

FREE DVD



GParted
64-bit

KALI LINUX
64-bit

ISSUE 232 MAR 2020

Double-Sided DVD
INSIDE!

DUPLICITY
Encrypt and back up files
with a single command

STOP ADS



LINUX PRO
MAGAZINE

MARCH 2020

STOP ADS

Before they reach
your browser



Firewalld and OpenSnitch
Stay safe from incoming and
outgoing threats

**Create a personal
web archive**

Curses
Gild your text-based IoT
apps with GUI effects

BBC micro:bit
Your intro to the
world of hardware
hacking

LINUXVOICE

- Organize your games with GameHub
- maddog: The promise of RISC-V
- Dead Simple VPN



FOSSPicks

- TreeSheets information organizer
- Polo file manager
- Xmonk virtual chanting

Tutorials

- PeerTube
- Readline

Issue 232
Mar 2020
US\$ 15.99
CAN\$ 18.50



7 25274 58049 1 03

Linux and Open-Source Enthusiasts

SCALE is Back!

Don't miss the next Southern California Linux Expo! It's North America's largest annual community organized Open Source gathering.

Register now as a *Linux Pro Magazine* subscriber to reserve your spot and save!

30% OFF Promo Code: LPRO

SOCALLINUXEXPO.ORG

Pasadena Convention Center

MAR 5-8 2020

ECONOMIES OF INK

Dear Reader,

The “paperless office” was once the dazzling vision of futurists and keynote speakers. Tech blogs and efficiency gurus have been talking about eliminating printers for at least 20 years, but it never really happened – or at least, not yet. Despite the predictions, contracts, school compositions, tax forms, and other essential documents sometimes require the corporeal manifestation that comes with a printer. In other cases, printing is just easier. I often print the articles I’m editing just because I like to make notes in the margins and draw lines and arrows to sketch out changes.

Printers are still around, but they are subject to the same forces that are affecting the rest of the industry. So even though they *look* the same, the business arrangements surrounding them are rapidly changing. So-called *cloud printing* has been with us for several years now. How do we make printers *even “smarter”*? And is “smart” always good?

I noticed a post on Slashdot recently from a guy whose printer quit printing, because he stopped paying the monthly fee for HP’s Instant Ink system. In case you’re wondering, yes, Instant Ink is a subscription service for printer ink. You pay a flat rate per month, and the ink is delivered automatically to your doorstep. You don’t even have to order it; your smart print cartridge knows when you’re about to run out and orders it for you.

The problem, apparently, is that some people don’t even know they have this service – they forget they signed up for the two month free trial and later notice an unexplained charge on their credit card. It appears that it is possible to exit the Instant Ink program in an orderly fashion, but you have to do it carefully and click all the right boxes. If you just stop paying, your smart print cartridge locks up and won’t print anything.

HP’s Instant Ink system has been around for a few years, so it isn’t exactly news, but they keep extending it to include more printers, so it is gradually gaining a higher profile. I talked to an HP guy once on an airport shuttle, and he told me that ink had always been the biggest source of the company’s profits. According to my source, HP used to *lose* money on the retail cost of a printer just to set up the chance to keep plying the owner with proprietary print cartridges. If you’re going to play that game, you really need to price the cartridges to cover the risk associated with estimating how much the user will actually print. Now, due to market forces, the company is less able to assume that risk, or perhaps, they want to provide the user with an incentive for assuming the risk of estimating print volume.

Instant Ink could be an attractive option – if you fit snugly into one of the available plans. Like a mobile phone company, the

Instant Ink service offers different prices for different levels of service. For instance, one plan lets you print 100 pages per month for \$4.99. That’s around 5 cents per page if you use all your pages, which isn’t too bad. But if you only print 50 pages, that’s more like 10 cents per page. (The plan does provide a means for rolling over unused pages, but it caps at 200 pages.) You owe the fee no matter how much you print, so if you only print one page, you pay \$4.99 per page for that month. If you go over the maximum page count for your plan, the per-page rate scales up, which can lead to costly overruns.

Interestingly, the company even offers a “Free” printing plan, which allows you to print 15 pages per month for no cost, and then you owe HP 10 cents per page for everything else you print, which is kind of like the old days, when we used to print faxes, documents, and photocopies at the local copy store for 10 cents a page, only this time, you are paying 10 cents per page to print them *on your own printer*. I know I’m old school, yet still I must admit some misgivings about passing from an era where you never really fully own your own software to this brave new tomorrow where you don’t even fully own your own hardware. However, from a financial viewpoint, it does appear that, if you have a supported HP printer and are pretty confident about a steady and predictable output, you probably could save money with Instant Ink. Just be sure you follow the instructions if you formally withdraw from the program so your printer doesn’t go on strike.

And shop around for other innovations. For instance, the Epson EcoTank series does away with cumbersome and costly print cartridges altogether, providing a refillable tank for a significant per-page print savings that Epson says can be as much as 90 percent.

In theory, as long as competition exists, some of the benefits of innovation should get passed back to the consumer, but read the fine print, and be aware that lower cost with greater risk is sometimes no savings at all.



Joe Casad,
Editor in Chief





WHAT'S INSIDE

Web ads and ad scripts can slow down your system and gum up your web experience. Browser-based ad-blockers are useful for controlling many types of popups and banners, but they are less effective with ads built into applications – and they are often difficult to implement on mobile devices. This month we look at a couple of alternative tools for blocking ads at the network level: Pi-hole and Privoxy. Also in this issue:

- **Firewalld and OpenSnitch** – a practical workshop on configuring firewalls for outgoing, as well as incoming, traffic (page 30).
- **Create a Personal Web Archive** – Future-proof your bookmark library by downloading and archiving web pages (page 46).

Check out MakerSpace for a study of the tiny BBC micro:bit, and turn to this month's LinuxVoice for a look at how to build a unified game library with GameHub.

SERVICE

- 3 Comment
- 6 DVD
- 96 Featured Events
- 97 Call for Papers
- 98 Preview

NEWS

08 News

- Kubuntu Focus Laptop Is Now Ready for Preorder
- Dell Adds a Much-Requested Feature to the New XPS Developer Edition Laptop
- Bonsai Promises to Make Syncing Gnome Devices Easier
- Huawei Releases CentOS-Based openEuler as Open Source

12 Kernel News

- Line Ending Issues
- Hardware Hinting
- Simplifying the Command Line

16 Interview – Wikimedia's Jaime Crespo

Jaime Crespo, from the Wikimedia Foundation, discusses the challenges of managing one of the world's largest collaborative knowledge projects.



COVER STORIES

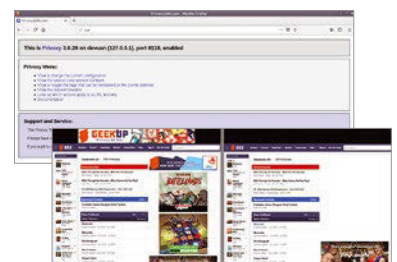
20 Pi-hole

The Pi-hole ad blocker filters ads and trackers from the data stream for all devices on the network, from your smartphone to your toaster.



24 Privoxy

Add-on ad blockers can help mitigate the degradation of your browsing experience, but sometimes you need to bring stronger weapons. A filtering web proxy can scrub web traffic to eliminate unwanted ads and scripts.



IN-DEPTH

30 Firewalld and OpenSnitch

For maximum security, you'd better watch traffic in both directions. This hands-on workshop takes you through the steps of setting up firewalls for outgoing as well as incoming traffic.



34 Programming Snapshot – Go Game States

How does a ferryman transport a wolf, a goat, and a cabbage across a river in a boat that can only carry the ferryman plus one item at a time, without the wolf eating the goat or the goat eating the cabbage? Mike programs the solution in Go.

