjustScore ( 2 )'
        value='Adjust'></div>
      <div>Pre-K <input type='text'
        id='scoreInput3'><input type='button'
        onclick='adjustScore ( 3 )'
        value='Adjust'></div>
      <div>Sports <input type='text'
        id='scoreInput4'><input type='button'
        onclick='adjustScore ( 4 )'
        value='Adjust'></div>

      <h3>Adjust Title</h3>
      <div>Leaders <input type='text'
        id='titleInput1'><input type='button'
        onclick='adjustTitle ( 1 )'
        value='Adjust'></div>
      <div>Arts <input type='text'
        id='titleInput2'><input type='button'
        onclick='adjustTitle ( 2 )'
        value='Adjust'></div>
      <div>Sports <input type='text'
        id='titleInput4'><input type='button'
        onclick='adjustTitle ( 4 )'
        value='Adjust'></div>
      <div>Pre-K <input type='text'
        id='titleInput3'><input type='button'
        onclick='adjustTitle ( 3 )'
        value='Adjust'></div>

      <div><input type='button' value=
        'Reset Titles' onclick='resetTitles()'>
        </div>
      <div><br><br><input type='button' value=
        'Start Timers' onclick='startTimers()'>
        </div>
      <div><br><br><input type='button' value=
        'Vince Casey' onclick='vinceCasey()'></div>
      <div><br><br><input type='button' value=
        'Casey Vince' onclick='caseyVince()'></div>


    """
    return html

  @cherrypy.expose
  def updateTitle ( self , screen , name ):
    self.screens [ int ( screen ) - 1 ].updateTitle
      ( str ( name ) )

  @cherrypy.expose
  def adjustScore ( self , screen , delta ):
    self.screens [ int ( screen ) - 1 ].updateScore
      ( int ( delta ) )

  @cherrypy.expose
  def startTimers ( self ):
    for screen in self.screens:
      screen.startTimer()

    firstPlace = False
    while firstPlace == False:
      for screen in self.screens:
        if screen.timerRunning() == False:
          screen.updateTitle ( "WINNER" )
          firstPlace = True


cherrypy.server.socket_host = "0.0.0.0"
cherrypy.quickstart ( web() , "/" )
```

MakerSpace

Open source e-paper

Recycled E-Ink

Combining open firmware with recycled hardware, Inkplate launches a crowdfunding campaign for an open source e-paper display. *By Bruce Byfield*

From the start, open hardware has repurposed used technology. Companies like Minifree and Technoethical have based their entire business plans around such recycling. By the time you read this article, these companies will have been joined by a crowdfunding campaign for Inkplate 6 [1], an e-paper display that combines open source firmware with parts from used Kindle e-readers.

Inkplate 6 is developed by e-radionica.com [2], a company started by David Zovko when he was 16. The company's first product was the Croduino Basic microcontroller, an Arduino-compatible board named for the fact that it was manufactured in Croatia. Today, e-radionica.com offers 70 Arduino-compatible boards, all of which are open hardware. "Since openness enabled my creation of the first board," Zovko says, "I have committed that all my maker projects will be open source."

E-paper (aka electronic paper or e-ink) is a display technology that imitates the look and resolution of ink on paper (Figure 1). Unlike the typical computer monitor, e-paper reflects light, just like paper. As a result, e-paper not only reduces eye strain and has a wider viewing angle, but it is readable in direct sunlight without appearing to fade. Although e-paper color displays have existed for close to a decade, the majority of e-paper is currently grayscale. The most popular use of e-paper is in e-readers like the Kindle or the Kobo, but other uses include arrival and departure displays in airports, electronic billboards, and smartphone displays.

"E-papers have always been a fascination to me," Zovko says. "They look pretty neat and use no power to show contents on the screen. [However], after trying some of the displays available to makers, I didn't find them simple enough to use, and some features were missing. After finding that there are recycled Kindle displays, one thing led to another and the Inkplate was made."

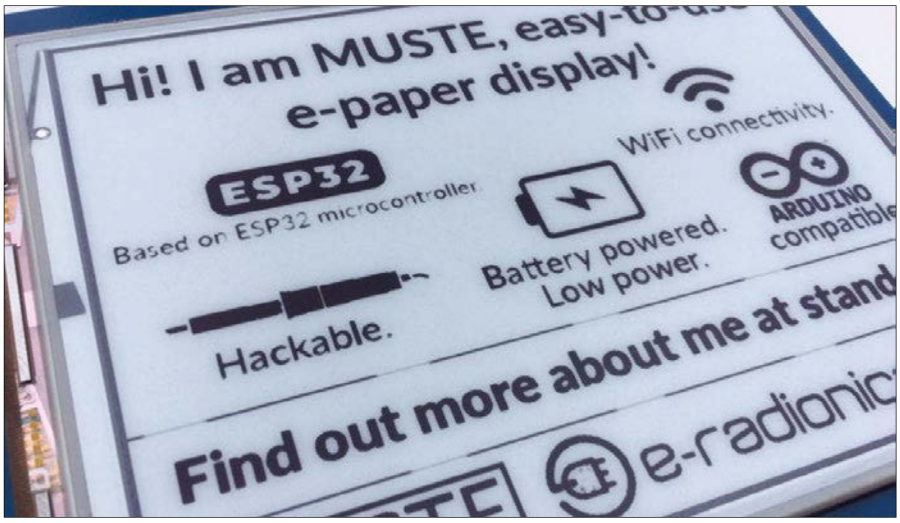


Figure 1: E-paper, like the Inkscape 6 screen shown here, features a high resolution and a more readable screen.

Inkplate Features and Audience

Zovko goes on to explain that "Inkplate 6 is a display designed to make using e-paper extremely simple. Simplicity is achieved in both hardware and software. Just plug in a USB cable or battery, open the Arduino IDE, and change what's on the screen with a few lines of code. No connecting extra cables or complicated code. The product is based on the powerful ESP32 microcontroller (Figure 2), which has WiFi and Bluetooth connectivity. There's a battery charger, GPIO ports for extra tinkering, an SD card for image and book storage, and three touch-buttons below the screen."

Other features will include:

- A recycled 6-inch Kindle e-paper display with a resolution of 800x600 pixels
- Low power consumption
- Arduino libraries that enable simple customizations: A few lines of code is all you need to display text and images, change grayscale settings, or enable partial updates for faster refresh cycles
- A 3D-printable enclosure

Like other devices built with an open source microcontroller, the Inkplate 6 is designed to have flashable firmware. "You can customize any part of the screen using the Arduino IDE or MicroPython. You can show custom text, images, and even greyscale images. In the Arduino, it's compatible with the well-known Adafruit GFX library. So, you can put any contents you can imagine on it."

"The product is made for makers, if that's not too vague to say," Zovko says, speaking of the do-it-yourself community that overlaps with the open source movement. "It's completely open and customisable, so anyone can make it into what they want. That might be a desktop planner, which includes today's calendar events, the latest emails and weather reports, and is updated automatically. Or it might be a meeting room sign, which shows who's having meetings in a room at which time, or even an e-book reader. We hope that everyone will have a specific idea, although we will provide examples."

Development Challenges

Recycling hardware can be a shortcut to market. "We had the option to pick between new and recycled displays," Zovko says, "and recycled ones just seemed like a natural option. Why not use something that still works?" However, Zovko notes that recycling can create its own problems. "There were a few major ones," he says, "the biggest being that documentation was not available. With no proper datasheet or information on how to drive the display, it was quite the challenge to get it working properly, especially the greyscale mode. But with a lot of salvaging from the Internet and reverse-engineering of the Kindle device with that specific display, we have come to a working product."

At the same time, the number of recycled Kindle displays is limited. "Right now, there are a few thousand available screens, and that's a really fair concern. [But] that should be enough for quite some time."

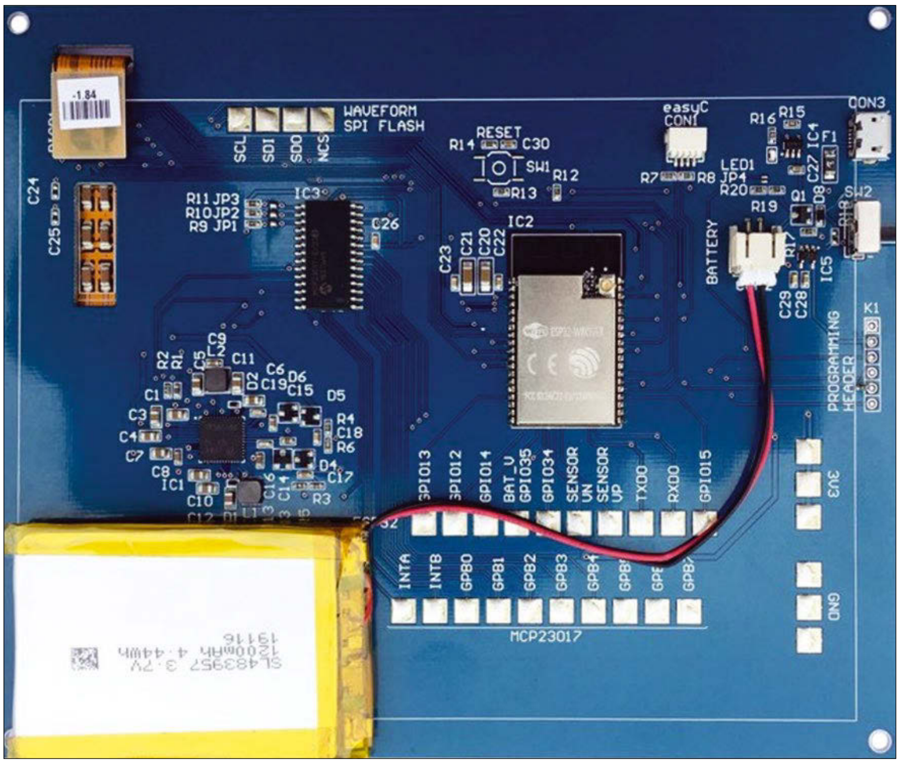


Figure 2: The Inkplate 6 is based on the ESP32 microcontroller, a versatile choice that can be used for multiple purposes.

Upcoming Crowdfunding Campaign

At the time of writing, the Inkplate campaign page is limited to basic information and a link to a mailing list. However, once the Inkplate campaign begins in December 2019, Zovko anticipates no trouble in raising a hundred backers. Already, five hundred names are on the mailing list, so, as Zovko remarks, the target probably "will be quite easy to surpass." The Inkplate is already designed, so the challenge after the funding campaign will be manufacturing. Zovko anticipates shipping the Inkplate by early March 2020 at the latest.

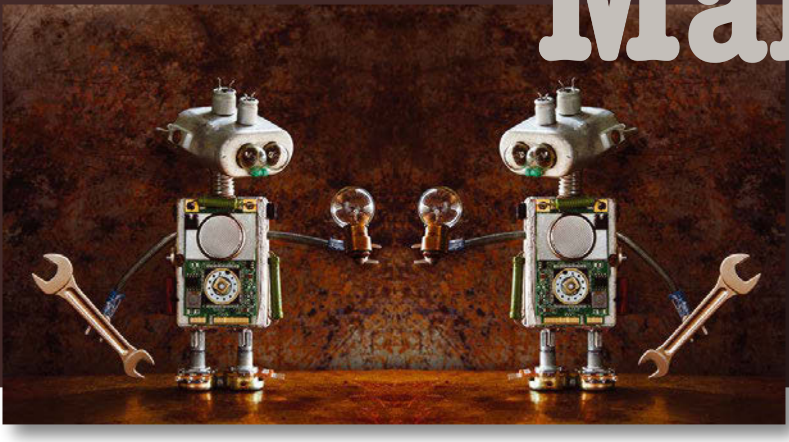
In the future, the limited number of recycled Kindles will sooner or later mean that Inkplate will need to be redesigned for other devices. Meanwhile, Zovko says, "Inkplate 6 is a good starting point for us to get known to the market and find out what customers are looking for. There are many possible uses, and with feedback from this campaign, we hope that Inkplate will become a series of products with different screen sizes and features while keeping simplicity of use as a main feature."

If all goes as planned – and there is little reason to doubt that it will – Inkplate seems likely to fill one of the gaps in open hardware and encourage the release of still other open devices. ■■■

Info

[1] Inkplate 6: *https://www.crowdsupply.com/e-radionica/inkplate-6*

[2] e-radionica.com: *https://e-radionica.com/en/*



MakerSpace

Simulate Raspberry Pi add-on hardware

Soft Pi

Python and tk_tools let you create software versions of Raspberry Pi mini-displays, LED keypads, and NeoPixel hardware. *By Pete Metcalfe*

Raspberry Pis have some great hardware options for displaying information or accepting input. You can use either specialty plates that mount directly on top of the Rasp Pi or a variety of wired components.

Although nothing beats using real hardware for projects, if you're missing the hardware or you'd like to duplicate a value remotely, then a soft version of the hardware can be very useful. In this article, I look at three examples of hardware simulated with the help of Python and the *tk_tools* Python library [1]: a seven-segment wired display, an LCD keypad, and a NeoPixel string.

Seven-Segment Display

A seven-segment display uses seven horizontal and vertical bars, familiar in clocks, meters, and other electronic devices, to represent numbers and letters. The displays are often based on the HT16K33 [2] or TM1637 [3] chipset (Figure 1).

The *tk_tools* Python library [1] contains a soft component for a seven-segment display that can save you writing code from scratch. To install the module, enter:

```
pip install tk_tools
```

The `tk_tools` seven-segment component can function like a TM1637 or HT16K33 display component, with support for various heights, digit colors, and background color. Listing 1 is an example

Listing 1: Show Pi CPU Temperature

```
import tkinter as tk
import tk_tools

root = tk.Tk()
root.title("CPU Temp")

# Create a 7 segment display that is yellow on black with
  5 digits
ss = tk_tools.SevenSegmentDigits(root, digits=5,
     background='black', digit_color='yellow', height=100)
ss.grid(row=0, column=1, sticky='news')

# Update the Pi CPU Temperature every 1 second
def update_gauge():
    # Get the Raspberry CPU Temp
    tFile = open('/sys/class/thermal/thermal_zone0/temp')
    # Scale the temp from milliC to C
    thetemp = int(float(tFile.read())/1000)
    # Show the CPU temp on the 7 segment display
    ss.set_value(str(thetemp))
    root.after(1000, update_gauge)

root.after(1000, update_gauge)
root.mainloop()
```

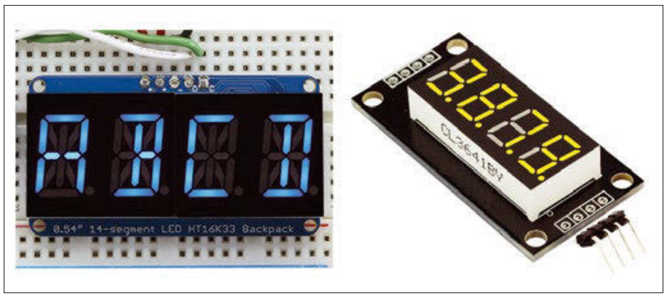


Figure 1: HT16K33 (left) and TM1637 (right) seven-segment displays.

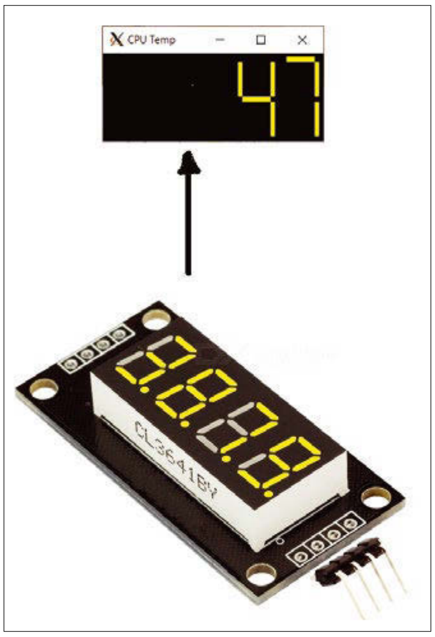


Figure 2: Simulated seven-segment display.

that displays the Rasp Pi's CPU temperature (Figure 2). After creating the seven-segment display object (line 8), simply call `set_value` (line 18) to display the updated Pi temperature value.

LCD Keypad

LCD keypad plates [4] have five or six configurable buttons and a 2x16-character LCD display. These keypads are ideal in small projects for which you want some local control. My daughters and I have used the LCD keypad plates on a number of Pi projects, such as FM radios (Figure 3) and streaming music players.

The *tk_tools* library doesn't have an LCD keypad component, but I found that it is easy to simulate with the standard Python Tkinter library. For my example, I tried to replicate the look and feel of the Pi plate that I had, but you could enhance or change it to meet your requirements.

Listing 2 is a Python LCD keypad example that displays key presses (Figure 4). Lines 13-17 create a single label with white on blue text. The grid's `rowspan` and `columnspan` properties (line 18) create a label two lines high that spans the entire window. The label automatically handles the line wrap to the second line, or a new line character (`\n`) forces text to the second line.

A common function, `myfunc` (line 4), is called when a button is pushed. In this example, the buttons pass their button text to `myfunc`, which then shows the custom text message in the two-line display.

Listing 2: Simulated Pi LCD Keypad