40 Charly's Column – Ishw

In order to avoid complaints from his children, Charly prefers to use Ishw instead of a screwdriver to analyze his home firewall PC's hardware details.

42 Command Line – duplicity

With a single command, duplicity lets you encrypt and back up files. All you need to do is learn its unconventional command structure.



46 Create a Personal Web Archive

A large collection of bookmarked pages is worth protecting. With the right script, you can create an archive so you never lose access to all your favorite web pages.

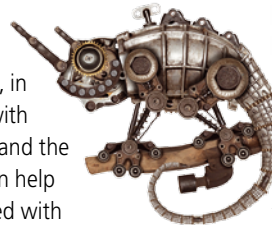
MAKERSPACE

52 curses

When you need some quick graphical output, the old school curses library can save you some time and effort.

56 BBC micro:bit

Designed for students, the BBC micro:bit, in conjunction with MicroPython and the Mu editor, can help you get started with microcontroller programming.



62 Open Hardware – DIY Soldering Kits

If you don't know how to solder, your DIY options are limited. Audio Builders Workshop gives you hands-on soldering experience.

LINUXVOICE

65 Welcome

This month in Linux Voice.

78 FOSSPicks

Graham looks at TreeSheets, rare, McFly, b2, DrumGizmo, A/B Street, Xmonk, and much more!

67 Doghouse – RISC-V Summit

While attending the second summit on RISC-V architecture, maddog was blown away by the level of openness and collaboration.

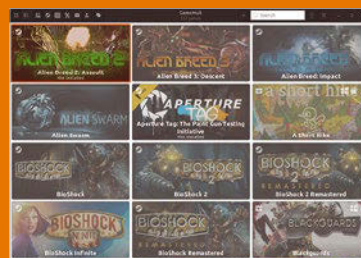
68 GameHub

Organize your games by bringing them all together into a single library.



74 Dead Simple VPN

With a single command, Dead Simple VPN builds a secure VPN connection.



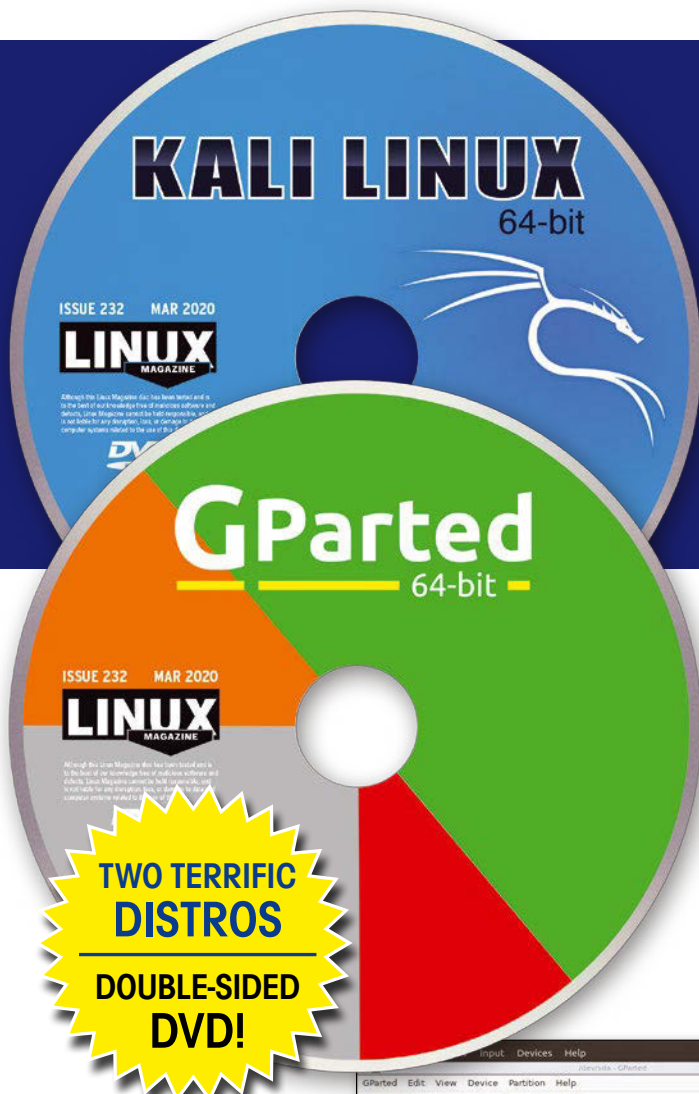
86 Tutorial – PeerTube

Self-host your videos without the limitations embedded in YouTube and similar platforms.

92 Tutorial – Readline

Readline provides a rich set of tools for moving around quickly on the command line.

On the DVD

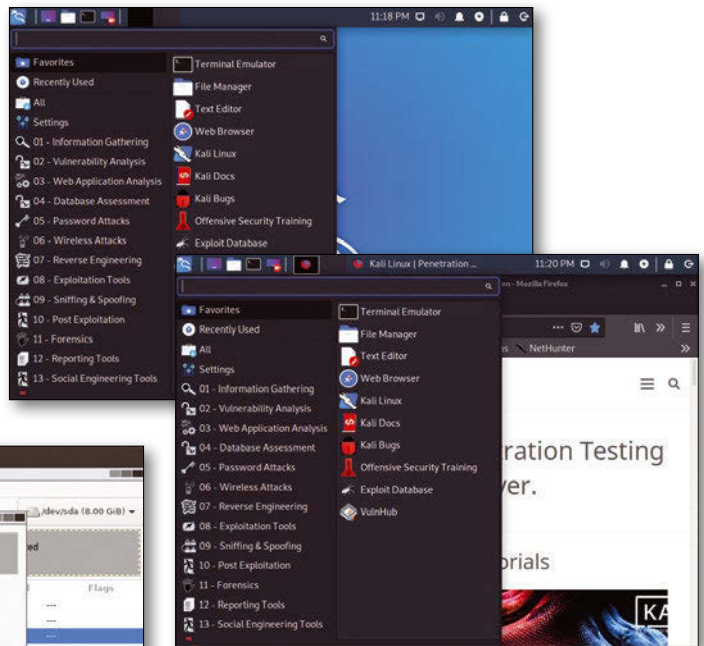


Kali Linux (64-bit)

Kali Linux is a popular distro dedicated to the craft of penetration testing. Kali comes with hundreds of practical tools for information gathering, vulnerability analysis, wireless attacks, and stress testing. A bootable Forensics mode leaves the drives unmounted and provides a powerful collection of forensics utilities.

GParted (64-bit)

The GParted Live DVD is the easiest and best way to access the GParted partition editor. Use GParted to create space for new operating systems, grow or shrink existing partitions, and rescue data from lost partitions.



Additional Resources

- [1] Kali Linux: <https://www.kali.org/>
- [2] Kali documentation: <https://www.kali.org/docs/>
- [3] Kali Linux Tools: <https://tools.kali.org/tools-listing>
- [4] GParted: <https://gparted.org/>
- [5] GParted documentation: <https://gparted.org/documentation.php>

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.

openSUSE + LibreOffice Conference



LibreOffice
The Document Foundation

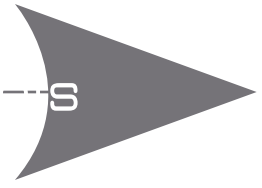


A Conference for
Open Source Developers
October 13 - 16
Nuremberg, Germany

events.opensuse.org

NEWS

Updates on tech



THIS MONTH'S NEWS

- 08 • **Kubuntu Focus Laptop Now Ready for Preorder**
- 09 • **Dell Adds a Much Requested Feature to the New XPS Developer Edition Laptop**
 - **Bonsai Promises to Make Syncing Gnome Devices Easier**
 - **More Online**
- 10 • **Huawei Releases CentOS-Based openEuler as Open Source**

Kubuntu Focus Laptop Now Ready for Preorder

The Kubuntu Focus is a new Linux laptop effort set to marry the Kubuntu Linux distro (<https://kubuntu.org/>) and a laptop aimed specifically for gamers, power users, developers, video editors, and anyone who seeks performance and seamless Linux compatibility. This brand new laptop is ready for preorder (<https://kubuntufocus.myshopify.com/>).