```
01 import tkinter as tk
02
03 # myfunc will show show on the screen the button that is pushed
04 def myfunc(action):
05    print ("Requested action: ",action)
06    Line1.config(text = "Requested action: \n" + action)
07
08 root = tk.Tk()
09 root.title("LCD Keypad Shield")
10 root.configure(background='black')
11
12 # Create a 2 line label that spans the window
13 Line1 = tk.Label(root,
14      text="ADC key testing     \nRight Key OK        ",
15      fg = "white",
16      bg = "blue",
17      font = "Courier 45", borderwidth=4, relief="raised")
18 Line1.grid(row = 0, column = 0, columnspan =15, rowspan = 2)
19
20 # Create 5 buttons and have them call myfunc to show some reback text
21 selectB = tk.Button(root, width=10,text= "SELECT",bg='silver',
   command = lambda: myfunc("SELECT"), relief="raised")
22 selectB.grid(row = 3,column = 0)
23
24 leftB = tk.Button(root, width=10,text= "LEFT", bg='silver',
   command = lambda: myfunc("LEFT"), relief="raised")
25 leftB.grid(row = 3,column = 1)
26
27 topB = tk.Button(root, width=10, text= "UP", bg='silver',
   command = lambda: myfunc("UP"), relief="raised")
28 topB.grid(row = 2,column = 2)
29
30 rightB = tk.Button(root, width=10,text= "DOWN", bg='silver',
   command = lambda: myfunc("DOWN"), relief="raised")
31 rightB.grid(row = 3,column = 3)
32
33 bottomB = tk.Button(root, width=10,text= "RIGHT", bg='silver',
   command = lambda: myfunc("RIGHT"), relief="raised")
34 bottomB.grid(row = 4,column = 2)
35
36 root.mainloop()
```

NeoPixels

NeoPixels [5] are addressable full-color RGB LEDs that come in a variety of arrangements, such as LED strings (Figure 5), matrix arrays, Pi plates, and a variety of sewable components that can be used on wearable products. NeoPixels were

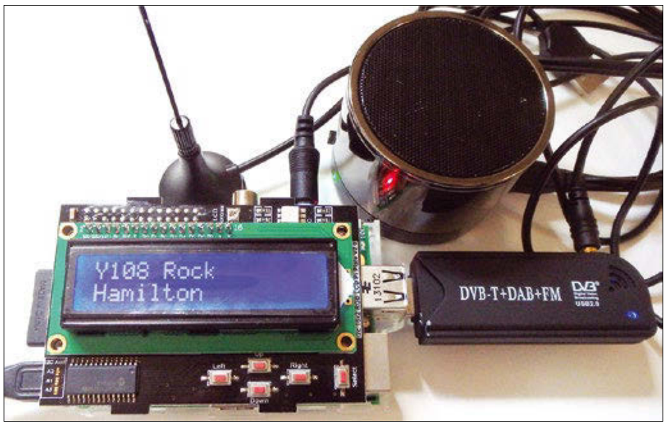


Figure 3: LCD keypad used on a Raspberry Pi FM radio project.

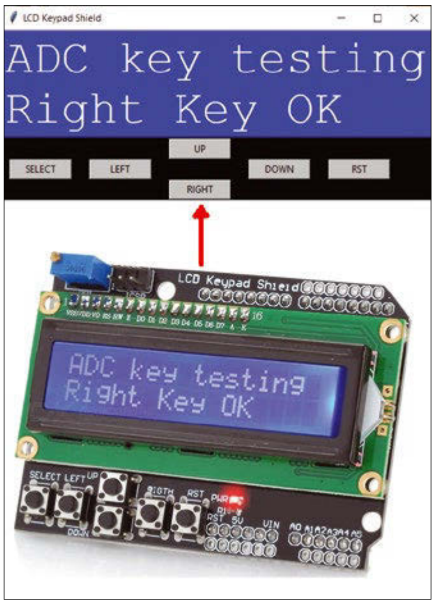

ADC key testing
Right Key OK

Figure 4: Simulated Raspberry Pi LCD keypad.

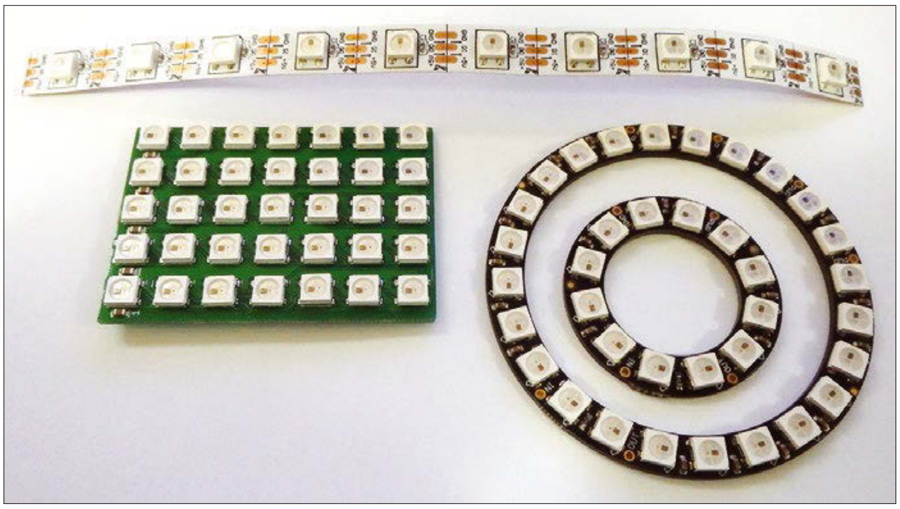


Figure 5: Some NeoPixel hardware.

first available only for Arduino projects, but now, Python libraries for Raspberry Pis are available, as well.

Listing 3 is a Python Tkinter example that simulates 10 NeoPixels in a string arrangement (Figure 6). The array object `ledstrip` (line 10) becomes a number of colored Tkinter labels (line 13), which are placed sequentially in a grid to form a string (line 14). However, you could also arrange the labels into a matrix or a circle. An LED color is set with the `configure(background= 'red')` command (line 17).

Listing 3: Simulated NeoPixels

```
01 import tkinter as tk
02
03 root = tk.Tk()
04 root.title("Soft NeoPixel Strip")
05
06 numleds = 10 # Have 10 neopixels in the strip
07
08 # Create an array object that can be used as Tkinter labels
09
10 ledstrip = ['' for i in range(numleds)]
11
12 for i in range(numleds):
13     ledstrip[i] = tk.Label(root,relief='raised',width=5,height=2,background='white')
14     ledstrip[i].grid(row = 0, column = i) # position the labels in a horizontal row
15
16 # As an example, set one 'neopixel' red
17 ledstrip[5].configure(background= 'red')
18
19 root.mainloop()
```

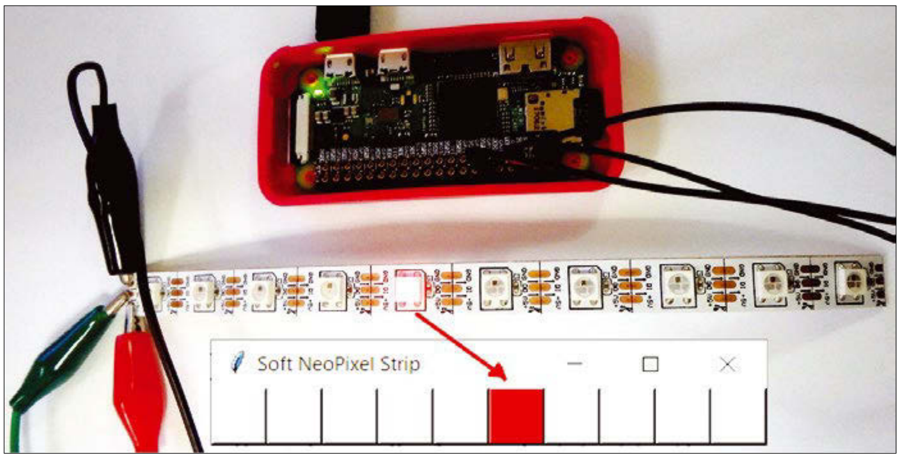


Figure 6: Simulated (bottom) and real (middle) NeoPixels.

Summary

As I mentioned at the top of the article, nothing beats using real hardware. However, if you're on a budget or you just want to do some playing around, then creating Python soft components is a great option. ∎∎∎

Author

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

Info

[1] tk_tools: *https://github.com/slightlynybbled/tk_tools*

[2] Adafruit Python HT16K33 hardware driver: *https://pypi.org/project/adafruit-circuitpython-ht16k33*

[3] Raspberry Pi Python TM1637 LED hardware driver: *https://pypi.org/project/raspberrypi-python-tm1637/*

[4] Pi LCD keyboard plate: *https://learn.adafruit.com/adafruit-16x2-character-lcd-plus-keypad-for-raspberry-pi*

[5] NeoPixels: *https://learn.adafruit.com/neopixels-on-raspberry-pi*

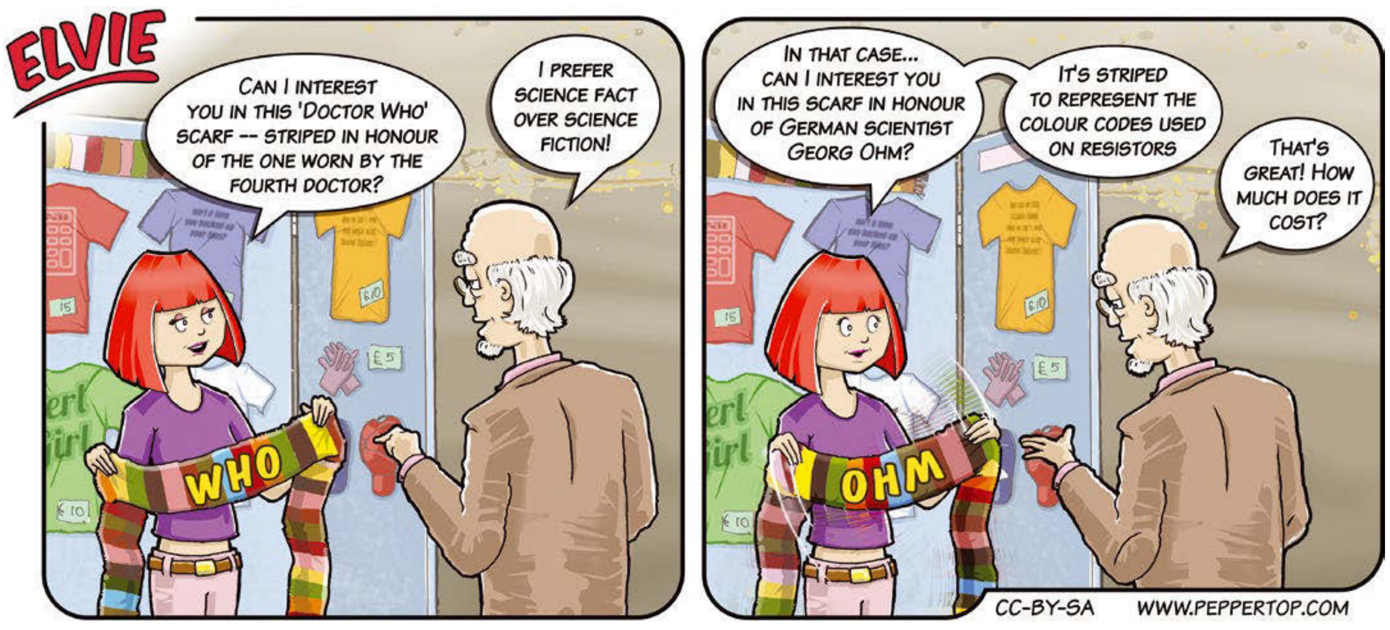
[6] More projects by Pete: *https://funprojects.blog/*

Push notification services are a popular option for users who want to receive updates and stay informed. A cloud-based push service receives push requests and forwards the notices to a client app running on an Android or iPhone. But what if you don't want to depend on the cloud – for reasons of privacy or reliability? Gotify is a push notification tool that helps you keep it local. Set up the Gotify server on your Linux system, and you can use it to forward notices to a client app on your Android phone.

And speaking of local, you don't really need to pay homage to Amazon in order to enjoy the convenience of ebooks. The Calibre ebook manager is a free tool for organizing and managing your ebook collection – and it's all open source!

Image © Olexandr Moroz, 123RF.com

LINUXVOICE


ELVIE
CAN I INTEREST YOU IN THIS 'DOCTOR WHO' SCARF -- STRIPED IN HONOUR OF THE ONE WORN BY THE FOURTH DOCTOR?
I PREFER SCIENCE FACT OVER SCIENCE FICTION!
WHO
IN THAT CASE... CAN I INTEREST YOU IN THIS SCARF IN HONOUR OF GERMAN SCIENTIST GEORG OHM?
IT'S STRIPED TO REPRESENT THE COLOUR CODES USED ON RESISTORS
THAT'S GREAT! HOW MUCH DOES IT COST?
OHM
CC-BY-SA
WWW.PEPPERTOP.COM

MADDOG'S DOGHOUSE

Maddog ponders how the number of women in programming has changed over the course of his career and is pleased to see more women coming back into the tech workspace. BY JON "MADDOG" HALL

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

In Support of Women Programmers

"Mamas don't let your babies grow up to be cowboys"
– Waylon Jennings and Willie Nelson
"It's a girl my lord in a flatbed Ford ..."
– Eagles "Take It Easy"

I've been thinking about these two songs, one of which has been playfully rewritten to be "coders" instead of "cowboys," and the other which, seemed to show astonishment that a female would be driving a "flatbed Ford" truck through the streets of Winslow, Arizona.

I have always loved both songs, but they seem a little out of date by modern standards. Today we would not be surprised to see cowgirls as well as cowboys out on the range or women driving flatbed trucks. But what about the programming profession?

When I started working with computers many programmers were women. Perhaps it was the fact that many mathematicians were women, and they naturally fit into the math-oriented world of computational logic, but by my estimate, 40 percent of the programmers back in those days were women.

At my first job, we had many women programmers and middle managers, perhaps because Aetna Life and Casualty was a progressive company even then, and we had a fairly diverse group of employees for 1973.

When I started teaching at Hartford State Technical College from 1977 to 1980, about 40 percent of the students were women, including a woman who enrolled in her late fifties, Florence Grebe.

Ms. Grebe had taken courses every time her husband, a military man, had been transferred, and by the time she met me she had three shoeboxes full of transcripts. I realized that she was two courses away from a BS in math, two courses from a BS in physics, and two courses away from a BS in business. With my recommendation, she took two courses at the University of Connecticut, got her BS in math, and then enrolled for an MSCS at the Hartford Graduate Center. She went from no degree to an MSCS in a little over one year.

When I went to Bell Laboratories in North Andover, Massachusetts, in 1980, I also found women programmers, researchers, operators, and supervisors. My hiring supervisor at Bell was a very smart woman named Beatrice Fink, and I respected her greatly.

It was in 1983 when I went to Digital Equipment Corporation that the percentage of women started to drop, with the added division of men as "developers" and women as "documentation people." Likewise there were a lot of women product managers, but fewer in middle management above that.

I was never able to put my finger on the reason for the drop off until recently. It is suggested that when the home PC and gaming systems came out that boys were given the PCs and gaming systems and girls were given dolls and "girl things." Of course this was not always true, but it fits a lot of what I saw.

I was lucky enough to meet Rear Admiral Grace Murray Hopper, generally known as the first modern-day programmer. She was brilliant and a true leader. I still treasure the foot-long wire she gave me as a representation of a "nanosecond."

A few years ago, I started going to "Campus Party" computer technology events, which started in Spain and have now spread through Latin America and the rest of Europe. At first, these events drew mostly male attendees with a smattering of women, but over time, more and more females came. Today I am happy to say that the mix is almost 50/50, and hopefully these young people will flow out into the tech workplace.

Not meaning to sound prejudiced, but some of my best managers were women. Most of the time they would just point me in a direction and not try to micromanage me. They would listen to what I had to say and trust that I knew what I was talking about. I am very glad to see more women come back into the tech workspace.

Recently, I was at a conference in Buenos Aires, Argentina, called Nerdear.la and there were several women- and diversity-oriented groups there, including LinuxChix, which tries to get women into tech areas, and Women Who Code, which specifically tries to get women to program.

Around 1995, a friend of mine started LinuxChix and created the website of *linuxchix.org*. She realized that someone had registered the top-level domain (TLD) of *linuxchix.com* and had set up a porn site, which (of course) was distressing to her.

Fortunately Linux International had just won a legal settlement where the Linux trademark was assigned to Linus. After one short telephone call to Linus and one short letter from our attorney, that website disappeared forever.

Now you know the rest of the story ... ■■■

Get push notifications with Gotify

Push Yourself

Replace proprietary cloud-based push notification services with a self-hosted open source notification solution.

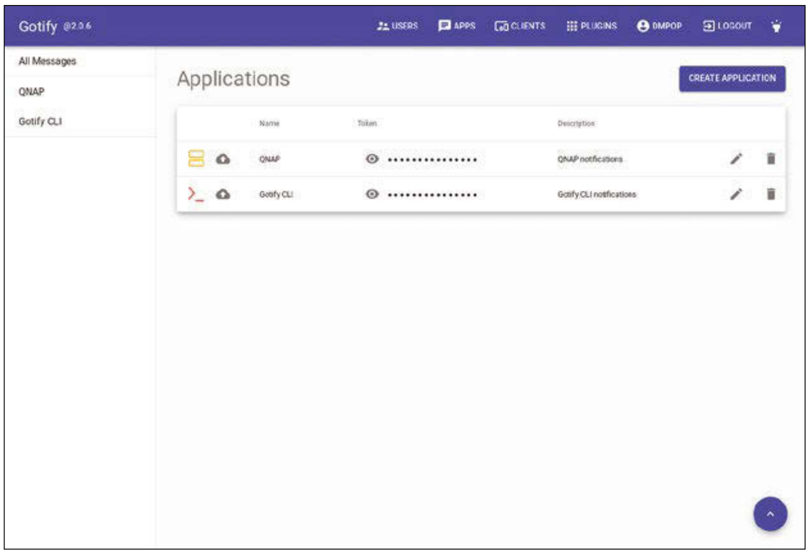
BY DMITRI POPOV

Push notifications can come in useful in many situations – from informing you that a backup job has been completed successfully to reminding you of upcoming deadlines. A push notification system usually consists of two parts: a cloud-based service and a client. The service receives and processes push notification requests received through the service's API and then sends them to the client app running on an Android or iOS device. Using any of the popular cloud-based push notification services brings a familiar set of problems. You are relying on a third party for processing your data, you remain at the mercy of a for-profit enterprise, and usually you have to pay for the services rendered.

But if you are willing to host your own notification system, Gotify [1] has got you covered. This open source solution allows you to roll out your own notification service with a minimum of effort and zero cost.

Installing Gotify

Gotify is based on the server/client model, and there are three pieces of the puzzle you need to take care of:



Name	Token	Description
QNAP	•••••••••••••••	QNAP notifications
Gotify CLI	•••••••••••••••	Gotify CLI notifications

Figure 1: You can set up multiple applications, or profiles, in Gotify.

1. Install and configure the Gotify server.
2. Install the Gotify Android app on your device.
3. Use the Gotify command-line interface (CLI) tool or cURL to send push notifications.

The server component of Gotify is written in Go, and the project's website [2] provides Linux binaries for the 386, AMD64, ARM7, and ARM64 architectures. This means that you can run the server on practically any Linux machine, including Raspberry Pi. Distributed as a single self-contained executable, the Gotify server requires no installation. To deploy the server on a Raspberry Pi, use the commands in Listing 1 to grab the latest binary from the releases section, unpack the downloaded archive, make the binary file executable, and then execute the server.

By default, the Gotify server runs on port 80 (that's why you need to run it with `sudo`). If you already have another server running on this port, you need to reconfigure Gotify. To do this, grab the `config.yml` example file using the command:

```
wget -O config.yml↩
  https://raw.githubusercontent.com/gotify/↩
  server/master/config.example.yml
```

Open the downloaded file, and replace the default port number with the desired one (e.g., 8080). Save the changes and place the file where the Gotify server executable is located. Then start the Gotify server and access its web UI by pointing a browser to the IP address and port of the machine on which Gotify runs. Log in using the default *admin/admin* username and password.

Listing 1: Gotify on a Raspberry Pi

```
wget https://github.com/gotify/server/
  releases/download/v2.0.6/
  gotify-linux-arm-7.zip
unzip gotify-linux-arm-7.zip
chmod +x gotify-linux-arm-7.zip
sudo ./gotify-linux-arm-7
```

Using Gotify

Gotify's web UI is pretty bare-bones, so finding your way around it is not particularly difficult. The first thing you might want to do is to remove the default user and add a new one. Switch to the *Users* section, press the *Create User* button, specify the desired username and password, and enable the *has administration rights* option. Press *Create*, and you are done. You can then delete the default admin user.

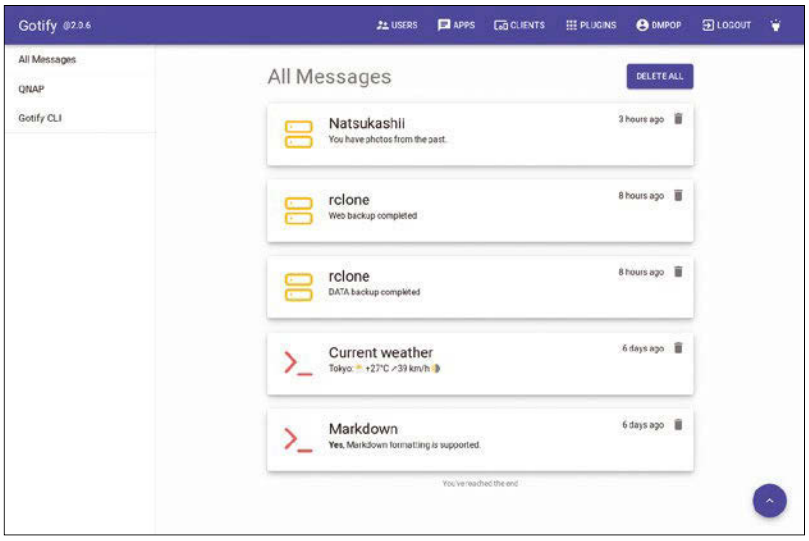
The next step is to create an application. A Gotify application is a profile that makes it possible to identify the source of push notifications via a unique token. For example, if you have a backup shell script running on your server, and this script sends a push notification when a backup job is successfully completed, you can create a profile in Gotify for this specific script. To do this, switch to the *Apps* section, press *Create application*, then provide a name and a short description. Press *Create* and note the generated token. Gotify also allows you to replace the default app icon with a custom one. This can be particularly useful if you have multiple apps, as it lets you visually identify each app (Figure 1). For each app, Gotify creates an entry in the left sidebar, so you can easily view notifications for a specific app.

With the Gotify server up and running, you are ready to send your first notification. There are several ways to do this. The most straightforward one is to use the cURL tool. Since practically all mainstream Linux distributions come with this tool preinstalled, you don't need to install and configure any additional software for sending notifications. More importantly, using cURL makes it easier to integrate push notification functionality into shell scripts.

Pushing a notification with cURL is a matter of running the following command (replace `IPADDRESS:PORT` with the actual IP address and port of the Gotify server and `TOKEN` with the token of the app you created on the Gotify server):

```
curl -X POST ⮯
"https://IPADDRESS:PORT/message?token=TOKEN" ⮯
  -F "title=This is a title" -F ⮯
  "message=Message goes here"
```

If everything works, you should see the received notification in the web UI (Figure 2). Of course, checking for new notifications using Gotify's web UI is not particularly practical. In most scenarios, you'd want to receive notifications instantly on your Android device. For that, you need to install the Gotify Android client either from the Google Play Store or F-Droid (just search for Gotify). Launch the Android app, enter the IP address of the Gotify server, provide your username and password, and press the *Login*



All Messages

Natsukashii
You have photos from the past. 3 hours ago

rclone
Web backup completed 8 hours ago

rclone
DATA backup completed 8 hours ago

Current weather
Tokyo: ☀ +27°C ↗39 km/h 🌓 6 days ago

Markdown
Yes, Markdown formatting is supported. 6 days ago

You've reached the end

Figure 2: You can view and manage incoming notifications using Gotify's web UI.

button (Figure 3). To prevent Android from killing the app, disable the battery optimization for the Gotify app. With the Gotify app running, you can receive and manage notifications on your Android device (Figure 4).

cURL is not the only tool that you can use to send notifications. In fact, the Gotify project provides a dedicated CLI tool for the job. Similar to

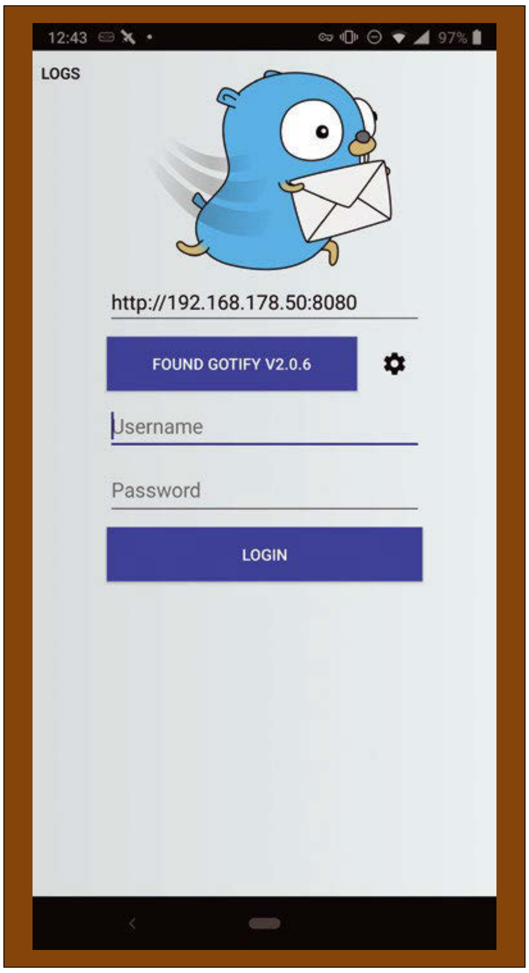


Figure 3: The Gotify Android app connects to the Gotify server.

Listing 2: PHP Push Notifications

```
<?php
    $data = [
    "title"=> $argv[1],
    "message"=> $argv[2],
        "extras" => [
        "client::display" => [
            "contentType" => "text/markdown"
        ]
    ]
];

$data_string = json_encode($data);

$url = "http://127.0.0.1:8080/message?token=AQiPyW7ATHGWgZg";

$headers = [
    "Content-Type: application/json; charset=utf-8"
];

$ch = curl_init();
curl_setopt($ch, CURLOPT_URL, $url);
curl_setopt($ch, CURLOPT_POST, 1);
curl_setopt($ch, CURLOPT_HTTPHEADER, $headers );
curl_setopt($ch, CURLOPT_RETURNTRANSFER, true );
curl_setopt($ch, CURLOPT_POSTFIELDS, $data_string);

$result = curl_exec($ch);
$code = curl_getinfo($ch, CURLINFO_HTTP_CODE);

curl_close ($ch);

switch ($code) {
case "200":
        echo "Message submitted";
        break;
    case "400":
        echo "Bad request";
        break;
    case "401":
        echo "Unauthorized error: invalid token";
        break;
    case "403":
        echo "Forbidden";
        break;
    case "404":
        echo "API URL not found";
        break;
    default:
        echo "Something went wrong or HTTP status code is missing";
}
?>
```

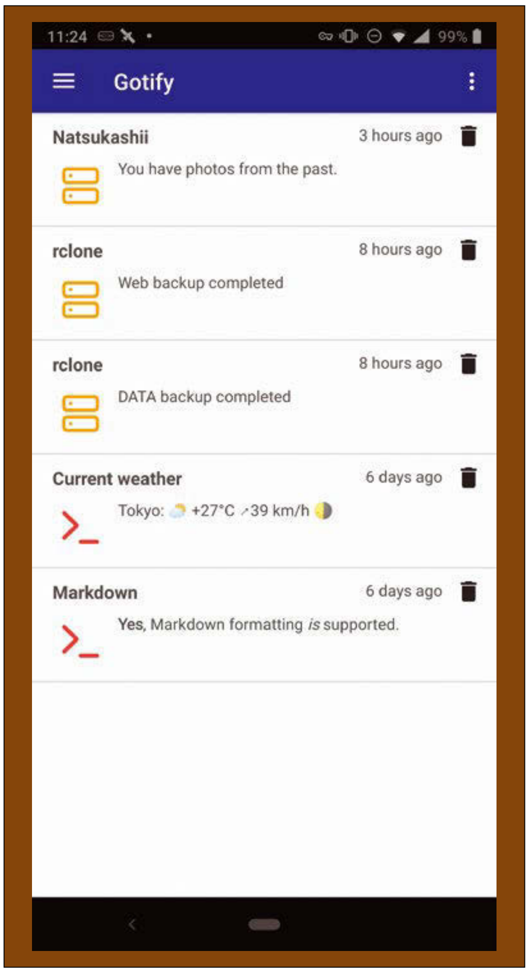
the server component, the CLI utility is distributed as a self-contained binary that requires no installation. Simply grab the latest release of the tool from the project's website [3], make the binary executable, and move it to the `usr/bin` directory using the command:

```
mv gotify /usr/bin/gotify
```

With the Gotify CLI tool, you don't have to specify the required information (IP address, port, and token) every time you send a notification. Instead, run the `gotify init` command, and the tool will guide you through a simple process of creating a configuration file containing all the necessary information. Once you've done that, sending a notification is as easy as running:

```
gotify push "Message goes here"
```

The Gotify CLI tool supports piping, so you can send a command's output as a notification message. The following example fetches weather conditions using the wttr.in service [4] for the specified city and pipes the output to the Gotify tool that pushes the data as a notification.



Natsukashii — 3 hours ago
You have photos from the past.

rclone — 8 hours ago
Web backup completed

rclone — 8 hours ago
DATA backup completed

Current weather — 6 days ago
Tokyo: +27°C ↗39 km/h

Markdown — 6 days ago
Yes, Markdown formatting *is* supported.

Figure 4: The Android app lets you view and manage notifications.

```
curl wttr.in/Tokyo?format="%l:+%c+%t+%w+%m" ⮐
  | gotify push
```

Want to send notifications directly from within the Firefox browser? The aptly named Gotify for Firefox add-on [5] allows you to do just that. Better still, you can use the add-on to send the currently viewed page as a notification.

Since pushing a notification to Gotify is done using a standard HTTP request, you can easily integrate notification functionality into scripts written in your preferred scripting language. The example in Listing 2 demonstrates how to send notifications using PHP. The astute reader will probably notice that this PHP script sends Markdown-formatted notifications. To see the script in action, paste the code in the listing into a text file, replace the default value of the `$url` variable with the actual IP address and token, and then save the file under the `gotify.php` name. Make sure that PHP and the *php-curl* package are installed on your system. You can then send a push notification using the following command as an example:

```
php gotify.php "Markdown!" ⮐
  "**Yes**, Markdown formatting _is_ supported."
```

With a little bit of tweaking, you can embed this example PHP code into your own PHP-based web application.

In Conclusion

Gotify is a perfect replacement for existing commercial push notification services. It requires no installation, it has modest requirements, and its web UI makes it easy to administer the server component and manage received notifications. The fact that you can use any tool or scripting language capable of sending HTTP requests to send notifications means that you can add the notification functionality to your scripts and applications with a minimum of effort. ■■■

Info

[1] Gotify: *https://gotify.net/*

[2] Gotify server: *https://github.com/gotify/server/*

[3] Gotify CLI tool: *https://github.com/gotify/cli*

[4] wttr.in: *https://github.com/chubin/wttr.in*

[5] Gotify for Firefox: *https://addons.mozilla.org/firefox/addon/gotify-for-firefox/*

Manage your ebooks

Easy Reading

Calibre can help manage your ebooks by bulk converting files, adding metadata, and making content available across all your devices.

BY NATE DRAKE

In July 2009, thousands of Amazon's Kindle users logged into their devices to find that their purchased copies of George Orwell's *1984* had been erased. Internet pundits quickly drew comparisons between the corporate giant's move and that of "Big Brother" in Orwell's dystopian imagining of England. More recently, in April of this year, Microsoft announced the closure of their own ebook store. Customers lost access to their book collections but were offered a full refund.

The abundance of e-reader devices such as the Amazon Kindle and the Barnes & Noble NOOK has made a lasting impact: In 2018, electronic books made up 25.8 percent of book sales worldwide. Most reading devices are designed to sell content for specific platforms, which has resulted in a number of differing, incompatible ebook formats. For instance, an EPUB book purchased from Barnes & Noble cannot be transferred and opened automatically on an Amazon Kindle. While Kindles support open format MOBI books, Amazon stores your Kindle ebook purchases in its own proprietary AZW. Although the formidable PDF format is compatible with a number of e-readers, it doesn't always work with their features, such as highlighting or sharing text.

It's unlikely the e-publishing industry as a whole will embrace a universal, open format anytime soon. In the meantime, the developer Kovid Goyal has addressed this problem by creating Calibre, a one-stop utility for all your ebook management.

Calibre is compatible with a huge number of devices and most crucially can convert from one ebook to another. This means that if you exchange or upgrade your device you can copy any non-DRM protected books straight over to a new e-reader. The software also supports scraping the Internet for book metadata like blurb and cover images. Readers with multiple devices will enjoy discovering Calibre's library feature that allows other machines on your local network to access your ebook collection. The latest version of Calibre available at this writing (4.1) boasts all new content server capabilities and a much-improved ebook viewer [1].

Setup

Calibre is free and open source software (FOSS), making it available for a number of platforms. You can use it to convert a variety of formats, such as EPUB, MOBI, PDF, DOCX, ODT, PRC, PDB, PML, RB, and RTF, among others.

Linux users benefit from an auto install script, which you can load by opening a terminal and running

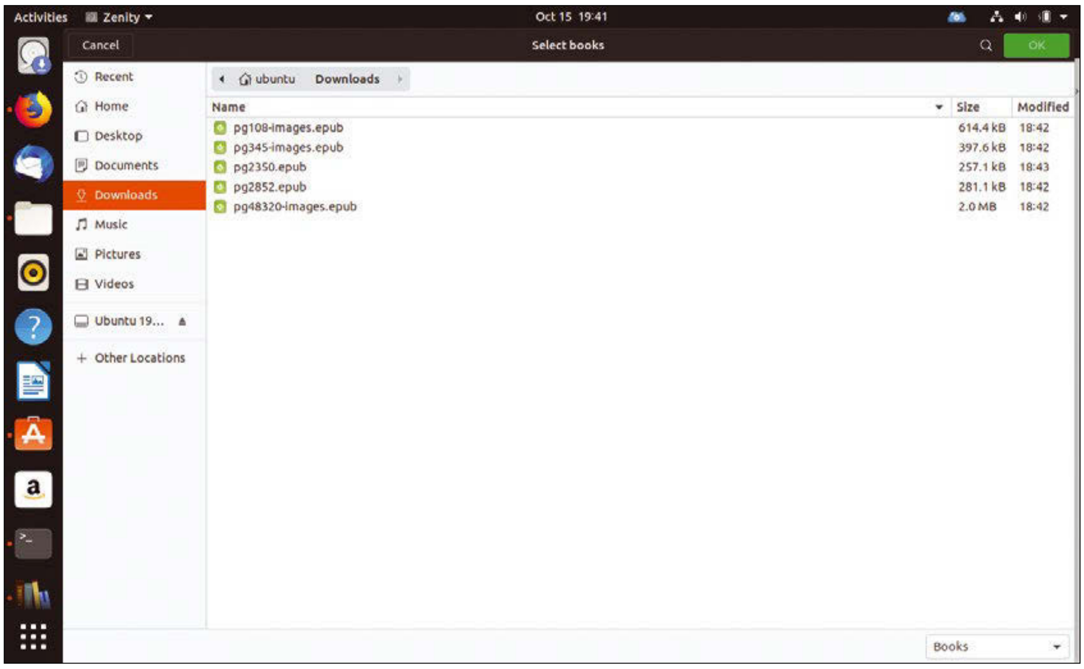
```
sudo -v && wget -nv -O- ⮯
  https://download.calibre-ebook.com/⮯
  linux-installer.sh | sudo sh /dev/stdin
```

On first launch, the Calibre Welcome Wizard appears. Start by choosing where you would like your books to be stored on your computer. If you are unhappy with the default settings, click *Change* to edit. Once you have chosen your preferred location, just click *Next*. The following screen asks you to choose your e-reader, for example, Amazon Kindle Oasis 3. Select *Generic* under *Device* and *Manufacturer*. Once this is done, Calibre is ready to use.

Add Your Books

Although Calibre doesn't support DRM protected content, it can open and convert a wide variety of ebook formats. There are a huge number of public domain books no longer under copyright available from websites such as Project Gutenberg [2] and Internet Archive [3]. However, copyright varies from jurisdiction to jurisdiction. For example, George Orwell's *1984* is no longer under copyright in Australia, but it is still protected in the US.

To add books already on your computer, simply click *Add Books* at the top left of the screen. Scroll through your folders to select the books you want



Name	Size	Modified
pg108-images.epub	614.4 kB	18:42
pg345-images.epub	397.6 kB	18:42
pg2350.epub	257.1 kB	18:43
pg2852.epub	281.1 kB	18:42
pg48320-images.epub	2.0 MB	18:42

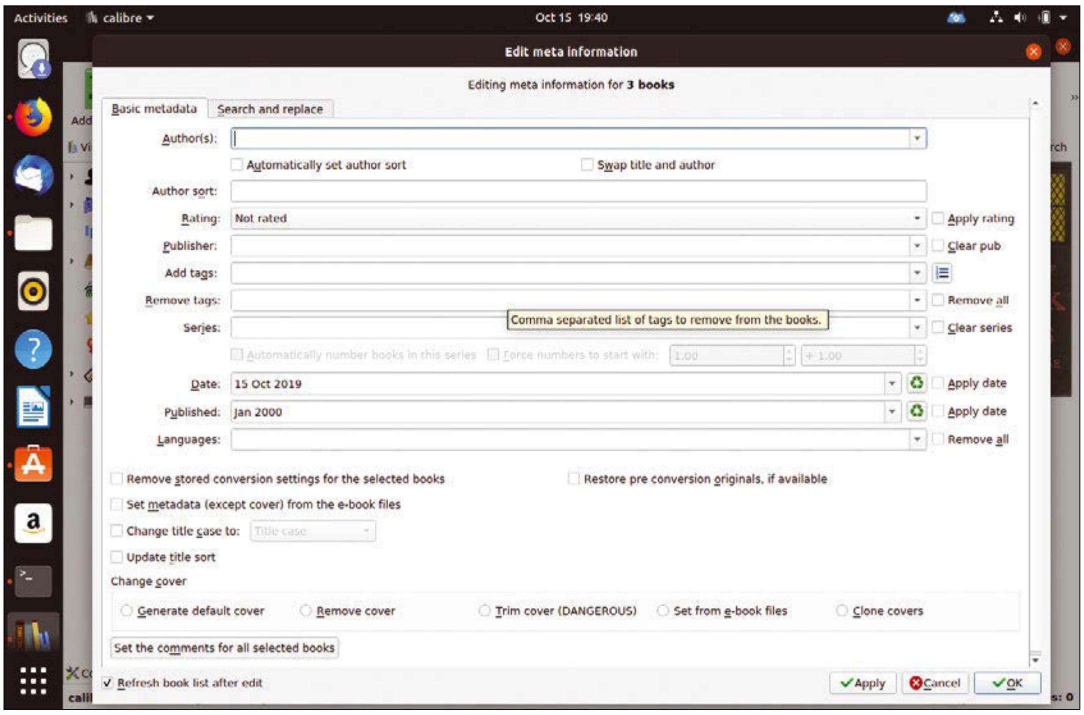
Figure 1: Scroll through your folders and select books to add to Calibre's library.

to upload (Figure 1). Click on the item, and it will be added to Calibre's library. It can now be located on Calibre's main screen.

Missing Metadata

The program automatically displays any existing book metadata, such as title, author, and cover illustrations. However metadata can often be missing or incomplete. You can rectify this by clicking on the *Edit Metadata* button at the top of the screen. You can now edit the author's name, book title, and series (Figure 2). If you wish, you can also change the book's rating, the tags associated with this particular novel, and publication dates. The book cover can also be incorrect, which can be changed from this screen. You can browse for new covers; trim or delete the current cover; and download or upload a new cover.

If you have a number of books that need updating, instead of entering the information manually, choose *Edit Metadata | Download Metadata*. Calibre will now search websites such as Google and Amazon for your chosen titles. In most cases, you'll



Editing meta information for **3 books**

Author(s):
Automatically set author sort
Swap title and author
Author sort:
Rating: Not rated — Apply rating
Publisher: — Clear pub
Add tags:
Remove tags: — Remove all
Comma separated list of tags to remove from the books.
Series: — Clear series
Automatically number books in this series — Force numbers to start with: 1.00 — + 1.00
Date: 15 Oct 2019 — Apply date
Published: Jan 2000 — Apply date
Languages: — Remove all
Remove stored conversion settings for the selected books
Restore pre conversion originals, if available
Set metadata (except cover) from the e-book files
Change title case to: Title case
Update title sort
Change cover
Generate default cover — Remove cover — Trim cover (DANGEROUS) — Set from e-book files — Clone covers
Set the comments for all selected books
✓ Refresh book list after edit

Figure 2: Edit or add metadata to your titles, such as author, rating, and publication dates.

see different editions of the same book. Choose the match that seems most relevant to you and click *Next*. You can select from a number of book covers that Calibre scrapes from various online book sellers. Choose your preferred cover and click *OK*.

Convert Books

Calibre supports conversion to and from a number of ebook formats. Choose the title you wish to convert, and then click on the *Convert Books* icon at the top of the screen. The pop-up box that appears displays all the information for the book you are converting (Figure 3). This includes the input format, title, author, publisher, and tags (if any). Select the output format by clicking on the drop-down button at the top right of the screen. The left-hand side of the screen displays options you may wish to edit, such as metadata, page setup, table of contents, heuristic processing, etc.

There may be times where you need to convert a lot of different books to the same output format. You don't have to go through this process for each book manually. Use your mouse to select all your chosen titles, and then click on the drop-down button beside the *Convert Books* icon. Choose *Bulk Convert* to start the conversion process. Jobs currently being processed can be viewed via the spinning wheel at the bottom right-hand side of the app window.

Get Books

Calibre can search through all the websites that sell ebooks and will not only find the books you want, but also will show the different prices for each. Not only will prices appear from sellers in different currencies, Calibre will also show you the DRM status of each book.

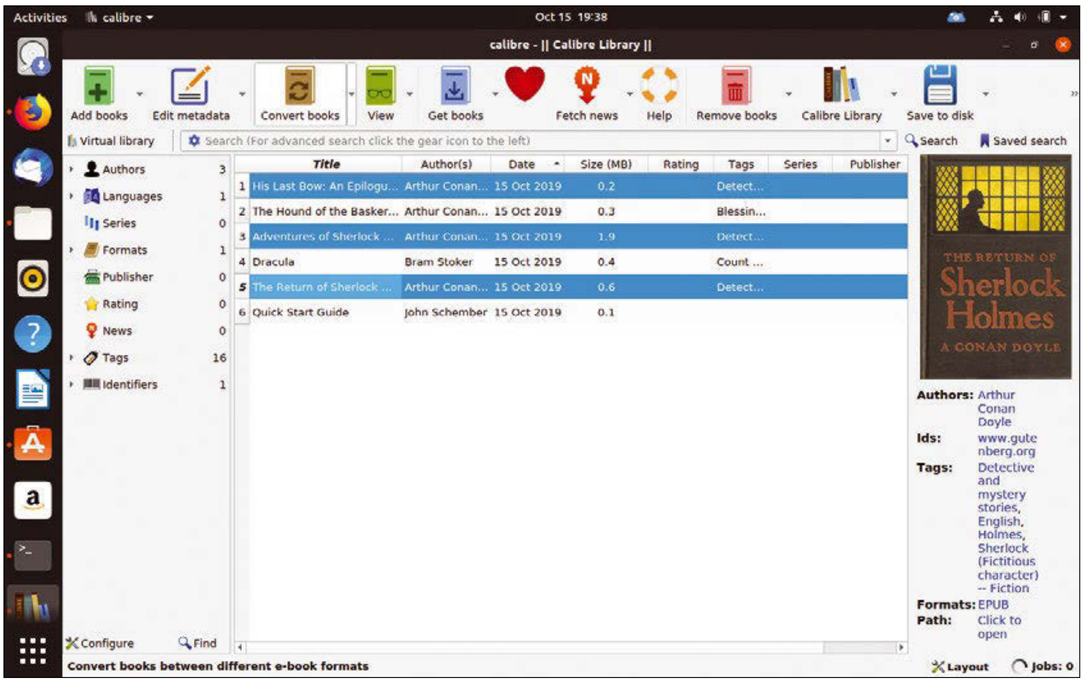
Your Device

When you connect your e-reader to your computer, Calibre detects and displays it at the top right of your screen. Click on *Device* to see the contents of your e-reader's library. If you want to add books from your Calibre library to your device, just click on the item you want to move and drag and drop over the *Device* icon. If your book isn't in the right format, Calibre will ask if you want to *Auto convert the following books before uploading to the device*. Click *Yes* to begin conversion. The rotating spinner on the bottom right of the Calibre screen lets you see how far your book is into the conversion process.

Calibre's main screen shows which books are on the device and/or on Calibre. Books that are on both are indicated by a green tick. To add books from your device's library, just right click on the item and choose *Add to Library*.

News Updates

Calibre has another neat little feature that lets you keep up to date with the news (Figure 4). Click on the *Fetch News* icon. A pop-up box called *Schedule News Download* will now appear. On the left-hand side pane there is a list of all the different news sources arranged by language and country. For example, when you click on *English* you can see a list of English language news sources. When you click on a news source, you can either choose to download now or schedule a download. If you choose to



	Title	Author(s)	Date	Size (MB)	Rating	Tags	Series	Publisher
1	His Last Bow: An Epilogu...	Arthur Conan...	15 Oct 2019	0.2		Detect...		
2	The Hound of the Basker...	Arthur Conan...	15 Oct 2019	0.3		Blessin...		
3	Adventures of Sherlock ...	Arthur Conan...	15 Oct 2019	1.9		Detect...		
4	Dracula	Bram Stoker	15 Oct 2019	0.4		Count ...		
5	The Return of Sherlock ...	Arthur Conan...	15 Oct 2019	0.6		Detect...		
6	Quick Start Guide	John Schember	15 Oct 2019	0.1				

THE RETURN OF Sherlock Holmes A CONAN DOYLE

Authors: Arthur Conan Doyle
Ids: www.gutenberg.org
Tags: Detective and mystery stories, English, Holmes, Sherlock (Fictitious character) -- Fiction
Formats: EPUB
Path: Click to open

Convert books between different e-book formats

Figure 3: You can bulk convert ebooks into a number of file formats.

schedule a download, you must leave Calibre running in order for this to take place.

Content Sharing

If you use several e-readers, syncing new content manually from Calibre will become tedious very quickly. You can make things much simpler by turning Calibre into a content server. This will make all your content available across all your devices. You can even upload new content to your Calibre library from each device.

To do this, click on the *Connect/Share* icon at the top right of the app. Select *Start Content Server*. Your IP address and port number will now be visible. Make a note of this and go to your device. In your device's browser, enter your IP address and port number. The contents of your Calibre library are now visible.

Splitting Large Volumes

You may have some rather voluminous novels, or some large reference books that you would rather have split into sections. Also, larger volumes take longer to download, and this can cause problems if you are using a slower connection or are not in a WiFi zone. Calibre has a solution to this, you can use the EpubSplit and EpubMerge plugins to split your books. To get started, click on *Preferences*, select *Get plugins to enhance Calibre*. Scroll to *EpubSplit* and choose *Install*. Select the toolbars/menus you want the plugin to be added to and click *OK*. Repeat the same process for *EpubSplit*.

Once this is completed, click on the book you want to split or merge and select the necessary plugin. After this, you just need to choose the sections you want split or merged. This feature can only be used on books in the EPUB format.

Calibre Conundrums

Sometimes the conversion process may not go as smoothly as desired. One of the common reasons is converting to PDF. This is because they are in a fixed size and text template placement format, making it very difficult to determine where one paragraph ends and another starts. Where possible, try to avoid converting out of or into a PDF format. The best formats are those specifically designed for display on e-readers such as MOBI, LIT, AZW, and EPUB, among others.

Calibre offers users a chance to become part of its development community [4]. You can also contribute financially to its development by clicking on the button at the top right of its web page. ■

Info

[1] Download Calibre: *https://calibre-ebook.com/download*

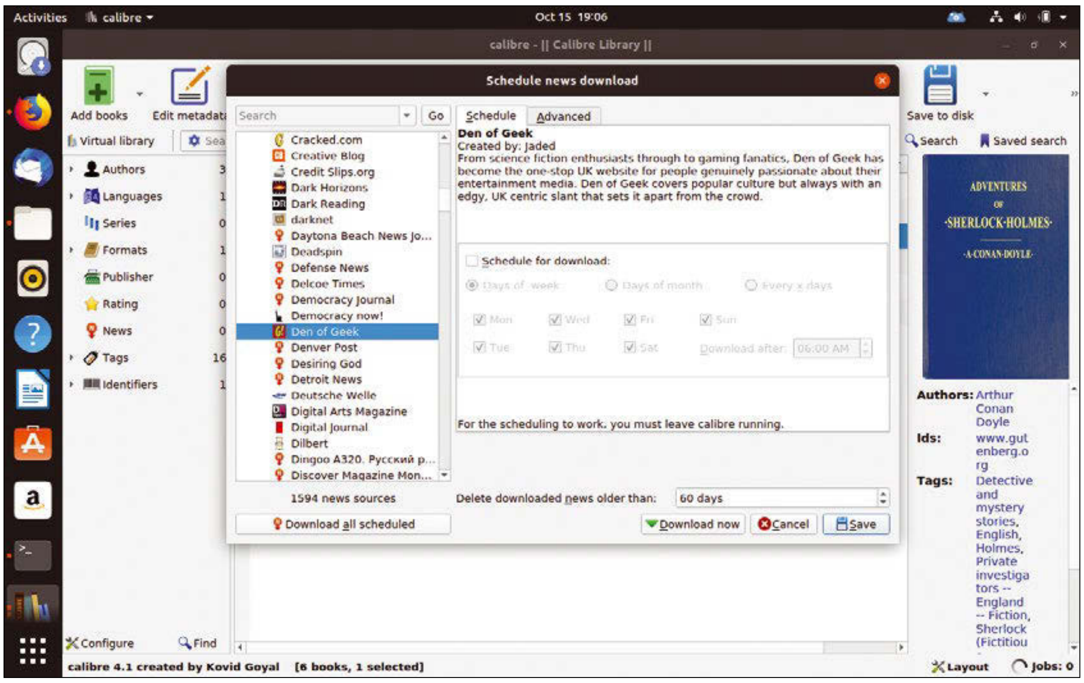
[2] Project Gutenberg: *http://www.gutenberg.org/*

[3] Internet Archive: *https://archive.org/*

[4] Get Involved: *https://calibre-ebook.com/get-involved*

The Author

Nate Drake is a freelance journalist specializing in cybersecurity and retro tech.



Den of Geek
Created by: Jaded
From science fiction enthusiasts through to gaming fanatics, Den of Geek has become the one-stop UK website for people genuinely passionate about their entertainment media. Den of Geek covers popular culture but always with an edgy, UK centric slant that sets it apart from the crowd.

Schedule for download:
Days of week | Days of month | Every x days
Mon | Wed | Fri | Sun
Tue | Thu | Sat | Download after: 06:00 AM

For the scheduling to work, you must leave calibre running.

1594 news sources

Delete downloaded news older than: 60 days

Authors: Arthur Conan Doyle
Ids: www.gutenberg.org
Tags: Detective and mystery stories, English, Holmes, Private investigators -- England -- Fiction, Sherlock (Fictitiou

calibre 4.1 created by Kovid Goyal [6 books, 1 selected]

Figure 4: Not just for books, Calibre lets you choose a news source and schedule article downloads.

FOSSPicks

Sparkling gems and new releases from the world of Free and Open Source Software

This issue's copy was almost late after Graham rediscovered just how immersive and addictive the 1997 game, Blade Runner, can be when recreated on the latest release of ScummVM. BY GRAHAM MORRISON

3D designer/editor/renderer

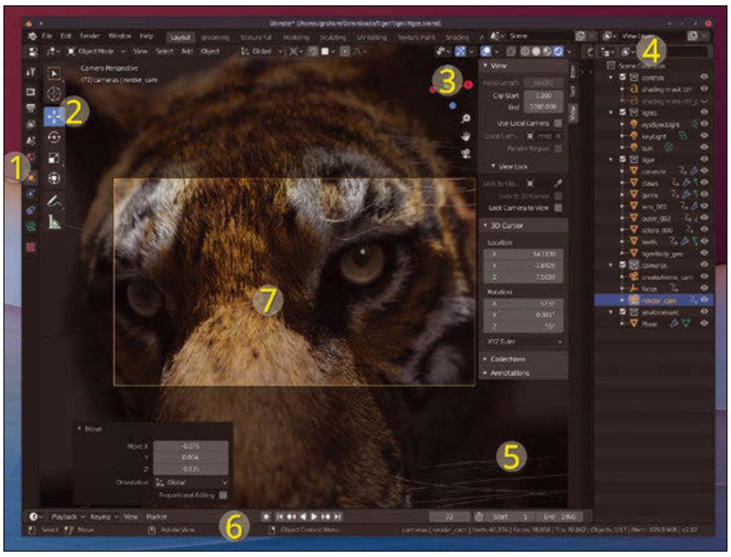
Blender 2.8

Blender really needs no introduction. It's one of the best known open source projects in the world. By showing open source's capabilities, regardless of platform, it has become a flag bearer for the entire movement. All of this is a long way from the uncertainty of 2002's Free Blender campaign, which raised over $100,000 to release the formerly proprietary source code. Blender is now a 3D platform to be reckoned with. In recent months, the project has been given grants worth millions of dollars by Epic Games and NVidia and is celebrating this huge milestone release of Blender 2.8. It has features that compete with some of the best commercial offerings. In some cases, like its functional 3D viewport, it even surpasses them.

Version 2.8 has been a long time coming. So long, that we've already looked at some of its features, such as the Principled BSDF shader (issue 203, October 2017), through their long gestation in Blender 2.7. But it's the GUI overhaul that's going to get the most attention. The old GUI was functional but arcane. In the past, right-click select objects, and users needed to frequently remember keyboard commands when the GUI made it difficult to understand which mode you were working in. All of this made learning to use Blender harder than it should have been, and many casual users would mention this difficulty in relation to its user interface.

When you launch 2.8, you immediately notice the huge UI improvements. Not only has the iconography been redesigned, it's now colorful! The scene collection pops in green and orange, and the various properties actually look tabbed.. The viewport has a brilliant little x, y, and z indicator in red, green, and blue that can be dragged to rotate the view. This means you no longer need to use a keyboard's number pad for setting the view, which is especially useful for those of us with keyboards that lack one. But the main visual upgrade is the toolbar on the left of the viewport. This changes according to context and lets you explore what's possible from your current state. In *Object* mode, for instance, there are icons for moving and transforming. Switch to *Texture Paint* for paint and mask. Switch to *Sculpting* mode and you realize how many options you never knew existed. It makes Blender much more discoverable and easier to learn.

Another huge feature for this release is the Eevee rendering engine. What makes it so different is that it builds scenes on your GPU in real time, much like a game engine. The results are absolutely stunning and are easily the equivalent of a long render time from just a few years ago, but you can now play with the scene, layout, animation, and objects with output good enough for most animations. If you still want to render with the Cycles renderer, you can switch to this without changing any further configuration in the scene. It's seamless, and we've only scratched the surface of what this amazing new release has to offer. If you've ever been put off using Blender because of its perceived complexity, now is the time to try it again.



1. Colors: All the old icons have been redesigned and injected with a little contextual color. **2. Toolbar:** At last! There's now an easily navigable tool palette. **3. 3D navigation:** Easily drag the viewport angle and attitude without using the number pad. **4. Layers:** The grouping code has been completely overhauled. **5. Viewport upgrades:** Many tasks can now be accomplished purely from the main viewport view. **6. Mouse Buttons:** Left mouse click now selects an object! **7. Eevee:** With a decent graphics card, the Eevee renderer can push almost movie-like output to the viewport in real time, which is perfect for editing, animation and material design.

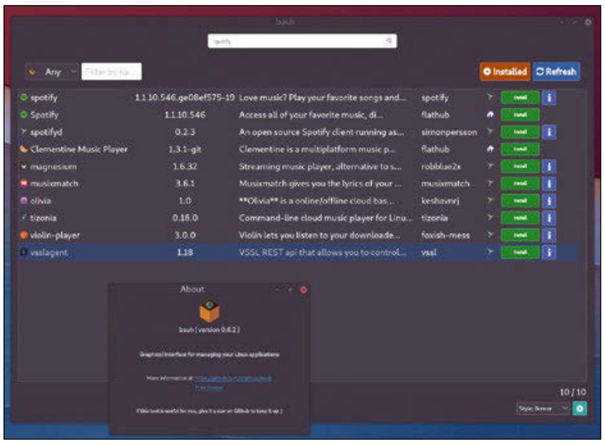
Project Website
https://blender.org

Package manager

Bauh

It's a little surprising that while there are battles being fought on the distro for which packaging format to use, the same revolution hasn't really affected the GUIs used to install them. Gnome's Software Center and KDE's Discovery are slowly incorporating new features, but they've changed very little in recent years. Distro-specific applications like elementary's ApPCenter and Ubuntu's package and Snap stores can feel a little more like "app stores," but there's still plenty of room for other applications to innovate. And Arch Linux, with its multifarious packaging formats and its incredible user package repository, is one of the best places to try out any new ideas, especially one that aims to bridge the brave new worlds of Flatpak, Snap, and the AUR itself.

Bauh (pronounced ba-oo, apparently) attempts to do this by wrapping support for Flatpak, Snap, and AUR packaging types within its own simple discovery and package management UI. The best thing about Bauh is that it's very easy to use, and it makes a refreshing change to the command line or a package manager that has its origins in the crazy world of Debian or RPM dependencies. This is mostly thanks to its excellent design, which will first check and list any installed package for updates before letting you easily switch between whichever back end you prefer, as well as searches for installed and uninstalled packages. Packages can be installed with a simple click; the clean design always makes it clear whether you're installing a Flatpak, Snap, or AUR package, all of which are handled automatically and can be disabled individually if you'd rather not have results littered with AUR packages, for instance. It's obviously early days, but if this simple design continues while the application becomes more complex, Bauh is going to be a brilliant package manager.



spotify	1.1.10.546.ge08ef575-19	Love music? Play your favorite songs and...	spotify
Spotify	1.1.10.546	Access all of your favorite music, di...	flathub
spotifyd	0.2.3	An open source Spotify client running as...	simonpersson
Clementine Music Player	1.3.1-git	Clementine is a multiplatform music p...	flathub
magnesium	1.6.32	Streaming music player, alternative to s...	robblue2x
musixmatch	3.6.1	Musixmatch gives you the lyrics of your ...	musixmatch
olivia	1.0	**Olivia** is a online/offline cloud bas...	keshavnrj
tizonia	0.18.0	Command-line cloud music player for Linu...	tizonia
violin-player	3.0.0	Violin lets you listen to your downloade...	foxish-mess
vsslagent	1.18	VSSL REST api that allows you to control...	vssl

One great little feature of Bauh is that it attaches itself to the system tray, so you can quickly ascertain whether any of your third-party packages need updating.

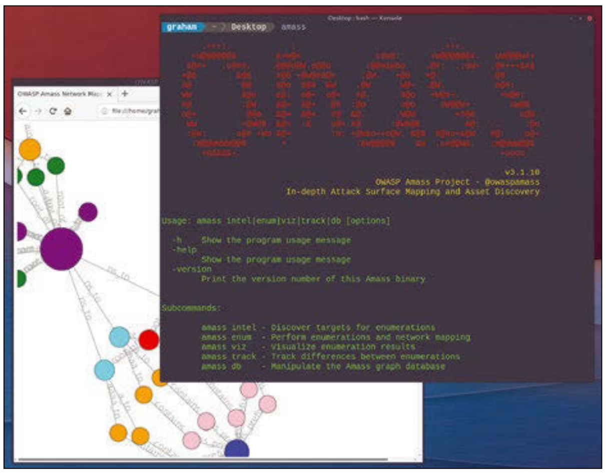
Project Website
https://github.com/vinifmor/bauh

Online security

Amass

The Open Web Application Security Project's Amass project (also known as OWASP Amass) is a serious tool that's been developed to help security experts analyze network traffic to and from a specific domain and its subdomains. It includes data gathering techniques built around open source information gathering that can be used to scan any domain and help identify potential targets, obviously in the hope you can fix them before anyone else uses Amass on your own sites. Its capabilities include basic enumeration and reverse DNS sweeping, certificate tracking, online API use, and access to web archives. A session will typically start with the `intel` argument, which you can use to find out more information about your selected domain. You can then get specific details using `enum` and generate images for analysis with `viz`. With that done, you can monitor changes in your analysis with the `track` command.

All of this control comes from the command line and the `amass` command. At its simplest, you can use `amass` with the `enum` argument for DNS enumeration against a domain name. This will return all the subdomains for a given domain. Similarly, you can use the `net` argument to effectively scan a network range using a CIDR for a slice of the same kind of information across a set of IP addresses. You don't need to be a security expert to get some value from all of this, because Amass can help you probe and better understand all kinds of domain infrastructure. It will also let you go deep into what systems might be exposed and where in a way that isn't otherwise easily achievable, especially with a single tool.



```
                                                              v3.1.10
                                      OWASP Amass Project - @owaspamass
                          In-depth Attack Surface Mapping and Asset Discovery

Usage: amass intel|enum|viz|track|db [options]

  -h      Show the program usage message
  -help
          Show the program usage message
  -version
          Print the version number of this Amass binary

Subcommands:

        amass intel - Discover targets for enumerations
        amass enum  - Perform enumerations and network mapping
        amass viz   - Visualize enumeration results
        amass track - Track differences between enumerations
        amass db    - Manipulate the Amass graph database
```

The HTML visualizations generated by Amass use the D3 JavaScript framework to create complex, beautiful, and bouncily interactive output.

Project Website
https://github.com/OWASP/Amass

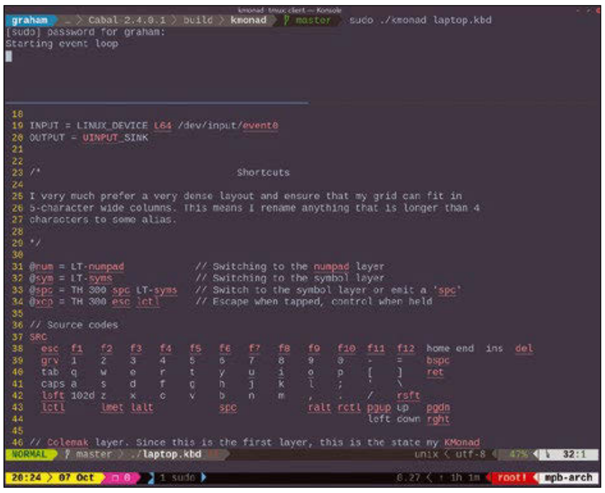
Keyboard configurator

KMonad

A little while ago, we looked at a keyboard configuration utility called Chrysalis that lets you modify the keyboard layout and layer functionality of Kaleidoscope-powered open source keyboards. It's a brilliant tool for people who type a lot, but its effectiveness is restricted to a small set of keyboards, including the Keyboardio Model 01, the Atreus, Dygma's Raise, the ErgoDox EZ, and other keyboards wired like the original ErgoDox. KMonad is another keyboard configuration tool, only this time targeting keyboards using the super-powerful and more populous Quantum Mechanical Keyboard Firmware. It even includes support for the aforementioned ErgoDox EZ. While it officially supports only a handful of keyboards (pun intended), the community maintains support for dozens more.

You've probably guessed by now that KMonad isn't a KDE application. It actually takes its name from the minimal window manager, xmonad, with the implication being that KMonad manages your keyboard rather than your X windows. This kind of DIY attitude toward both the hardware and the firmware obviously requires some serious commitment, which is just as well. KMonad is written in Haskell and requires some serious installation patience to build from source, which is currently the only way to get hold of it. The tool itself is equally minimal on the command line. With the binary in your path, you run it with a single argument pointing to a configuration file. KMonad will then sit between the raw keyboard inputs and the kernel, so you're able to almost completely manipulate and transform the input events before they're passed to your user-level operating system. There's an excellent syntax document included with the package that explains everything you can do with the configuration file, which includes aliases, layers, and macros. It's powerful and tricky but capable of building almost any keyboard configuration you can imagine.

Project Website
https://github.com/david-janssen/kmonad/



```
kmonad : tmux client — Konsole
graham  … 〉 Cabal-2.4.0.1 〉 build 〉 kmonad   master   sudo ./kmonad laptop.kbd
[sudo] password for graham:
Starting event loop


INPUT = LINUX_DEVICE L64 /dev/input/event0
OUTPUT = UINPUT_SINK


/*                                   Shortcuts

I very much prefer a very dense layout and ensure that my grid can fit in
5-character wide columns. This means I rename anything that is longer than 4
characters to some alias.

*/

@num = LT-numpad            // Switching to the numpad layer
@sym = LT-syms              // Switching to the symbol layer
@spc = TH 300 spc LT-syms   // Switch to the symbol layer or emit a 'spc'
@xcp = TH 300 esc lctl      // Escape when tapped, control when held

// Source codes
SRC
  esc  f1   f2   f3   f4   f5   f6   f7   f8   f9   f10  f11  f12  home end  ins  del
  grv  1    2    3    4    5    6    7    8    9    0    -    =    bspc
  tab  q    w    e    r    t    y    u    i    o    p    [    ]    ret
  caps a    s    d    f    g    h    j    k    l    ;    '    \
  lsft 102d z    x    c    v    b    n    m    ,    .    /    rsft
  lctl      lmet lalt           spc            ralt rctl pgup up   pgdn
                                                         left down rght

// Colemak layer. Since this is the first layer, this is the state my KMonad
NORMAL  master 〉 ./laptop.kbd                              unix 〈 utf-8 〈 47% 〈  32:1
20:24 〉 07 Oct  □ 0  1 sudo                               0.27 〈 ↑ 1h 1m 〈 root! 〈 mpb-arch
```

Take complete control over your keyboard by changing its layout, triggering macros, and replacing characters.

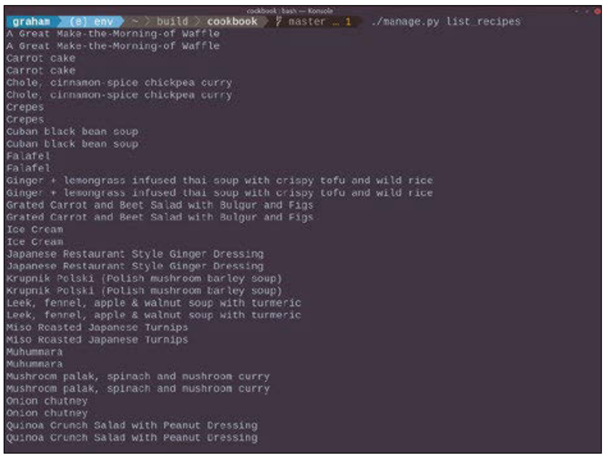
Recipe manager

Cookbook

Cookbook is a surprisingly geeky pastime. Of course, it's a form of chemistry, but it's also a great way of spending time with other people and avoiding the screen. Anything that can help make this easier is surely a good thing, right? Even if that means more screen time? Cookbook is a command-line tool for managing your own recipes. While it does mean more screen time, it's simple enough to not add any additional distractions. This is what makes Cookbook better than following recipes on something like YouTube, where you're beholden to Google's addictive-by-design further watching suggestions that suck hours from your life and potentially lead to burnt cake.

Cookbook includes a selection of recipes to get you started. You can see these with the `list_recipes` command, and they include things like *Japanese Restaurant Style Ginger Dressing* and *Vietnamese green soup*. As with any recipes you eventually add yourself, you can use these as the source of a menu for an evening's entertainment, combining them together with the `add_menu` command. The advantage with this is that the `shop` command can then be used to list all the ingredients you need to buy for every course on a single menu, which is a brilliant idea. When it comes to cooking, simply use the `cook` command to see the instructions for your entire meal. Of course, all of this depends on the quality of the recipes you have access to, and fortunately, Cookbook's best feature is its clear and concise YAML template, which is used to import your own recipes. This is a simple text file with a name, source, tags, notes, a list of ingredients, and the steps to follow. It's a great way of describing everything you need for a meal without over-elaborating. It's missing specific time metadata, which may be useful for each step, but that's something that can either be added manually or added as a worthwhile patch for a future release.

Project Website
https://github.com/cproctor/cookbook



```
cookbook : bash — Konsole
graham  (e) env  ~ 〉 build 〉 cookbook   master … 1   ./manage.py list_recipes
A Great Make-the-Morning-of Waffle
A Great Make-the-Morning-of Waffle
Carrot cake
Carrot cake
Chole, cinnamon-spice chickpea curry
Chole, cinnamon-spice chickpea curry
Crepes
Crepes
Cuban black bean soup
Cuban black bean soup
Falafel
Falafel
Ginger + lemongrass infused thai soup with crispy tofu and wild rice
Ginger + lemongrass infused thai soup with crispy tofu and wild rice
Grated Carrot and Beet Salad with Bulgur and Figs
Grated Carrot and Beet Salad with Bulgur and Figs
Ice Cream
Ice Cream
Japanese Restaurant Style Ginger Dressing
Japanese Restaurant Style Ginger Dressing
Krupnik Polski (Polish mushroom barley soup)
Krupnik Polski (Polish mushroom barley soup)
Leek, fennel, apple & walnut soup with turmeric
Leek, fennel, apple & walnut soup with turmeric
Miso Roasted Japanese Turnips
Miso Roasted Japanese Turnips
Muhummara
Muhummara
Mushroom palak, spinach and mushroom curry
Mushroom palak, spinach and mushroom curry
Onion chutney
Onion chutney
Quinoa Crunch Salad with Peanut Dressing
Quinoa Crunch Salad with Peanut Dressing
```

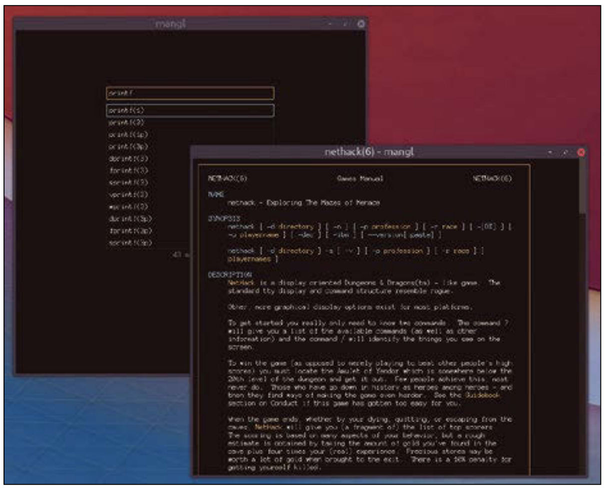
It's simple, but Cookbook is a great management interface for your own recipes and cooking ideas from the command line.

Man page viewer

Mangl

Before all the information was a simple search away, you were dependent on the tools you were using to properly document their own usage. This meant using the `man` command to read the documentation that was always installed alongside an executable, and consequently, the humble Unix man page has been around for a long time. The first was reportedly written by Dennis Ritchie and Ken Thompson at the insistence of their manager in 1971. Since then, especially when first learning about Linux or after installing a new command, typing `man` followed by a command name has been a right of passage for many of us. But in the age of Stack Overflow and Google, many of us have forgotten just how useful man pages can be, especially when it comes to providing some insight into why a certain command works a certain way. Which is why anything that makes your catalog of man pages more accessible and readable is definitely a good thing, and that's what `mangl` does.

When first launched, `mangl` shows nothing but a search box, but as soon as you start typing, a dynamic list of results appears, updating to reflect the page name you're looking for. It's like a private, concise suggest mode. The numbers after the names of the pages represent the manual sections where each page has been placed, such as `1` for user commands, `3` for C library functions and `6` for NetHack. Selecting a page will present the same man page you can see on the command line, but it all feels so much more civilized through its own minimal GUI application. You can still use Vim navigation keys, or your mouse, but pressing Escape will quit the application. This is because it's designed to be a quick replacement to `man` on the command line, rather than a book you can work through. And it does a rather more interesting job.



```
NETHACK(6)                    Games Manual                    NETHACK(6)

NAME
       nethack - Exploring The Mazes of Menace

SYNOPSIS
       nethack [ -d directory ] [ -n ] [ -p profession ] [ -r race ] [ -[DX] ] [
       -u playername ] [ -dec ] [ -ibm ] [ --version[:paste] ]

       nethack [ -d directory ] -s [ -v ] [ -p profession ] [ -r race ] [
       playernames ]

DESCRIPTION
       NetHack is a display oriented Dungeons & Dragons(tm) - like game.  The
       standard tty display and command structure resemble rogue.

       Other, more graphical display options exist for most platforms.

       To get started you really only need to know two commands.  The command ?
       will give you a list of the available commands (as well as other
       information) and the command / will identify the things you see on the
       screen.

       To win the game (as opposed to merely playing to beat other people's high
       scores) you must locate the Amulet of Yendor which is somewhere below the
       20th level of the dungeon and get it out.  Few people achieve this; most
       never do.  Those who have go down in history as heroes among heroes - and
       then they find ways of making the game even harder.  See the Guidebook
       section on Conduct if this game has gotten too easy for you.