The laptop was born from a collaboration between Kubuntu, TUXEDO Computers, and MindShareManagement Inc. The Kubuntu Focus will not only highlight the KDE desktop environment, but it will be the first officially recognized laptop created specifically for the Kubuntu Linux distribution.

But before you visit the site for preorder, understand this is a premium piece of hardware with a premium price tag. The hardware specs alone should clue you in on the price. The base model includes:

- **Display** – Full HD 16.1-inch matte 1080p IPS 144Hz
- **CPU** – 6-core/12-thread Intel Core i7-9750H processor with 4.5GHz Turbo
- **GPU** – NVidia GeForce RTX 2060 6GB GDDR6 with PhysX (<https://www.softpedia.com/get/Multimedia/Graphic/Graphic-Others/NVIDIA-PhysX.shtml>) and CUDA graphics card
- **RAM** – 32GB dual channel DDR4 2666 RAM
- **Storage** – 1TB Samsung 970 EVO Plus NVMe SSD

You can bump the RAM up to 64GB and the GPU to an NVidia RTX 2080.

Other noteworthy features include:

- **Backlit keyboard**
- **Kensington lock**
- **NVMe and SSD near-silent operation**
- **Temperature controlled fans**
- **Metal surface chassis and plastic bottom**
- **Dual-mode Bluetooth 5**
- **Optical S/PDIF output**
- **2-in-1 audio**
- **6-in-1 card reader**
- **Full disk encryption**



MORE ONLINE

The base unit sells for \$2,395. A maxed out version will set you back \$3,555. The units are set to start shipping in February 2020. For more details (as well as benchmarks), check out the Kubuntu Focus online (<https://kfocus.org/>).

Dell Adds a Much-Requested Feature to the New XPS Developer Edition Laptop

To anyone who has spent any time researching companies that offer hardware with Linux preinstalled, chances are you know about the Dell XPS Developer edition. This began as Project Sputnik in 2011, when Dell's Barton George realized that no major original equipment manufacturer (OEM) was building a fully-supported Linux laptop that included drivers and provided a great out-of-the-box experience.

Fast forward nine years later, and the project is still going strong. In fact, the Dell XPS Developer Edition has been declared a best in show Linux laptop by numerous reviewers and outlets. Dell knows this and understands the audience for which this hardware is targeted. Dell also listens to the communities they serve.

Case in point, the Linux community.

One thing that has been sorely missing from Linux laptops is support for the fingerprint reader. This form of biometric security is not only superior to passwords, it's more efficient. And Dell is finally bringing a fingerprint reader to the 10th generation XPS 13" Developer Edition.

Although details on the fingerprint reader are sparse, Dell has announced that support for the fingerprint reader will be available as an over-the-air (OTA) update soon after the hardware is released in February 2020.

For those that are curious, the specs for the machine look like:

- 10th generation Intel Core 10nm mobile processors
- Ubuntu 18.04 LTS
- Fingerprint reader support (driver initially available via OTA update)
- Up to 32GB memory
- Up to 3x faster wireless with Killer AX1650 built on Intel WiFi 6 chipset, supports up to 2TB PCIe SSD
- Up to 4K Ultra HD+ (3840x2400) display

Cost for new XPS Developer Edition will start at \$1199.

For more information, see Barton George's website (<https://bartongeorge.io/2020/01/01/introducing-the-2020-xps-13-developer-edition-this-one-goes-to-32/#comments>).

Bonsai Promises to Make Syncing Gnome Devices Easier

If you're a GNOME user with multiple devices and have longed for the day when those devices could easily be synced with one another, that wish may be coming true. Red Hat developer Christian Hergert has started developing a project called Bonsai, which will serve as a sort of personal cloud for all of your Gnome-based devices.

On his blog, Hergert stated, "I want access to my files and application data on all my computing devices but I don't want to store that data on other peoples computers." This idea led Hergert to create Bonsai. Although this tool is very much in the experimental phase



Photo by Sarah Brink on Unsplash

Linux Magazine

www.linux-magazine.com

Paw Prints • Jon "maddog" Hall

TCO of FOSS vs. Closed Source: Mr. Bacil, you are wrong

Today I received a link to a news article from Brazil where first-term Paraná state deputy Emerson Bacil of Jair Bolsonaro's PSL party is proposing to change a law that prefers Free and Open Source Software (FOSS) used by government over closed source, proprietary software.

GPU Computing

<http://www.admin-magazine.com/HPC/>

Exploring AMD's Ambitious ROCm Initiative

• Joe Casad

Three years ago, AMD released the innovative ROCm hardware-accelerated, parallel-computing environment.

Porting CUDA to HIP

• Joe Casad

You've invested money and time in writing GPU-optimized software with CUDA, and you're wondering if your efforts will have a life beyond the narrow, proprietary hardware environment supported by the CUDA language. Welcome to the world of HIP.

ADMIN Online

<http://www.admin-magazine.com/>

Transcoding Optical Media in Linux

• Erik Bärwaldt

We test two candidates for viewing and converting video DVDs and Blu-ray discs, HandBrake and MakeMKV, and point out legal hurdles.

Storage Monitoring with Grafana

• Andreas Stolzenberger

Create intuitive and meaningful visualizations of storage performance values with a "TIG" stack: Telegraf, InfluxDB, and Grafana.

Call Web Pages in the Terminal with Browsh

• Tim Schürmann

The Browsh command-line browser displays web pages with text characters and thus supports true-to-layout browsing at the command line.

(which means it's not an official Gnome project), it is being hosted on Hergert's Gnome GitLab repository.

The gist of Bonsai is simple – a daemon and shared library for providing personal cloud-like services specifically to the Gnome desktop. The goal of the project is to include services such as:

- File storage
- Mail
- Calendar
- To-do list/checklist/notes
- Photo albums
- Music/podcasts/audio books/radio
- Videos
- Search
- Backup
- System migration
- VPN

At the moment, there is no telling if Bonsai will become an official Gnome application/service, but considering this fills a much-needed hole in the Linux desktop landscape, it's a safe bet that it (or something similar) will become a reality in the near future.

For more information, see <https://blogs.gnome.org/cherbert/2020/01/01/introducing-bonsai/>.

Huawei Releases CentOS-Based openEuler as Open Source

With the US ban on Huawei, the company has opted to focus some of their efforts on their own server solution. To prevent a continued dependence on US software, the Chinese company has released an open source version of EulerOS, a server platform based on CentOS.

Geared for IoT and cloud infrastructure, openEuler (<https://openeuler.org/en/>) has been released for ARM64 architecture and is compatible with the Open Containers Initiative (OCI). Huawei claims it has made changes to CentOS to give it a performance boost and reliability.

OpenEuler currently has more than 50 contributors and 600 commits. The openEuler repositories include two new projects: iSulad, a lightweight container runtime daemon created specifically for IoT and the cloud (<https://gitee.com/openeuler/iSulad>), and A-Tune, an OS-tuning software (<https://gitee.com/openeuler/A-Tune>).

At the moment, there is very little information about openEuler, and the only documentation to be found has yet to be translated to English. However openEuler's source code has been made available and can be found on Gitee (<https://gitee.com/openeuler>).