       When the game ends, whether by your dying, quitting, or escaping from the
       caves, NetHack will give you (a fragment of) the list of top scorers.
       The scoring is based on many aspects of your behavior, but a rough
       estimate is obtained by taking the amount of gold you've found in the
       cave plus four times your (real) experience.  Precious stones may be
       worth a lot of gold when brought to the exit.  There is a 10% penalty for
       getting yourself killed.
```

It's easy to forget there's a huge set of brilliant documentation already installed on your Linux box – all thanks to the man page.

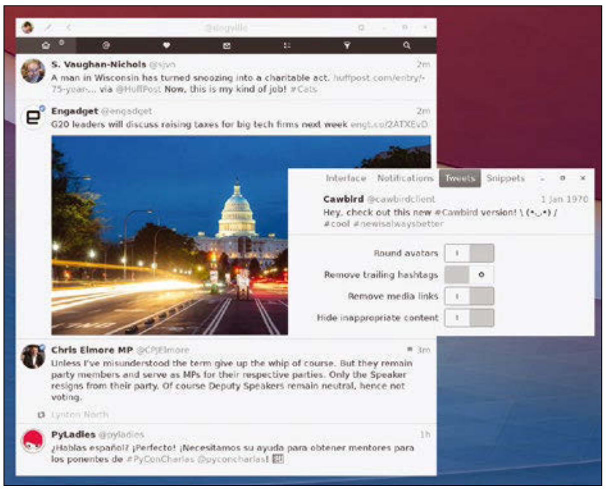
Project Website
https://github.com/zigalenarcic/mangl

Twitter client

Cawbird

Despite all the fear and loathing on social media, especially Twitter, it can still be an informative and insightful platform, especially if you take care to only follow people who interest you. This is particularly true of open source and free software luminaries, many of whom are still happy to share their wisdom in 280 characters or less. The web interface to Twitter is also perfectly acceptable, such as its advanced TweetDeck web UI for the multicolumn view required by power users, but it's always good to see a new native client for the Linux desktop being released. Cawbird is a new client, but it's not an entirely new application. That's because it's a fork of the already well-established Corebird, which fell victim to Twitter removing its User Stream API and replacing it with the new Accounts Activity API. Cawbird forked and successfully made this transition, bringing a slick and modern GTK+ native Twitter client to your Linux desktop.

Cawbird behaves much like any other Twitter client. Its use of Gnome means it has very little window decoration and very few configuration options, although you can change the avatars from their default squares to a circle. It also feels more quick and responsive than either of the web UIs we mentioned, and this means you can fly around your timeline and conversations much more effectively than with a web browser. You can also create lists, web filters, and block lists from the icon tabs that appear at the top of the window. While it would be great to see multicolumn support, this is a rare feature for a desktop client. While we're wishing for features, we'd love to see a dark mode, too. Fortunately, there is excellent spell-checking and even emoji support when you do decide to tweet these requests to the developers.



S. Vaughan-Nichols @sjvn 2m
A man in Wisconsin has turned snoozing into a charitable act. huffpost.com/entry/-75-year-... via @HuffPost Now, this is my kind of job! #Cats

Engadget @engadget 2m
G20 leaders will discuss raising taxes for big tech firms next week engt.co/2ATXEvD

Cawbird @cawbirdclient 1 Jan 1970
Hey, check out this new #Cawbird version! \ (•◡•) / #cool #newisalwaysbetter

Round avatars
Remove trailing hashtags
Remove media links
Hide inappropriate content

Chris Elmore MP @CPJElmore 3m
Unless I've misunderstood the term give up the whip of course. But they remain party members and serve as MPs for their respective parties. Only the Speaker resigns from their party. Of course Deputy Speakers remain neutral, hence not voting.
Lynton North

PyLadies @pyladies 1h
¿Hablas español? ¡Perfecto! ¡Necesitamos su ayuda para obtener mentores para los ponentes de #PyConCharlas @pyconcharlas!

Until Mastodon can take over the world, Twitter still has its uses.

Project Website
https://github.com/IBBoard/cawbird

Procedural sequencer

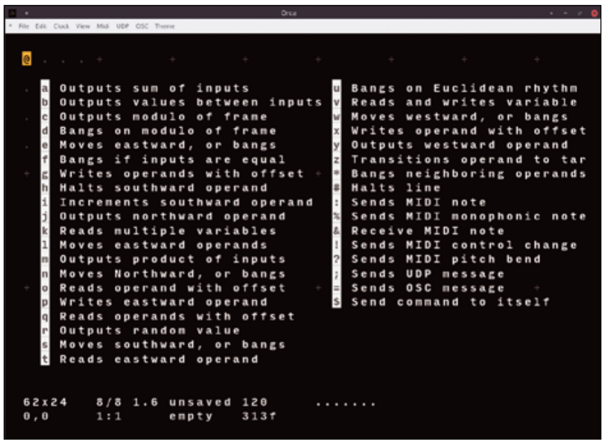
Orca

Orca is something special. But before you realize this, you need to overcome a serious learning curve. Even in the esoteric world of music sequencers, Orca is unique. It's a console application that bills itself as a "programming language designed to create procedural sequencers," and there are definitely elements that could be described as being a programming language. There are operands, loops, counters, conditionals, and registers, for example. But each of these is represented by a single character on your keyboard, from *a* all the way through to *z*, with * and # thrown in for good measure. A few more special characters are used to handle input and output, and Orca can talk to both MIDI- and OSC-compatible software and hardware devices.

However, you don't type any of these character objects into a text editor, save the file, and run it through an interpreter/compiler. Instead, Orca is also a kind of visual development environment for its own language. It starts with a cursor that you move across the default blank canvas of the background. The + symbols on the background break this canvas into smaller grids, and you move the cursor around these too break down into smaller squares, with dots marking the magnified grid in the background. You can effectively zoom in and out of this grid using the square and curly brackets. You then use any of the aforementioned keys to create the object you want at the cursor position, using the grid as a reference to help you position things in musically meaningful places.

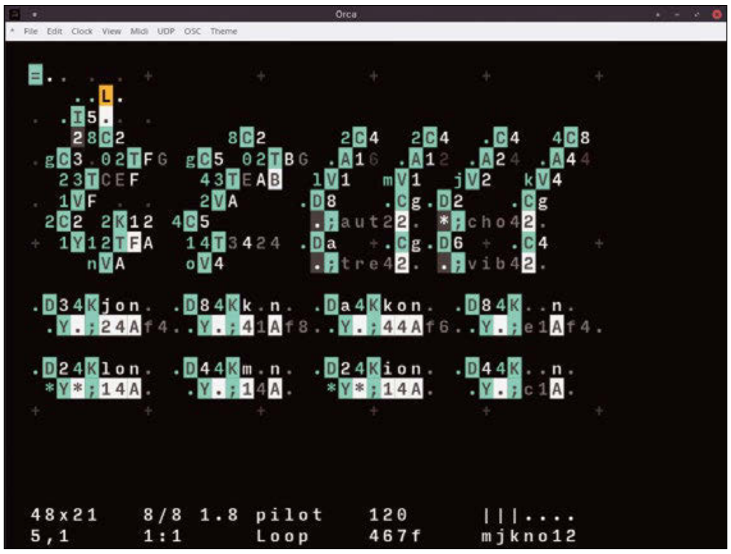
A sequence starts with the creation of an operator by pressing its designated key. The letter *D* is a good place to start. This is a delay function. When it's created, dots appear in positions to its left, right, and bottom. In Orca, these directions are known as west, east, and south, respectively. If you press the spacebar, a counter on the lower screen border starts continually incrementing. Each value here is a frame, or click, where an event can be triggered. At this point, the delay operator's south position will start showing a * every eight clicks. This asterisk is known as a bang, and it's the character used to generate output. To change its timing frequency, you enter a single digit in the operator's east position, and you can use any hex value. The operator uses the modulo of this to generate the bang. The position to the west of the operator is used as a modulation source from other operators, so you can build huge chains of operators and even encapsulate these into functions that you can copy and paste across the canvas – all running concurrently and generating output.

The final step is to add an output operator east of the bang being generated south of the delay. The colon (:) sends a MIDI note. When you create this, five dots appear to its right, which hold values for channel, octave, velocity, note, and length. Don't worry, you don't need to memorize all these special locations, as there's always a hint at the bottom to explain where your cursor is. Provide values for these, and your MIDI synthesizer will play something. Congratulations! We've made a sound, but barely scratched the surface. Which is why Orca really is something special.



```
@. . . +

a Outputs sum of inputs              u Bangs on Euclidean rhythm
b Outputs values between inputs      v Reads and writes variable
c Outputs modulo of frame            w Moves westward, or bangs
d Bangs on modulo of frame           x Writes operand with offset
e Moves eastward, or bangs           y Outputs westward operand
f Bangs if inputs are equal          z Transitions operand to tar
g Writes operands with offset        * Bangs neighboring operands
h Halts southward operand            # Halts line
i Increments southward operand       : Sends MIDI note
j Outputs northward operand          % Sends MIDI monophonic note
k Reads multiple variables           & Receive MIDI note
l Moves eastward operands            ! Sends MIDI control change
m Outputs product of inputs          ? Sends MIDI pitch bend
n Moves Northward, or bangs          ; Sends UDP message
o Reads operand with offset          = Sends OSC message
p Writes eastward operand            $ Send command to itself
q Reads operands with offset
r Outputs random value
s Moves southward, or bangs
t Reads eastward operand

62x24   8/8 1.6 unsaved 120      .......
0,0     1:1     empty   313f
```

Your entire performance can be saved or even inserted into a completely new project.



```
=. .   . .   +          +          +          +          +
    . . L .
 .  . I 5 .  .
    2 8 C 2           8 C 2          2 C 4    2 C 4    . C 4    4 C 8
 . g C 3 . 0 2 T F G  g C 5  0 2 T B G  . A 1 6  . A 1 2  . A 2 4  . A 4 4
   2 3 T C E F          4 3 T E A B   1 V 1    m V 1    j V 2    k V 4
 .  1 V F  .  .         2 V A        . D 8    . C g . D 2      . C g
  2 C 2  2 K 1 2  4 C 5              . ; a u t 2 2 .  * ; c h o 4 2 .
 +  1 Y 1 2 T F A   1 4 T 3 4 2 4  . D a   + . C g . D 6  +  . C 4   +
      n V A         o V 4           . ; t r e 4 2 .  . ; v i b 4 2 .

 . D 3 4 K j o n .  . D 8 4 K k . n .  . D a 4 K k o n .  . D 8 4 K . . n .
  . Y . ; 2 4 A f 4 . . Y . ; 4 1 A f 8 . . Y . ; 4 4 A f 6 . . Y . ; e 1 A f 4 .

 . D 2 4 K l o n .  . D 4 4 K m . n .  . D 2 4 K i o n .  . D 4 4 K . . n .
  * Y * ; 1 4 A .    . Y . ; 1 4 A .   * Y * ; 1 4 A .    . Y . ; c 1 A .
 +          +          +          +          +          +


48x21   8/8 1.8 pilot   120    |||....
5,1     1:1     Loop    467f   mjkno12
```

Orca feels like the evil musical hybrid of an assembler language, NetHack, and Conway's Game of Life.

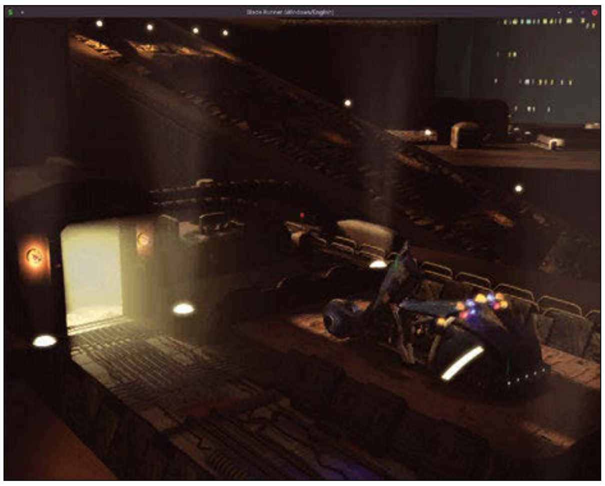
Project Website
https://hundredrabbits.itch.io/orca

Old games emulator

ScummVM 2.1

ScummVM is a little like Blender, in that it's been around for a long time, and most people are aware of it. It's the game engine that first allowed us to replay those old 8- and 16-bit era LucasArts games like Monkey Island 2: LeChuck's Revenge and Maniac Mansion. In fact, the SCUMM in ScummVM represents the original games engine used to write these games, the Script Creation Utility for Maniac Mansion (SCUMM). But if you don't follow the project closely, you won't realize just how many games, and gaming platforms, it's grown to support in the almost two decades since its first release. There are dozens, including the entire King's Quest and Space Quest series, Simon the Sorcerer, Discworld, Starship Titanic, Zork Nemesis, Eye of the Beholder, Broken Sword, and Myst.

This release is no different, but it does add support for a very special PC Windows game that dates all the way to 1997. It's the game Blade Runner, which was a point-and-click adventure inspired by the famous film and developed by Westwood Studios while it took a sabbatical from its Command & Conquer franchise. But the game was far more than a rehash of Ridley Scott's classic. It was a game that ran in parallel to the original storyline, and was figuratively drenched in the same atmosphere and philosophical duplicity of the film, thanks to its wonderful graphics, reinterpreted Vangelis soundtrack, in-game AI, and compelling multifaceted endings. But it also expanded on the film's themes to add new aspects to the story, all within what felt like a living and breathing dystopian future vision of Los Angeles 2019 as seen from 1982. It was also a game that remained very difficult to play on modern hardware, thanks to it being a Windows-only title. But that limitation is finally over – thanks ScummVM!

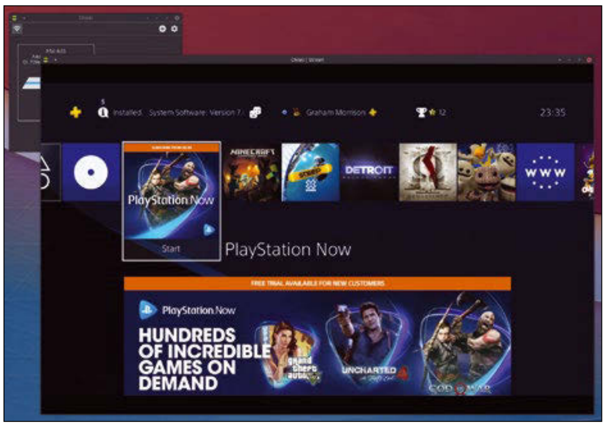

It's fitting that in the year the film *Blade Runner* was set, ScummVM has added support for this hugely enjoyable and influential PC game.

Project Website
https://www.scummvm.org/

PlayStation remote client

Chiaki

Sony has announced the PlayStation 5, but it's some way off, and the PlayStation 4 (PS4) still has plenty of life in it. In many ways, this late stage of a console's life cycle is often the best, with developers knowing how to get the best out of the hardware and a huge, stable number of players to sell games to. It's also a stage where small tools come out of left field that can really upgrade your experience, and that's what Chiaki does. It's a remote client for accessing your PS4 across a network on your Linux box. The idea isn't new, and Sony has been offering its own remote play functionality to accomplish the same thing from the now defunct Vita handheld and PlayStation TV, as well as later clients for Windows and macOS. But there are obviously huge advantages for getting this feature natively, not least because the cooling fans on the PS4 sound like a runway at Heathrow, and it's better if you can get as far away from the console as you can to play your games in quiet.
The client can be easily built or run from an AppImage. When first started, it will hunt around your network for a console, which will hopefully soon appear in the discovered list. To connect to your console, you need to run a Python script that extracts a unique identifier connected to your PSN account from a new web-based account login. You then need to register a new device on your PS4 to get an eight-digit pin and enter all this into the Chiaki client. Fortunately, you only need to do this once. Your remote screen will then open on your Linux box, and you can then use your keyboard or controller to navigate around your PlayStation and play your favorite games. It works brilliantly!



If you have a PS4, Chiaki lets you stream games to your Linux box at 60hz/720p. Pro users can push this to 1080p.

Project Website
https://github.com/thestr4ng3r/chiaki

Build your own Mastodon client
Status Quo

Creating your own clients to interact with your friends in the Fediverse is easy. A bit of Python and an off-the-shelf module will do the trick.

BY PAUL BROWN

Although there are plenty of Mastodon [1] clients out there (as we saw in the prior installment [2]), sometimes you just want that something special to satisfy your needs. Fortunately, Mastodon's API is open and well-organized and there is a Python wrapper [3] that makes it even easier.

Registration

First, you need to install the `Mastodon.py` wrapper. The easiest way to do this is by using `pip`:

```
pip install Mastodon.py
```

or, if `pip` complains about permissions:

```
sudo pip install Mastodon.py
```

Next you want to register your application with the Mastodon network. You can do this from the Mastodon website where you have opened your account. We talked about how to do this in *Linux Magazine*, issue 227 [4].

Or you could do it all from the comfort of your command line with something like what you see in Listing 1.

Line 1 of Listing 1 just sets the interpreter you want to use to run this script (your default Python interpreter).

Listing 1: readtoots_register.py

```
01 #!/usr/bin/env python
02
03 from mastodon import Mastodon
04
05 Mastodon.create_app(
06   'readtoots',
07   scopes=['read'],
08   api_base_url =
       'https://your.mastodon.server',
09     to_file = '.secrets'
10 )
```

Line 3 pulls in the bits of the `mastodon` module you need. Lines 5 through 10 registers the app, giving it a name the server can identify it by (`readtoots` on line 6). On line 7, you tell the server what sort of actions it will be able to carry out (in this case only `'read'` – other actions are `'write'`, `'follow'`, and `'push'`). The `api_base_url` on line 8 tells the application what instance of Mastodon you want to go through. This is the address of a Mastodon server – you would usually use the instance where you have your account for this. Finally, the `to_file` argument tells `readtoots_register.py` where to store the credentials information the server will send back down the line for the application.

Save the file as `readtoots_register.py`, make it executable with

```
chmod a+x readtoots_register.py
```

and run it with:

```
./readtoots_register.py
```

After running it for the first time, nothing apparently happens, but you will find a new `.secrets` file in your app's directory. If you look inside, you will see something similar to Listing 2.

The `Mastodon.create_app()` function shown in Listing 1 not only registers the application, it also returns a `client_id` (line 1 in Listing 2) and a `client_secret` (line 2 in Listing 2) that you will be able to use to identify your application each time it has to interact with the Mastodon instance. It also kindly adds the address of the instance the app is registered with, making it super convenient as you will see later on.

Listing 2: .secrets

```
01 53eM1n9lyHR4nd0mUnUMB3r5vAN6G133TeR5X4g
02 KMOr340FdTh3p54M3A60nTnUkKNOWCOhweMigLcng6U
03 https://your.mastodon.server
```

You will only need to run `readtoots_register.py` once, and, when you're done, you can crack on with `readtoots.py` proper.

Reading Toots

Let's build a command-line Mastodon client that reads, by default, the toots of the account you log into, but that will also let you read toots from any federated account you pass as a parameter to the script.

Listing 3: readtoots.py

```
01 #!/usr/bin/env python
02
03 from mastodon import Mastodon
04 from optparse import OptionParser
05
06 if __name__ == '__main__':
07
08   parser = OptionParser ()
09   parser.add_option ('-u', '--username',
       help = 'your email', dest = 'maUser')
10   parser.add_option ('-p', '--password',
       help = 'your password',
       dest = 'maPassword')
11   parser.add_option ('-a', '--account',
       help = 'account you want to read',
       dest = 'sosAccount', default = 'me')
12
13   (options, args) = parser.parse_args()
14
15   app=Mastodon (
16     client_id = '.secrets'
17     )
18
19   app.log_in (
20     username = options.maUser,
21     password = options.maPassword,
22     to_file = '.token',
23     scopes = ['read']
24     )
25
26   if (options.sosAccount == 'me'):
27     maId = app.me ()['id']
28
29   else:
30     maId = app.account_search (
         options.sosAccount)[0]['id']
31
32   maToots = app.account_statuses (maId)
33
34   for status in maToots:
35     print ('====================
         Status ' + str (status['id']) +
         ' ====================')
36     print (status['content'])
```

As you will also need login credentials to access an account, you are going to need at least three options on the command line. As command line arguments tend to pile up as you add features, let's keep things organized and use the `optparse` Python module to keep things tidy.

Take a look at Listing 3 to see how it all works.

Once you have imported the modules you need (lines 3 and 4) and set up the `OptionParser()` to manage your command line arguments (lines 8 through 13), it is time to activate your application (lines 15 through 17).

See how convenient this is: If you hadn't registered and stored the credentials and the address of the Mastodon instance in `.secrets`, you would have had to hard code them in with the `client_secret`, `client_id`, and `api_base_url` option when you instantiate the `Mastodon` class.

Next, you log into your account (lines 19 through 24). This is not always necessary. In theory, you can have an application read public toots from public accounts without having to log in, but this only works on some Mastodon servers. Turns out mine is not one of them.

Whether logging in is necessary or not will depend on the version of the Mastodon software the server is running and how the administrator has configured it. To be on the safe side, always log in, and your application will not fail.

The logging in itself is straightforward: pass the username and password collected from the command line to Mastodon.py's `log_in()` function and you will get an access token in return (which you can store in a file for when you need it – line 22).

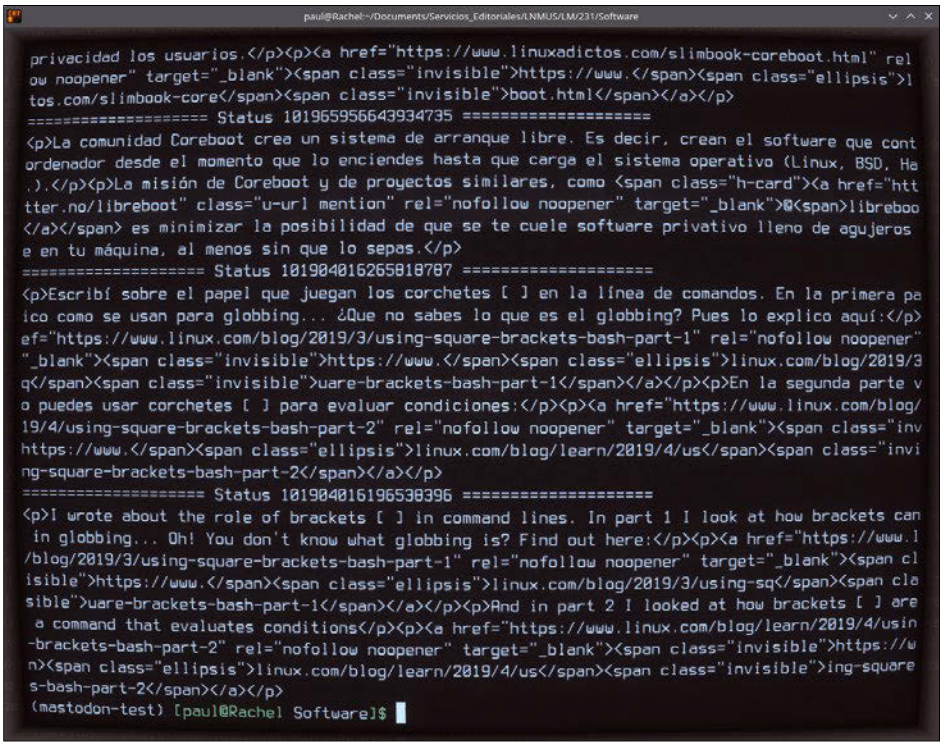
For some reason not explained in Mastodon's API documentation, my server also required the application's scope as defined when registering it (see Listing 1, line 7). Oh well! I added it on line 23 of Listing 3, and everything worked.

Next you get the `id` of the account you want to read. The `id` is not the same as the name of the account (*@someusername@some.mastodon.instance*), but a unique numerical value assigned to each account. There is no easy way to discover an account's id on a Mastodon website, but you can discover what yours is with `Mastodon.py`'s `me()` function (line 27).

The `me()` function returns a Mastodon `user` dictionary. A `user` dictionary contains information about the account. It includes, among other things, the `display_name`, the `username`, when it was created, how many followers it has, and, yes, the account's `id`.

In case the user has decided to peruse the statuses of another account, you can use the `app.account_search()` to find the account passed on by the `-a` option on the command line. As `app.account_search()` returns a list of `user` dictionaries, you have to pick the first one (line 30).

The `account_statuses()` function returns a list of `toot` dictionaries containing all the toots of an account identified by `maId` (line 32). You then loop through them (lines 34 through 36), extract the `content` field (line 36), and print it to the command line.


paul@Rachel:~/Documents/Servicios_Editoriales/LNMUS/LM/231/Software

```
privacidad los usuarios.</p><p><a href="https://www.linuxadictos.com/slimbook-coreboot.html" rel
ow noopener" target="_blank"><span class="invisible">https://www.</span><span class="ellipsis">l
tos.com/slimbook-core</span><span class="invisible">boot.html</span></a></p>
==================== Status 101965956643934735 ====================
<p>La comunidad Coreboot crea un sistema de arranque libre. Es decir, crean el software que cont
ordenador desde el momento que lo enciendes hasta que carga el sistema operativo (Linux, BSD, Ha
.).</p><p>La misión de Coreboot y de proyectos similares, como <span class="h-card"><a href="htt
tter.no/libreboot" class="u-url mention" rel="nofollow noopener" target="_blank">@<span>libreboo
</a></span> es minimizar la posibilidad de que se te cuele software privativo lleno de agujeros
e en tu máquina, al menos sin que lo sepas.</p>
==================== Status 101904016265818787 ====================
<p>Escribí sobre el papel que juegan los corchetes [ ] en la línea de comandos. En la primera pa
ico como se usan para globbing... ¿Que no sabes lo que es el globbing? Pues lo explico aquí:</p>
ef="https://www.linux.com/blog/2019/3/using-square-brackets-bash-part-1" rel="nofollow noopener"
"_blank"><span class="invisible">https://www.</span><span class="ellipsis">linux.com/blog/2019/3
q</span><span class="invisible">uare-brackets-bash-part-1</span></a></p><p>En la segunda parte v
o puedes usar corchetes [ ] para evaluar condiciones:</p><p><a href="https://www.linux.com/blog/
19/4/using-square-brackets-bash-part-2" rel="nofollow noopener" target="_blank"><span class="inv
https://www.</span><span class="ellipsis">linux.com/blog/learn/2019/4/us</span><span class="invi
ng-square-brackets-bash-part-2</span></a></p>
==================== Status 101904016196538396 ====================
<p>I wrote about the role of brackets [ ] in command lines. In part 1 I look at how brackets can
 in globbing... Oh! You don't know what globbing is? Find out here:</p><p><a href="https://www.l
/blog/2019/3/using-square-brackets-bash-part-1" rel="nofollow noopener" target="_blank"><span cl
isible">https://www.</span><span class="ellipsis">linux.com/blog/2019/3/using-sq</span><span cla
sible">uare-brackets-bash-part-1</span></a></p><p>And in part 2 I looked at how brackets [ ] are
 a command that evaluates conditions</p><p><a href="https://www.linux.com/blog/learn/2019/4/usin
-brackets-bash-part-2" rel="nofollow noopener" target="_blank"><span class="invisible">https://w
n><span class="ellipsis">linux.com/blog/learn/2019/4/us</span><span class="invisible">ing-square
s-bash-part-2</span></a></p>
(mastodon-test) [paul@Rachel Software]$ 
```

Figure 1: Outputting toots to the command line using your custom-made Mastodon client.

Save the file as `readtoots.py` and make it executable with:

```
chmod a+x readtoots.py
```

You can then run it like this:

```
./readtoots.py -u your@email.com -p ↩
  "Your secret password"
```

where `your@email.com` is the email you used when you registered for your Mastodon account and "`Your secret password`" is the password you use to log into your account.

This will output your account's 20 latest toots to the command line (Figure 1). It doesn't look pretty, but it works.

In the first run, you don't specify any account, so the application goes with the default ("`me`") and prints out the statuses of the account it is logged in to.

Listing 4: readtoots.py (better version)

```
01 #!/usr/bin/env python
02
03 from mastodon import Mastodon
04 from optparse import OptionParser
05
06 if __name__ == '__main__':
07
08   parser = OptionParser ()
09   parser.add_option ('-u', '--username',  help = 'your
                       email',     dest = 'maUser')
10   parser.add_option ('-p', '--password',  help = 'your
                       password',  dest = 'maPassword')
11   parser.add_option ('-a', '--account',   help = 'account
                       you want to read', dest =
                       'sosAccount', default = 'me')
12
13   (options, args) = parser.parse_args()
14
15   app=Mastodon (
16     client_id = '.secrets'
17     )
18
19   app.log_in (
20     username = options.maUser,
21     password = options.maPassword,
22     to_file = '.token',
23     scopes = ['read']
24     )
25
26   if (options.sosAccount == 'me'):
27     maId = app.me ()['id']
28
29   else:
30     maId = app.account_search (options.sosAccount)[0]
         ['id']
31
32   maMaxId = None
33   wannaRepeat = "Y"
34
35   while (wannaRepeat in "YyYesyesYeahyeah"):
36
37     maToots = app.account_statuses (maId, max_id =
                                 maMaxId)
38
39     for status in maToots:
40       print ('==================== Status ' + str
             (status['id']) + ' ====================')
41       print (status['content'])
42
43     maMaxId = status['id']
44     print ('========================================')
45     print ("More?")
46     wannaRepeat = input()
```

If you run `readtools.py` like this:

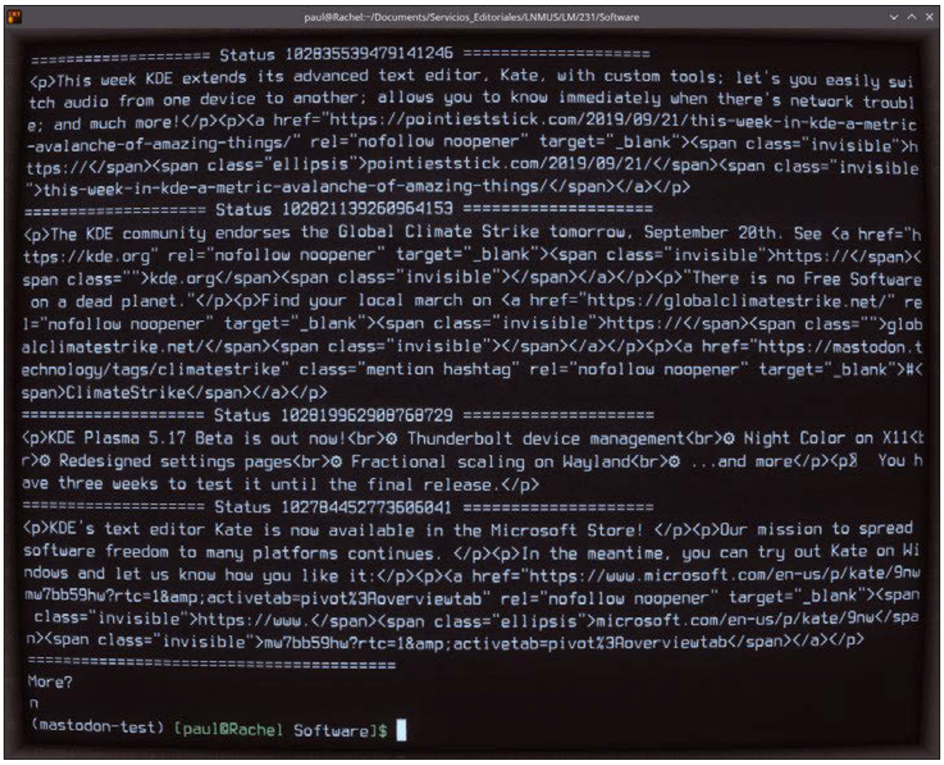
```
./readtoots.py -u your@email.com -p↵
  "Your secret password" -a ↵
  @kde@mastodon.technology
```

you will see the latest 20 toots from the KDE community account.

I have mentioned several times that, by default, all you get is the 20 latest statuses. You can push up that limit to a maximum of 40 by adding `limit = 40` as an argument in `account_statuses()` on line 32, but that is still a paltry feed if ever there was one. Who posts only 40 statuses and stops there? Incidentally, this is a limitation of the Mastodon API, not of `Mastodon.py`. You can get past this limitation, however. Take a look at Listing 4.

The trick is using `account_statuses()`'s `max_id` argument (line 37). All you have to do is put the `for` loop inside a `while` loop (lines 35 through 46) and collect the last status id after the `for` loop exits (line 43). Then you ask the user if they want to continue (lines 45 and 46) and, if they say *Yes* (line 35), use the status id in `account_statuses()` and start over.

When you run `readtoots.py` now, it prints out the 20 latest toots and then prints *More?*; if the user enters *Y*, *Yes*, or *Yeah*, it prints out the next 20 toots and asks again. If the user types in *No*, the application exits (Figure 2).