The official mission of openEuler states "through community contributions, openEuler builds an innovative platform and a unified and open OS that supports the multi-processor architecture, and promotes the robustness of the software and hardware application ecosystem."

Anyone wishing to contribute to the project can read up on the process in the Contributions to the Community documentation (<https://openeuler.org/en/developer.html>).

Huawei also claims that openEuler is one of the most secure operating systems available by offering:

- Configurable hardening policies
- Kernel-level OS security capabilities
- China's Ministry of Public Security operating system information security technology certification
- CC EAL4+ certification with the German BSI Protection Profile (PP) standard
- CC EAL2+ certification with the US NIAP PP standard
- US NIST CAVP cryptographic algorithm certification
- Support for Nessus security leak detection tool
- Support for NSFOCUS RSAS security leak detection tool



MEET OVER 300
LEADING SUPPLIERS AND HEAR
FROM OVER 250 INDUSTRY SPECIALISTS.
BOOK YOUR FREE TICKET TODAY:
WWW.CLOUDEXPOEUROPE.COM/DEVOPS-LINUX

Collaborate to innovate.

Join the pack at DevOps Live 2020 and start running with the collaborative method that delivers vastly streamlined ways of working. That will make your security more agile. That will improve communication to help you identify bugs or code vulnerability. Tuck into all the industry hot topics and more, including DevSecOps, AIOps, Machine Learning and DataOps.

Register for your FREE ticket today:
www.cloudexpoeurope.com/devops-live/free



DEVOPS LIVE

11 - 12 March 2020 ExCel London
www.cloudexpoeurope.com/devops-live

ORGANISED BY  CloserStill

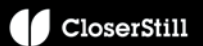
PART OF
**LONDON
TECH SHOW**

INCORPORATING



EVERY EMERGING TECHNOLOGY. ONE DIGITAL TRANSFORMATIONAL JOURNEY.

ORGANISED BY



HEADLINE SPONSOR



MANAGED CLOUD SPONSOR:



THEATRE SPONSORS



STREAM SPONSORS



GOLD SPONSORS



SILVER SPONSORS



Zack's Kernel News



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

By Zack Brown

Line Ending Issues

Jonathan Corbet posted some kernel documentation updates gathered from many contributors. However, Linus Torvalds spotted some file format problems. Specifically, some of the patches used CRLF (carriage return, line feed) characters as line endings. This dates back to MS-DOS of the elder days, and it's hard to imagine these line endings appearing in any Linux development tool chain.

Jonathan C. said it wasn't his tool chain, but that Jonathan Neuschäfer had submitted the patches to him with those line endings. More importantly, he said, "The problem repeats if I apply those patches now, even if I add an explicit '--no-keep-cr' to the 'git am' command line. It seems like maybe my version of git is somehow broken?"

Linus replied, also ccing the git development mailing list. He speculated, "I wonder if the CRLF removal is broken in general, or if the emails are somehow unusual (patches in attachments or MIME-encoded or something)? Maybe the CRLF was removed from the envelope email lines, but if the patch is then decoded from an attachment or something it's not removed again from there?"

Meanwhile, Jonathan N. was equally confused, saying he was "not sure why, or where in the mails' path this happened, but the base64 with CR/LF inside is also present in the copies that went directly to me, rather than via the mailing lists."

Jonathan C. checked his email history and reported that although the email containing the patch was plain text, the patch itself was a base-64-encoded attachment. So git's '--no-keep-cr' would not have filtered that encoded text.

Linus agreed this was probably the case – but wondered why the patches would have successfully applied to

Jonathan C.'s tree in the first place. Linus said, "I'm surprised, though – when git applies patches, it really wants the surrounding lines to match exactly. The extra CR at the end of the lines should have made that test fail."

Jonathan C. confirmed that he used the git flag '--ignore-whitespace' in his process. When he applied these patches without that flag, git indeed choked on it. Jonathan explained, "Docs patches often come from relatively new folks, and I've found that I needed that to apply a lot of their patches. But clearly that was not a good choice; among other things, I've lost the opportunity to tell people when their patches have the types of whitespace issues that this option covers over. I've taken it out of the script and will only use it by hand in cases where I'm sure that it won't cause problems."

Junio C. Hamano, the git maintainer, affirmed that the behavior produced by Jonathan C. was normal for git. But Linus did not agree. Linus said:

"I think it's a mistake that --no-keep-cr (which is the default) only acts on the outer envelope.

*"Now, *originally* the outer envelope was all that existed so it makes sense in a historical context of "CR removal happens when splitting emails in an mbox". And that's the behavior we have.*

"But then git learnt to do MIME decoding and extracting things from base64 etc, and the CR removal wasn't updated to that change."

Junio was skeptical about making this change. He replied:

"What was the reason why '--no-keep-cr' was invented and made default? Wasn't it because RFC says that each line of plain-text transfer of an e-mail is terminated with CRLF? It would mean that, whether the payload originally had CRLF terminated or LF terminated, we would not be

able to tell – the CR may have been there from the beginning, or it could have been added in transit. And because we (the projects Git was originally designed to serve well) wanted our patches with LF terminated lines most of the time, it made sense to strip CR from CRLF (i.e. assuming that it would be rare that the sender wants to transmit CRLF terminated lines).

“If the contents were base64 protected from getting munged during transit, we know CRLF in the payload after we decode MIME is what the sender meant to give us, no?”

The question was not decided in the discussion, which petered out around here.

An interesting tidbit is that although Linus is not the git maintainer, he has enormous, probably decisive, influence over its course of development. Linus originally wrote git himself over the course of a few weeks in 2005, after Larry McVoy canceled the BitKeeper license, and no other open source revision control system proved capable of the speed and efficiency Linus needed for kernel development.

After totally and forever transforming revision control across the known universe, Linus handed off leadership of the project to its most active contributor, Junio, who has maintained it ever since. Junio generally has the last word in git development decisions – but there’s an unspoken reality that if Linus wants git to move in a certain direction, that’s the direction Junio moves it. This makes some amount of sense, since the entire reason for git’s creation was to support Linux kernel development; it’s difficult to imagine a legitimate change to git that would make git better but Linux development worse.

In this particular case, the question of how to handle line endings and file encodings will probably take the natural course of reasoned discussion, rather than anyone simply pronouncing an arbitrary decision.

Hardware Hinting

Given hardware vulnerabilities like Meltdown and Spectre, Linux has had to take strong measures to work around those problems. But recently Vitaly Kuznetsov asked, what about container-based virtual systems that only appear

to be running on affected hardware? Depending on the true CPU topology of a given system, one or another security strategy would be best – but as Vitaly pointed out, the user would need to know the true CPU topology in order to make that choice.

It’s not just a question of protecting against security vulnerabilities. The hard core STIBP patch solves the Spectre and Meltdown problems entirely and simply, but at a high cost of speed. The various available alternatives are all ways to accept a more complicated solution while regaining some of the speed lost by STIBP.

Of course, more complicated solutions have other trade-offs, and there are several floating around. One option, Vitaly said, would be to simply opt in to the STIBP patch and be done with it. That’s a legitimate option, especially if the user is not certain what hardware their system is truly running on.

One attempt to speed things up is Symmetric Multithreading (SMT). It allows one CPU to appear to be two and lets them engage in normal kernel threading and load balancing. This does result in better overall CPU utilization, but traditionally it’s shown to have some potential security risks. Then there is Peter Zijlstra’s core scheduling patch, which tries to mitigate SMT’s security risks by grouping certain of these virtual CPUs together, to avoid potentially risky interactions with other virtual CPUs on the same system.

All of these possibilities, Vitaly said, would be legitimate options for a user to consider, if only they knew what hardware the system actually used. As he put it, the question boiled down to, “does the topology the guest see[s] match hardware or if it is ‘fake’ and two vCPUs which look like different cores from guest’s perspective can actually be scheduled on the same physical core. Disabling SMT or doing core scheduling only makes sense when the topology is trustworthy.”

Liran Alon remarked that this was not only a security issue – the same considerations were needed for other speed optimizations as well. For example, it would need to ask the same questions when deciding whether to run tasks that share memory on the same Non-Uniform Memory Access (NUMA) node.

There were a few criticisms of Vitaly's patch. Peter felt that "The only way virt topology can make any sense what so ever is if the vcpus are pinned to physical CPUs. And I was under the impression we already had a bit for that [...]. So I would much rather you have a bit that indicates the 1:1 vcpu/cpu mapping and if that is set accept the topology information and otherwise completely ignore it."

The conversation meandered a bit. Since it's about security, objections can come from anywhere, and the final concept may look like anything. In this case, it's clear that giving hints about the underlying architecture to higher level code would be useful for both speed and security. But it looks like Vitaly's ideas about how to do this may need significant revision before they get into the kernel.

One thing I find fascinating about this type of discussion is that when it comes to new security features, someone generally needs to take the plunge and actually implement something, with the full knowledge that once they do, there will be a set of nearly random objection-vectors coming at them. And only then will it start to become clear what the proper solution will end up looking like. So in a way, developers need to create something they know will be destroyed, just in order to then be able to create it again.

Simplifying the Command Line

Masami Hiramatsu posted a patch to support Extra Boot Config (XBC) to allow users to pass a configuration file to the kernel at boot-time. Configuration would have a tree structure, and users could access it via Linux's Supplemental Kernel Cmdline (SKC) API. The great benefit of this patch would be simplifying the kernel command line. Pretty much everyone agrees the command line has gotten out of hand. Other attempts to simplify it – including support for arbitrary binary blobs of data right there in the command line – have come and gone.

Randy Dunlap had no immediate objection, though he noticed that Masami had set XBC support to be enabled by default in all kernel builds. Normally, Randy said, such a decision would

need a lot of justification. Marami said he'd change it to be disabled by default, but he also remarked, "I thought that was OK because most of the memories for the bootconfig support were released after initialization. If user doesn't pass the bootconfig, only the code for /proc/bootconfig remains on runtime memory."

Steven Rostedt also affirmed Randy's idea that new features are normally disabled by default. But Steven said in this case, Masami's patch should be enabled by default. Steven stated his case:

*"This is not some new fancy feature, or device that Linus complains about 'my X is important!'. I will say this X *is* important! This will (I hope) become standard in all kernel configs. One could even argue that there shouldn't even be a config for this at all (forced 'y'). This would hurt more not to have than to have. I would hate to try to load special options only to find out that the kernel was compiled with default configs and this wasn't enabled.*

"This is extended boot config support that can be useful for most developers. The only ones that should say 'n' are those that are working to get a 'tiny' kernel at boot up. As Masami said, the memory is freed after init, thus this should not be an issue for 99.9% of kernel users."

And Masami agreed with Steven, saying, "Yes, for the users point of view, it is hard to notice that their kernel can accept the boot config or not before boot. To provide consistent system usability, I think it is better to be enabled by default. Anyway, if there is no boot config, almost all buffers and code are released after init (except for /proc/bootconfig entry point, which will return an empty buffer)."

But Masami did acknowledge that the actual compiled kernel binary would be about 15 or 20KB larger with this patch than without it.

There was no further discussion, but it seems clear that this is a very welcome patch, at least for some developers. Whether it will make it all the way through the gauntlet is another question. Certainly everyone including Linus would like to see a better way of dealing with the kernel command line. Maybe Masami's patch will be it. ■■■

SUSECON™

'20

Be the Difference

March 23-27, 2020 | Dublin, Ireland

#SUSECON #BeTheDifference



Elevate your business through increased agility and faster innovation.

Learn how at SUSECON with:

- Access to SUSE leadership and technical experts
- 100+ hours of hands-on training
- Technology showcase
- Certification exams
- 150+ sessions
- Tutorials

Register at susecon.com!



The Wikimedia Foundation

Streamlined
Management

Jaime Crespo, from the Wikimedia Foundation, discusses the challenges of managing one of the world’s largest collaborative knowledge projects. *By Mayank Sharma*

At Percona Live Europe 2019, I joked with Jaime Crespo, Senior Database Administrator (DBA) for the Wikimedia Foundation, that the Wikipedia project is essentially just one big database. You might expect that it takes hordes of DBAs to keep one of the world’s largest knowledge pools constantly online, but you’d be wrong. Crespo shares the unique challenges Wikimedia faces in managing its servers.

Linux Magazine: Tell me about yourself and how you got started as a database administrator.

Jaime Crespo: My name is Jaime Crespo, and, by my name, it’s easy to spot that I’m from Spain. I’m currently the Senior Database Administrator for the Wikimedia Foundation. A bit of follow up to that I have to do is that the Wikimedia Foundation is the nonprofit organization that provides support, mostly technology, legal and management, to the Wikipedia project and to a lot of other open knowledge projects that we maintain. The actual work is done by everyone, that is, our volunteers. I’m part of the SRE team (Site Reliability Engineering) that helps maintain the servers, so that volunteers and contributors can edit.

And concerning the second part, I was always interested in computing. Like many people, I started with more of a developer profile, but I quickly got more interested into systems, into the back end side of things. Almost by chance, I got involved into MySQL, because the only MySQL AB partner in Spain wanted someone else to deliver their courses.

LM: Was training your first interest?

JC: Yeah, I knew MySQL, and I had used it before, but that was when I got like very deep into it. Fast forward to like four or five, six years ago, that’s when the Foundation had an open position for a database administrator. I was already involved with Wikipedia, as a volunteer. At the beginning, I was just to help the other DBA, but then the other DBA left, and that’s how I got here. I was the only one for some time, and then we finally brought Manuel [Arostegui] onboard, who is the other current DBA.

LM: What is the scale of data at Wikipedia and what’s the risk?

JC: Oh, as you joked earlier, Wikipedia is “almost just a database.” That’s obviously an oversimplification, but for the presentation here at Percona Live Europe 2019, we did a bit of research on “how large is Wikipedia?” Because we work with it every day, we know every single database server and every single database instance, but we never did the math of how large it is. In terms of relational data (so think MySQL/MariaDB), we have over half a petabyte of data in total. That’s only the relational data, so think of that as wiki content, plus a lot of miscellaneous internal services for development and support, you know, those kinds of things that people normally don’t see. But that’s just the text kind of content. And then there is the other side that is media files. And I know that’s much larger. Images, videos, and other non-relational data are stored in OpenStack Swift.



LM: Is Wikipedia a read-intensive deployment?

JC: Yes. So that’s one of the benefits and curses of our kind of installation; the fact that we can cache very effectively. So anecdotally, I don’t have firsthand data on this, but I think around 90 percent of the requests that we get are served directly from the edge cache. We maintain our own CDN [content delivery network] and 90 percent [of] the requests don’t even have to reach the application or the database. Those other 10 percent, they normally arrive at the application, and we also have several layers of rather complex caching, and some of them end up on the application and end up querying the databases. In terms of that database traffic, again, I’m speaking loosely here, I think we’ve got around 400,000 SQL queries per second. We serve that with around 150 to 200 MySQL instances. They’re not one to one to physical machines, because we do some consolidation.