```
paul@Rachel:~/Documents/Servicios_Editoriales/LNMUS/LM/231/Software
==================== Status 102835539479141246 ====================
<p>This week KDE extends its advanced text editor, Kate, with custom tools; let's you easily swi
tch audio from one device to another; allows you to know immediately when there's network troubl
e; and much more!</p><p><a href="https://pointieststick.com/2019/09/21/this-week-in-kde-a-metric
-avalanche-of-amazing-things/" rel="nofollow noopener" target="_blank"><span class="invisible">h
ttps://</span><span class="ellipsis">pointieststick.com/2019/09/21/</span><span class="invisible
">this-week-in-kde-a-metric-avalanche-of-amazing-things/</span></a></p>
==================== Status 102821139260964153 ====================
<p>The KDE community endorses the Global Climate Strike tomorrow, September 20th. See <a href="h
ttps://kde.org" rel="nofollow noopener" target="_blank"><span class="invisible">https://</span><
span class="">kde.org</span><span class="invisible"></span></a></p><p>"There is no Free Software
 on a dead planet."</p><p>Find your local march on <a href="https://globalclimatestrike.net/" re
l="nofollow noopener" target="_blank"><span class="invisible">https://</span><span class="">glob
alclimatestrike.net/</span><span class="invisible"></span></a></p><p><a href="https://mastodon.t
echnology/tags/climatestrike" class="mention hashtag" rel="nofollow noopener" target="_blank">#<
span>ClimateStrike</span></a></p>
==================== Status 102819962908768729 ====================
<p>KDE Plasma 5.17 Beta is out now!<br>⚙ Thunderbolt device management<br>⚙ Night Color on X11<b
r>⚙ Redesigned settings pages<br>⚙ Fractional scaling on Wayland<br>⚙ ...and more</p><p>  You h
ave three weeks to test it until the final release.</p>
==================== Status 102784452773606041 ====================
<p>KDE's text editor Kate is now available in the Microsoft Store! </p><p>Our mission to spread
software freedom to many platforms continues. </p><p>In the meantime, you can try out Kate on Wi
ndows and let us know how you like it:</p><p><a href="https://www.microsoft.com/en-us/p/kate/9nw
mw7bb59hw?rtc=1&amp;activetab=pivot%3Aoverviewtab" rel="nofollow noopener" target="_blank"><span
 class="invisible">https://www.</span><span class="ellipsis">microsoft.com/en-us/p/kate/9nw</spa
n><span class="invisible">mw7bb59hw?rtc=1&amp;activetab=pivot%3Aoverviewtab</span></a></p>
========================================
More?
n
(mastodon-test) [paul@Rachel Software]$ 
```

Figure 2: Scroll through the whole shebang with the new and improved `readtoots.py`!

Improvements

You will immediately notice that statuses come laced with HTML tags. To make them more readable, you could strip them out. Or, even better, use the HTML to format the text as the author intended. Modern terminal emulators accept colored, bold, and highlighted text, even emojis. Most also allow for "live" links the user can click.

Another thing is that images, videos, and other media files included in toots are lost. To be able to see them, you would have to design a client with a graphical interface, with boxes that display pictures and clips. Implementing a client with these features goes well beyond the scope of this article.

Conclusion

As mentioned elsewhere, creating a basic Mastodon reader is dirt simple with `Mastodon.py`. Making it attractive is another matter, but that is like everything: Your mileage will vary according to how far you want to go and how many features you want to pack into your client.

However, there is one thing you must always be aware: Different nodes in the Mastodon network implement different versions of the Mastodon server software. And different versions implement slightly different features or implement similar features in slightly different ways. Check what version your instance is running with `Mastodon.py`'s `Mastodon.instance()` function [5].

Also bear in mind that the administrator can enable or lock down certain features, so don't expect all the API's features to work everywhere.

In the next issue, we'll see how to build an application that lets you post toots and even schedule toots for later posting. See you then! ■■■

Info

[1] Mastodon: *https://joinmastodon.org/*

[2] "Tutorial – Mastodon Clients" by Paul Brown, *Linux Magazine*, issue 228, November 2019, pp. 92-95

[3] Mastodon.py documentation: *https://mastodonpy.readthedocs.io/en/stable/index.html*

[4] "Welcome to the Fediverse" by Paul Brown, *Linux Magazine*, issue 227, October 2019, pp. 90-94: *http://www.linux-magazine.com/Issues/2019/227/Tutorial-Fediverse*

[5] Mastodon.instance: *https://mastodonpy.readthedocs.io/en/stable/index.html#reading-data-instances*

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 *http://linux-magazine.com/events.*

If you know of another Linux event you would like us to add to our calendar, please send a message with all the details to *events@linux-magazine.com*.

36C3 - Chaos Communication Congress

Date: December 27-30, 2019

Location: Leipzig, Germany

Website: *https://events.ccc.de/*

The Chaos Computer Club e. V. (CCC) is Europe's largest association of hackers. For the 36th time, the CCC will host the Chaos Communication Congress between Christmas and New Year's Eve. Around 17,000 participants are expected from December 27-30, 2019, including more than 2,000 volunteers.

FOSDEM '20

Date: February 1-2, 2020

Location: Brussels, Belgium

Website: *https://fosdem.org/2020/*

FOSDEM is a free and non-commercial event organized by the community for the community to promote the widespread use of free and open source software. Join thousands of developers from all over the world for this two-day event.

SCaLE 18x

Date: March 5-8, 2020

Location: Pasadena, California

Website: *https://www.socallinuxexpo.org/scale/18x*

SCaLE is the largest community-run, open source and free software conference in North America. It is held annually in the greater Los Angeles area. SCaLE 18x expects to host 150 exhibitors, along with nearly 130 sessions, tutorials, and special events.

Events

IT Tage 2019	December 9-12	Frankfurt, Germany	https://www.ittage.informatik-aktuell.de/
Node+JS Interactive 2019	December 11-12	Montreal, Canada	https://events19.linuxfoundation.org/events/nodejs-interactive-2019/
Kubernetes Forum Sydney 2019	December 12-13	Sydney, Australia	https://events19.linuxfoundation.org/events/nodejs-interactive-2019/
Open Source Forum 2019	December 16	Tokyo, Japan	https://events19.linuxfoundation.org/events/open-source-forum-2019/
36C3 - Chaos Communication Congress	December 27-30	Leipzig, Germany	https://events.ccc.de/
FOSDEM 2020	February 1-2	Brussels, Belgium	https://fosdem.org/2020/
Kubernetes Forum Bengaluru	February 17-18	Bengaluru, India	https://events19.linuxfoundation.org/events/kubernetes-forum-bengaluru-2020/
Kubernetes Forum Delhi	February 20-21	Delhi, India	https://events19.linuxfoundation.org/events/kubernetes-forum-delhi-2020/
Software Architecture Conference	February 23-26	New York, New York	https://conferences.oreilly.com/software-architecture/sa-ny
SCaLE 18x	March 5-8	Pasadena, California	https://www.socallinuxexpo.org/scale/18x
AI Hardware Summit	March 10-11	Munich, Germany	https://www.aihardwaresummiteu.com/events/ai-hardware-summit-europe
Cloud Expo Europe	March 11-12	London, United Kingdom	https://www.cloudexpoeurope.com/
CloudFest 2020	March 14-19	Europa-Park, Germany	https://www.cloudfest.com/
Artificial Intelligence Conference	March 15-18	San Jose, California	https://conferences.oreilly.com/artificial-intelligence/ai-ca
Strata Data Conference	March 15-18	San Jose, California	https://conferences.oreilly.com/artificial-intelligence/ai-ca

CALL FOR PAPERS

We are always looking for good articles on Linux and the tools of the Linux environment. Although we will consider any topic, the following themes are of special interest:

- System administration
- Useful tips and tools
- Security, both news and techniques
- Product reviews, especially from real-world experience
- Community news and projects

If you have an idea, send a proposal with an outline, an estimate of the length, a description of your background, and contact information to *edit@linux-magazine.com*.

The technical level of the article should be consistent with what you normally read in *Linux Magazine*. Remember that *Linux Magazine* is read in many countries, and your article may be translated into one of our sister publications. Therefore, it is best to avoid using slang and idioms that might not be understood by all readers.

Be careful when referring to dates or events in the future. Many weeks could pass between your manuscript submission and the final copy reaching the reader's hands. When submitting proposals or manuscripts, please use a subject line in your email message that helps us identify your message as an article proposal. Screenshots and other supporting materials are always welcome.

Additional information is available at: *http://www.linux-magazine.com/contact/write_for_us.*

NOW PRINTED ON recycled paper from 100% post-consumer waste; no chlorine bleach is used in the production process.

Authors

Erik Bärwaldt	23
Swapnil Bhartiya	8
Jens-Christoph Brendel	16
Paul Brown	92
Zack Brown	12
Bruce Byfield	38, 70
Andy Carlson	44
Joe Casad	3
Mark Crutch	75
Nate Drake	82
Jon "maddog" Hall	77
Jörg Hofmann	26
Charly Kühnast	51
Christoph Langner	32
Vincent Mealing	75
Pete Metcalfe	72
Martin Mohr	16
Graham Morrison	86
Dmitri Popov	48, 78
Mike Schilli	56
Scott Sumner	60
Ferdinand Thommes	52
Jack Wallen	8

Contact Info

Editor in Chief
Joe Casad, jcasad@linux-magazine.com

Copy Editors
Amy Pettle, Megan Phelps

News Editor
Swapnil Bhartiya and Jack Wallen

Editor Emerita Nomadica
Rita L Sooby

Managing Editor
Lori White

Localization & Translation
Ian Travis

Layout
Dena Friesen, Lori White

Cover Design
Lori White

Cover Image
© andreysuslov, 123RF.com

Advertising
Brian Osborn, bosborn@linuxnewmedia.com
phone +49 89 3090 5128

Marketing Communications
Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
2721 W 6th St, Ste D
Lawrence, KS 66049 USA

Publisher
Brian Osborn

Customer Service / Subscription
For USA and Canada:
Email: cs@linuxpromagazine.com
Phone: 1-866-247-2802
(Toll Free from the US and Canada)

For all other countries:
Email: subs@linux-magazine.com

www.linuxpromagazine.com – North America
www.linux-magazine.com – Worldwide

Printed in Nuremberg, Germany by hofmann infocom GmbH on recycled paper from 100% post-consumer waste; no chlorine bleach is used in the production process.

Distributed by Seymour Distribution Ltd, United Kingdom

LINUX PRO MAGAZINE (ISSN 1752-9050) is published monthly by Linux New Media USA, LLC, 2721 W 6th St, Ste D, Lawrence, KS, 66049, USA. Periodicals Postage paid at Lawrence, KS and additional mailing offices. Ride-Along Enclosed. POSTMASTER: Please send address changes to Linux Pro Magazine, 2721 W 6th St, Ste D, Lawrence, KS 66049, USA.

Published monthly in Europe as Linux Magazine (ISSN 1471-5678) by: Sparkhaus Media GmbH, Zieblandstr. 1, 80799 Munich, Germany.

Approximate

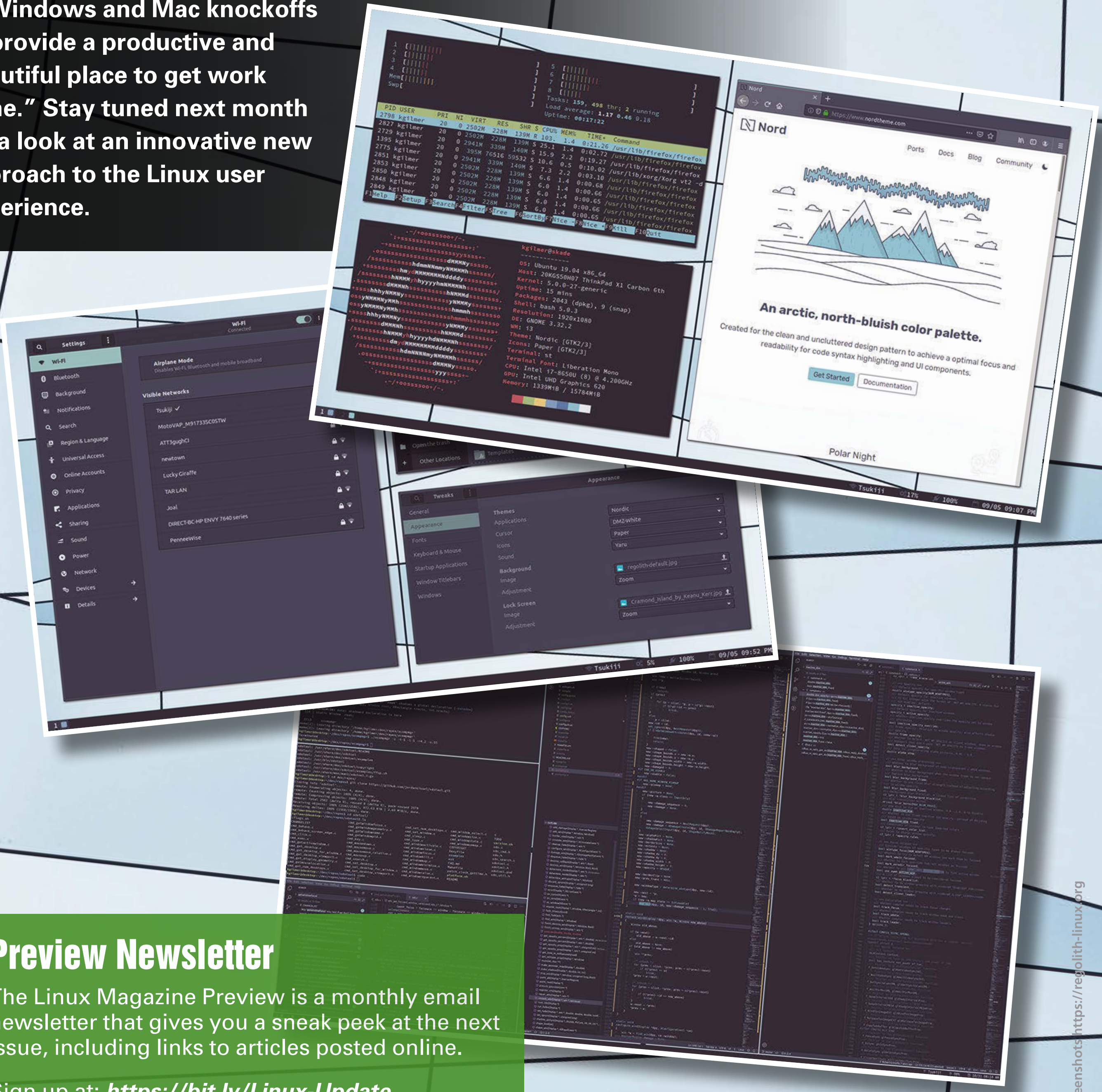
UK / Europe	Jan 04
USA / Canada	Jan 31
Australia	Mar 02

On Sale Date

Issue 231 – February 2020

Regolith Linux

Regolith Linux combines a tiling window manager with Gnome system management tools for a unique user experience that "ditches the cruft of Windows and Mac knockoffs to provide a productive and beautiful place to get work done." Stay tuned next month for a look at an innovative new approach to the Linux user experience.



```
  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+  Command
 2798 kgilmer    20   0 2502M  228M  139M R 103.  1.4  0:21.26 /usr/lib/firefox/firefox
 2827 kgilmer    20   0 2502M  228M  139M S 25.1  1.4  0:02.72 /usr/lib/firefox/firefox
 2729 kgilmer    20   0 2941M  339M  140M S 15.9  2.2  0:19.27 /usr/lib/firefox/firefox
 1395 kgilmer    20   0  395M 76516 59532 S 10.6  0.5  0:10.02 /usr/lib/xorg/Xorg vt2 -d
 2775 kgilmer    20   0 2941M  339M  140M S  7.3  2.2  0:03.10 /usr/lib/firefox/firefox
 2851 kgilmer    20   0 2502M  228M  139M S  6.6  1.4  0:00.68 /usr/lib/firefox/firefox
 2853 kgilmer    20   0 2502M  228M  139M S  6.0  1.4  0:00.66 /usr/lib/firefox/firefox
 2850 kgilmer    20   0 2502M  228M  139M S  6.0  1.4  0:00.65 /usr/lib/firefox/firefox
 2848 kgilmer    20   0 2502M  228M  139M S  6.0  1.4  0:00.66 /usr/lib/firefox/firefox
 2849 kgilmer    20   0 2502M  228M  139M S  6.0  1.4  0:00.65 /usr/lib/firefox/firefox
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nice +F9Kill  F10Quit
```

```
kgilmer@skade
-------------
OS: Ubuntu 19.04 x86_64
Host: 20KGS50H07 ThinkPad X1 Carbon 6th
Kernel: 5.0.0-27-generic
Uptime: 15 mins
Packages: 2043 (dpkg), 9 (snap)
Shell: bash 5.0.3
Resolution: 1920x1080
DE: GNOME 3.32.2
WM: i3
Theme: Nordic [GTK2/3]
Icons: Paper [GTK2/3]
Terminal: st
Terminal Font: Liberation Mono
CPU: Intel i7-8650U (8) @ 4.200GHz
GPU: Intel UHD Graphics 620
Memory: 1339MiB / 15784MiB
```

Nord

An arctic, north-bluish color palette.

Created for the clean and uncluttered design pattern to achieve a optimal focus and readability for code syntax highlighting and UI components.

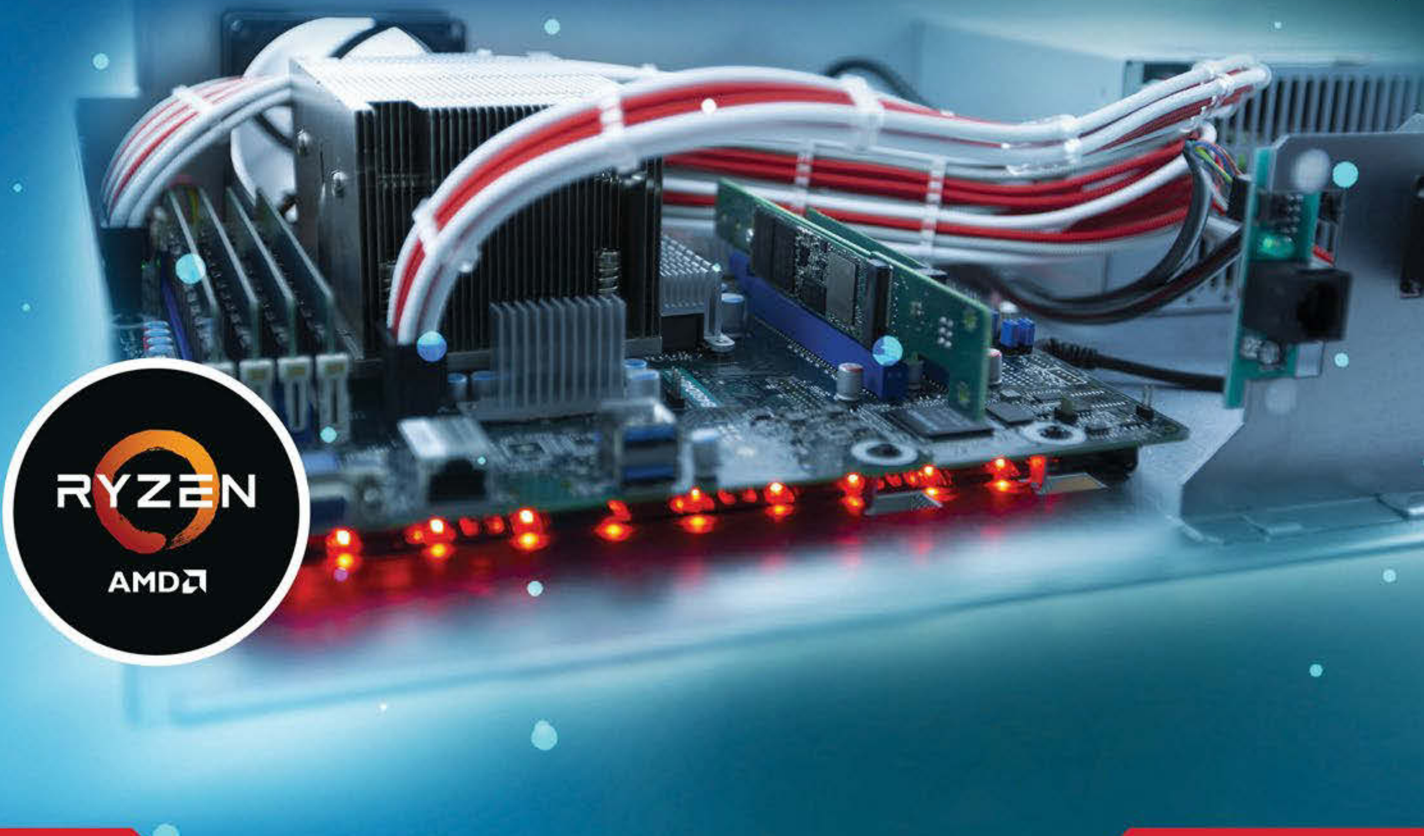
Preview Newsletter

The Linux Magazine Preview is a monthly email newsletter that gives you a sneak peek at the next issue, including links to articles posted online.

Sign up at: ***https://bit.ly/Linux-Update***

HETZNER

DEDICATED ROOT SERVER
DESIGNED FOR PROFESSIONALS



Dedicated Root Server AX41-NVMe

- ✓ AMD Ryzen 5 3600 Hexa-Core "Matisse" (Zen2) Simultaneous Multithreading
- ✓ 64 GB DDR4 RAM
- ✓ 2 x 512 GB NVMe SSD
- ✓ 100 GB Backup Space
- ✓ Traffic unlimited
- ✓ Location Finland and Germany
- ✓ No minimum contract
- ✓ Setup Fee $37.70

monthly from $37.70

Dedicated Root Server AX51-NVMe

- ✓ AMD Ryzen 7 3700X Octa-Core "Matisse" (Zen2) Simultaneous Multithreading
- ✓ 64 GB DDR4 ECC RAM
- ✓ 2 x 1 TB NVMe SSD
- ✓ 100 GB Backup Space
- ✓ Traffic unlimited
- ✓ Location Finland and Germany
- ✓ No minimum contract
- ✓ Setup Fee $59.90

monthly from $59.90

All prices exclude VAT and are subject to the terms and conditions of Hetzner Online GmbH. Prices are subject to change. All rights reserved by the respective manufacturers.

www.hetzner.com