So what happens if the database doesn’t work? We are relatively safe with reads, because they end up in caching; 90 percent of the reads usually will continue to be served. The problem is editing. The reads and writes that cannot be cached will obviously have to go to the database. MySQL is essential for the editors basically. That’s why we need high availability; we need redundancy on the database system.

LM: Has something like that happened during your time at Wikipedia?

JC: We regularly schedule downtime in terms of “we have to go to read-only.” When that happens, the systems are still available for reads. People don’t actually realize many of the times we do maintenance, because it’s a very compartmentalized system, and even if there is an

unscheduled outage, it only affects a small part of the database. But it will be very rare to have like a full blown outage, because first we have redundancy within each wiki; but whenever there’s a larger issue, usually it would only affect a small subset of the wikis. So that is my job: to make it more reliable. We’re constantly working to make things more stable, more fast, more performant.

LM: In addition to managing the data, which features are essential to your deployment?

JC: So I’m talking only about the database. There are various things that we put in place. The first one is redundancy. Our topology is that we have, as I said, those different sections (replica sets). We have around 24 replica sets, which is basically groups of servers, independent servers which are physically isolated. So they’re on different parts of the data center with its own redundant power and redundant network, and each of them is a replica of each other, and they are kept in sync. So if one fails, usually there’s nothing to do manually. As usual, we need to do work on making things more automated. With such a size, the less the intervention the better.

One thing we are working on right now is redundancy at the data center level. So if the worst case scenario happens – a meteorite hits the US East Coast – I think we can be on the other data center in at most 15 minutes. We already have full duplication of all the resources on two different geographical locations. So we have, I think, a relatively serious approach to availability.

The problem right now is that the setup is active/passive for databases. So one of the data centers is less used. It’s a complicated issue for the database which has most of the application state, but we are working right now on making the setup active/active; the primary data center for both reads and writes and for reads only on the second one. And that, in addition to availability, will improve also performance, for editors.

LM: What tools do you develop in house?

JC: So we DBAs are not into [the] heavy kind of development except in terms of

“system development,” so think automation, configuration management, and such. We at Wikimedia develop a lot of tools in terms of the application layer. One of the most famous is MediaWiki, and that by itself has a lot of features, such as automatic load balancing, and so in many ways, those features are done kind of by the application itself.

We DBAs are heavy reusers of existing open source solutions, for things like middlewares or proxies, rather than trying to develop our own. We’re always keeping an eye out for something interesting. And in terms of the tools that we use, we try to use open source tools, because again, with the small resources that we have, we take all open source solutions, and we integrate with them. In my talk [at Percona Live Europe 2019], I talk about two of them: mydumper, which was initially developed by Domas Mituzas (former Wikimedia volunteer, who is now working for Facebook) and now mainly maintained by Percona, and XtraBackup, which is Percona although we use MariaDB, and the MariaDB version of XtraBackup (Mariabackup) is maintained by the MariaDB project. Those projects, as well as the database themselves, usually only get light patches from us DBAs.

We DBAs do some development, but in most cases they’re not big, reusable tools. What we do normally in SRE is small utilities that glue existing ones to deploy them in our environment. What we give in exchange for that is, first, everything that we do is obviously always open source. So if you want to replicate our way of doing things, you can directly take it. In fact, I was talking to someone before, and they were shocked that in order for something to be in production in Wikipedia, it has to be in a public repository first. So everything is transparent.

There are, however, some bits of proper in-house developed standalone software. One of them is called Cumin [1], which is kind of an orchestration tool. We use it, for example, to schedule backups and to do remote copies. Another is called PyBal [2] that’s our load balancer manager. It uses LVS (Linux Virtual Server) to load balance the traffic, but not for the database, only for the HTTPS traffic. Because as I said earlier, the database load balancing is part of the application layer. We

also employ the main developer of our DNS server, `gdnssd` [3].

LM: Has Wikipedia always used MariaDB? What made you upgrade and how did you decide to do it?

JC: A bit of a disclaimer: I wasn’t there when the decision was taken. Look at the blog post for more accurate reporting [4]. But let me give you my spin on it as I saw it. I understand why it was done, and I would probably have done the same thing. At that time, MySQL was going through a bit of a rocky phase with several purchases of several companies. So first of all, there was some fear in terms of the future of MySQL that on the kind of open source policy side. And the other thing was in the pure technical side [that] vendors like Percona and MariaDB were doing a lot of cool developments, a lot of really nice things. I remember, one of the technical reasons why MariaDB was needed was because it supported useful features much earlier than its peers, features like multisource replication – the idea of having several servers replicating to a single one. So the combination of, “Oh, now there is a MariaDB Foundation, which has pledged to support open source and we are a foundation too.” So we went through several iterations including regular MySQL, and I think at some point we used Facebook’s version of MySQL.

LM: Wikimedia used Facebook’s version of MySQL?

JC: Yes, and I know that there was either some testing or some at production. But at that time, the number of servers was really small. I think maybe three servers or three servers and three backups, so relatively small. I think the previous DBA had worked with MariaDB, so he knew that better than the other products.

That’s when I came in, and I finished the migration to MariaDB. At that time, we were migrating either to Maria DB 5.5 or MariaDB 10.0. We are now so large that migrations kind of overlap. By the time you finish one, there’s a new version to migrate to. We just finished the move from MariaDB 10.0 to MariaDB 10.1.

Right now, we have to decide what’s the next migration path. Just speaking from [the] technical side, we are seeing

MySQL doing a lot of cool things. For example, let me tell you about the specific features that we are looking at. The new data dictionary in MySQL is very interesting for us, because we have a lot of objects on certain databases, and there’s a lot of improvements in that regard. So there are features that we are looking at on the newer versions, both MariaDB and MySQL, that we want to have.

LM: So let me see if I understand this correctly: You don’t want to move from the current version of MariaDB to a later version, because, if you do that, it will be very difficult for you to migrate to another database, since MySQL and MariaDB are no longer drop-in replacements.

JC: One of the things in terms of the vision of the Wikimedia Foundation is that we don’t like vendor lock-ins. Because we’ve had bad experiences with open source vendors, that they either decide to go for a non-free license or an open-core model. We don’t have any problems with the open-core model as such, but the issue, in my opinion, is that some companies tend to focus on the non-free product, the one that is paying their payrolls. And there have been instances where we had to abandon a product, because the open source version available was either abandoned or didn’t have the features that we needed. We want to always have an easy way out. This is why we have been talking with a lot of people about what they’re doing, what they’re using, what are their plans, and both users and vendors.

LM: You mentioned how privacy is an important consideration for the Wikimedia Foundation. You enforce SSLs and don’t use public CDNs and public clouds. In terms of infrastructure, what complications do these steps introduce that you need to address?

JC: Increased complexity. So it would be very easy if we were in a different environment where privacy was an afterthought. We could just say, “Oh, let’s put things in the cloud” and be done with it. The people we had to hire to be at the data centers and many of the things that we do will be taken care of automatically. The way we face it is “no, we want to host our own data.” The data

is not ours, in a way. Well, first we try to store the minimal amount of data, and, when I talk about private data for example, I don’t even mean things like publicly identifiable information. We of course have passwords and user accounts, but typically private data means useful things to preventing vandalism, since you can even edit Wikipedia without an account.

We have a responsibility. It’s not our data, those are not our edits, so we have to safeguard them. As a DBA, this is something that I’m very worried about. It has to be available, but also it has to be protected. As a foundation, our value is the trust the users have in what we do.

What does that imply in terms of infrastructure is that we have to host everything ourselves. So for example, typically you would have an external CDN; they would handle for you most of the traffic, the caching layers, and other things. The disadvantage to that, in most cases, [is that] they will ask you for your private certificates to serve the TLS termination. Basically, they will announce: “We are Wikimedia Foundation, send traffic to us,” and we don’t want to do that.

What are the consequences? Sometimes that creates a lot more work. We give us more work, because I have to handle bare metal. We create a bit of extra work, since we have to self-manage everything, but that is a value that our users demand.

My answer could look like as if “the cloud” is the way to go, and there are only advantages, while going bare metal is a “mere policy” reason. Not at all. Bare metal and self hosting also have a lot of advantages technically, like more independence and control on day-to-day operations, specially during outages. Being able to have a person physically on the data center has saved the day for us many times.

LM: What changes do you plan to introduce to the system in the future?

JC: As far as databases are concerned, one of them is backups. I particularly am concerned about the long-term survivability of the project. If we lose data, we lose the edits and that’s a big deal. So right now I am working on focusing a lot on improving our backup system. Just to

reassure people, we’ve always had a backup system. The problem that we have is that the backups haven’t historically been fast to recover from. After the work we did, they are now, but we still have a lot of work to do.

Secondly, since we use Debian in most cases for our base system and now that we have completed the database upgrade from MariaDB 10.0 to 10.1 a few months back, we would like to migrate from Stretch to Buster. And as part of that, it is most likely that we will upgrade the database as well. The third thing we’re working on always is improving automation.

LM: Finally, what’s a typical day in the life of a Wikipedia DBA?

JC: I am not sure if there is a typical day, because there’s always something happening. We don’t have a lot of outages, but we may have from time to time things that may be not going as fast as they normally do, such as long running queries, plus there are things such as maintenance and upgrades, almost always ongoing. So there is two kinds of work. One is more reactive: response to incidents, user and developer requests. And then there are projects – more long term tasks such as major version migration or a roll in of a new feature.

For the reactive tasks, it mainly involves interacting on Phabricator (our ticket management system) [5] and answering questions and requests, as well as code reviews or small code patches. Regarding projects, I am currently focusing on backups, so that involves quite some more design discussions and larger programming sessions.

The good and, at the same time, the bad thing about our infrastructure is that there is always something to work on! ■■■

Info

- [1] Cumin: <https://wikitech.wikimedia.org/wiki/Cumin>
- [2] PyBal: <https://wikitech.wikimedia.org/wiki/PyBal>
- [3] gdnasd: <https://github.com/gdnasd/gdnasd>
- [4] Blog post on MariaDB migration: <https://blog.wikimedia.org/2013/04/22/wikipedia-adopts-mariadb>
- [5] Wikimedia Phabricator: <https://phabricator.wikimedia.org>

MEET OVER 300
LEADING SUPPLIERS AND HEAR
FROM OVER 250 INDUSTRY SPECIALISTS.
BOOK YOUR FREE TICKET TODAY:
WWW.CLOUDEXPOEUROPE.COM/LINUX



Technology that performs for you.

Everyone's looking for flexibility and performance when it comes to cloud technology. At Cloud Expo Europe 2020, every up-to-the-minute solution you're looking for will be on show. Experience all-encompassing technology, applications, strategic guidance and actionable insights that will get you performing beautifully. In a fast-changing industry, Cloud Expo Europe 2020 has all the answers.

Register for your **FREE** ticket today:
www.cloudexpoeurope.com/free



11 - 12 March 2020 ExCel London
www.cloudexpoeurope.com

ORGANISED BY  **CloserStill**

PART OF
**LONDON
TECH SHOW**

INCORPORATING

- 
**CLOUD EXPO
EUROPE**
- 
**DEVOPS
LIVE**
- 
**CLOUD & CYBER
SECURITY EXPO**
- 
SMART IOT
- 
**BIG DATA
& AI WORLD**
- 
**BLOCKCHAIN
TECH WORLD**
- 
**DATA CENTRE
WORLD**

ORGANISED BY
 **CloserStill**

EVERY EMERGING TECHNOLOGY. ONE DIGITAL TRANSFORMATIONAL JOURNEY.

HEADLINE SPONSOR



MANAGED CLOUD SPONSOR:



THEATRE SPONSORS



STREAM SPONSORS



GOLD SPONSORS



SILVER SPONSORS





Block ads and trackers across your network with Pi-hole

The Trickster

The Pi-hole ad blocker filters ads and trackers from the data stream for all devices on the network, from your smartphone to your toaster. *By Christoph Langner*

Internet users, content providers, and ad blocker developers are in a constant arms race. Users have deployed ad-blocker web browser extensions for years, and these extensions work quite well for standard web pop-ups and banner ads. But browser extensions are a little more trouble to implement on cell phones and other mobile devices. Also, ads built into apps typically remain untouched by the filters imposed by browser extensions. In addition, conventional ad blockers do nothing to stop modern Internet-connected devices like smart TVs, stereos, and even washing machines from transmitting data to the Internet in a very talkative way.

Other alternatives have developed in recent years to give users new tools for stopping Internet ads in a more global and comprehensive way. Pi-hole [1] is a promising tool that provides a centralized means for stopping Internet advertisements across a local network. The Pi-hole developers refer to Pi-hole as a “black hole for Internet advertisements.” In more technical terms, Pi-hole is what is often called a “DNS sinkhole” [2]. A DNS sinkhole is a DNS server that gives out unroutable IP addresses for domains that are listed in a “sinkhole” list, which is basically a blacklist. Because Pi-hole leverages a standard process that is built into all TCP/IP networks (the DNS lookup process), it doesn’t require any client applications or special configuration, other than to point the client to the Pi-hole DNS server, which can happen automatically through DHCP.

Sinking the Putt

Pi-hole combines common Linux-based network tools such as the DNS forwarder dnsmasq with a lighttpd web server and other Linux tools. As the name suggests, many users install the program on a Raspberry Pi. In addition to Raspbian, the project also supports Debian, Ubuntu, Fedora, and CentOS (see the box entitled “Pi-hole on Linux”).

In principle, you should install Pi-hole on a computer on a LAN that runs 24/7. As soon as you configure your network for Pi-Hole, you’ll need a working Pi-Hole server or Internet access will not function properly. This need for continuous operation is one



Pi-hole on Linux

In our test with Ubuntu 18.04 and 19.04, we had no problems installing Pi-hole. However, users should be aware that Pi-hole intervenes quite deeply in the system. The installation routine deactivates the integrated DHCP client and replaces it with `dhcpcd5`, and the system sets up a static IP address. If you want to change the IP address later, call `pihole -r` with administrative privileges and select the *Reconfigure* option.

Listing 1: Installing Pi-hole

```
$ wget -O basic-install.sh https://install.pi-hole.net
$ sudo bash basic-install.sh
[...]
[+] Pi-hole Enabled
[i] Web Interface password: 1lw32Fil
[i] This can be changed using 'pihole -a -p'
[i] View the web interface at http://pi.hole/admin or http://192.168.188.29/admin
[i] You may now configure your devices to use the Pi-hole as their DNS server
[i] Pi-hole DNS (IPv4): 192.168.188.29
[i] Pi-hole DNS (IPv6): fd00::ef26:23c0:a6fd:eeac
[i] If you set a new IP address, please restart the server running the Pi-hole
[i] The install log is located at: /etc/pihole/install.log
```



A Gift, but Not for Free

Many DNS providers offer their services without requiring payment, but they are by no means free. That Google likes to collect data is well known. OpenDNS is now part of network giant Cisco. Quad9 is backed by IBM and Packet Clearing House (PCH), as well as the Global Cyber Alliance, which was founded by the police authorities of London and New York. The service promises not to store personal data, but its proximity to government agencies is enough to set the alarm bells ringing for some users.

reason a Raspberry Pi is often used as a Pi-Hole server: Even a brand new Rasp Pi 4B costs only EUR35 (~\$39) and hardly needs any electricity. Pi-hole itself requires only a limited amount of resources, so you can also use the Pi for other tasks.

Pi-hole is installed via a script you can download from the web using the commands in Listing 1. You'll need to run the `basic-install.sh` script with administrative privileges. At the end, the setup script displays the URL and a random password for the web interface, which you can change if necessary using the command `pihole -a -p`.

DNS Options

During the installation, you have to answer a number of questions: For *Upstream DNS Provider* (Figure 1) you have a choice between the DNS servers of Google, OpenDNS [3], and Quad9 [4] (see the box entitled “A Gift, but Not for Free”). Optionally, select *Custom* and enter any DNS servers in the system (one after the other, separated by commas), such as those operated by your Internet provider.

The installation routine asks which ad-blocker and anti-tracker lists you want to use. For the most comprehensive protection possible, leave all preselected options enabled. You will then need to configure the network settings. The setup automatically detects whether to enable IPv4 and IPv6. Then the program detects the current IPv4 address and asks if it should use this address automatically in the future. The *IPv4 default gateway* you need to specify is usually your router's IP address; however, the setup typically detects the gateway automatically.

Static IP Address

To avoid IP conflicts, open the settings of your wireless router and mark the IP address of the Pi-hole machine as static. For a FRITZ!Box router, for example, you will find the option *Always assign the same IPv4 address to this network device* by editing the device below *Home network | Network*. Alternatively, adjust the IP address entered on the Pi-hole server so that it comes from a range that the wireless router does not use (FRITZ!Box: *Home network | Network | Network settings | IPv4 addresses*).

If you want to change the configuration of Pi-hole later on, call the installation routine again with the `pihole -r` command. You then have the choice between *Repair*, which transfers the existing settings cleanly into the system again, and *Reconfigure*, with which you repeat the setup, specifying the previous settings.

For the remaining questions, you will not want to change the default selection. These questions allow you to (de-)activate the web-based admin interface, install the `lighttpd` server (also known as “Lighty”), and choose if the system should log data later on. *Privacy Mode* allows variants from

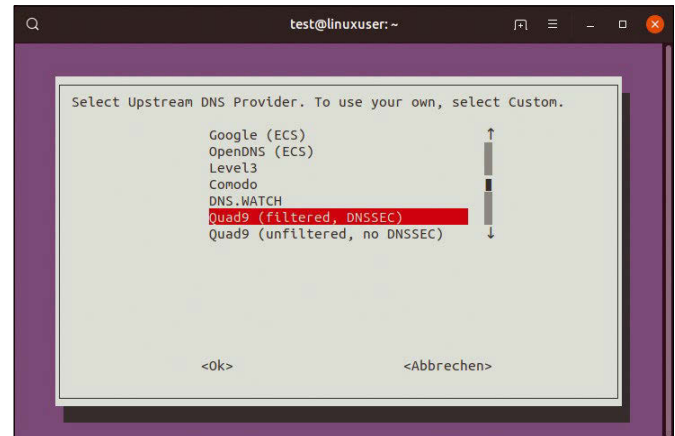


Figure 1: Pi-hole itself needs a DNS server for name resolution. The installation script comes with a selection of built-in DNS options.

0 Show everything to 3 Anonymous mode and also allows complete deactivation of all statistics.

Finally, the system shows a summary with the most important data, the path to the installation log, and the URLs through which you can reach the system in the future (Figure 2). This information can also be found as output in the terminal. After the completion of the setup script, you will only have to reboot if you have changed the IP address of the system.

Network Configuration

If you want the devices on your network to use the Pi-hole ad blocker as their DNS server, you can enter the configuration either on the clients or in the DHCP server configuration for the wireless router (see the box entitled “Local Name Resolution”). The router option offers the advantage that all devices active on the network, from smartphones to WLAN-enabled coffee machines, automatically use the DNS filter without further configuration.

For a FRITZ!Box, you will find the required configuration in *Home network | Network | Network settings | IPv4 addresses*. Enter the IPv4 address of the Pi-hole system below *local DNS server*. Repeat the same procedure in the *IPv6 address settings*. There is a separate section for the *DNSv6 servers on the home network*.

As soon as you restart a client's network connection, for example, using the Network Manager of the desktop environment,

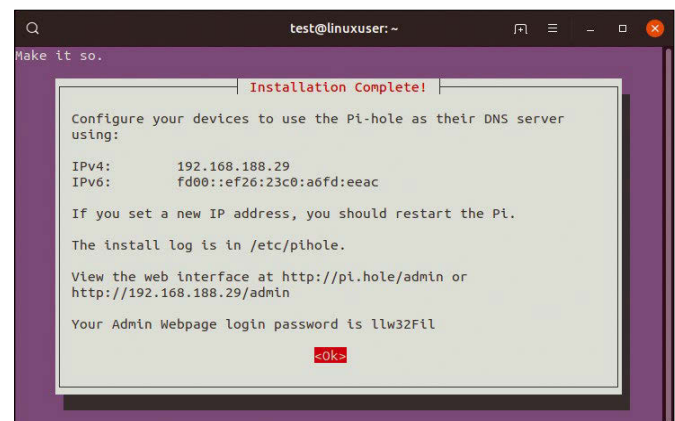


Figure 2: At the end of the installation, the setup script summarizes all the essential details of the Pi-hole system.



Local Name Resolution

The DNS server in the wireless router usually also handles name resolution on the local network. For example, you can control a Raspberry Pi running on the network via `ssh pi@raspberrypi` without knowing its IP address. The router knows the computer name assignments and IP addresses, because it assigns the IPs itself via its DHCP server. However, if you now route the DNS queries via the Pi-hole blocker, you lose this convenience as Pi-hole does not receive any information from the router. In order to be able to continue using host names on the network, you need to activate the *Settings* | *DNS Use Conditional Forwarding* option in the settings of Pi-hole and enter the IP address of the router as well as the local domain names (in the case of a FRITZ!Box, `fritz.box`). Pi-hole then forwards requests for local network names to the router.

For example, an Android smartphone used in the test network did not want to use the Pi-hole DNS until after a restart. And, yes, we did try turning it off and back on again.

You can check the effect of the Pi-hole ad blocker on Linux with the `dig` command (Listing 2). For example, the output in Listing 2 shows that the system uses a computer with an IP address of 192.168.188.11 as its DNS server and only receives an IP address for the `google.com` domain. However, querying the IP of `google-analytics.com` returns `0.0.0.0`; the query disappears into a black hole, provoking a time out. See the box entitled “Unique Local Addresses” for information on performance issues that might occur with Pi-hole.

Admin Back End

The configuration interface of the Pi-hole server can be accessed via `http://pi.hole/admin` or, if the name resolution does not work, by typing `http://IP_address/admin` in your browser’s address bar. Clicking on the *Login* entry in the sidebar lets you log in with the password randomly generated during installation. If you have forgotten it, you need to log on to the system via SSH or locally and set a new password with the `sudo pihole -a -p` command.

The system welcomes you to the dashboard, which displays a number of statistics (Figure 3). If logging is enabled, you will also find the most frequently queried domains and the domains most often blocked by Pi-hole. Details on the current queries can be found via *Query Log*; the overview in *Long term data* lets you view statistics for any time period. (Figure 4).

If this detailed logging of your Internet activities offends your sense of privacy, you can adjust the logging level in *Settings*

Listing 2: Checking Pi-hole

```
$ dig google.com
[...]
google.com.          50      IN      A       216.58.210.14
;; Query time: 9 msec
;; SERVER: 192.168.188.11#53(192.168.188.11)
;; WHEN: Mo Aug 05 21:10:09 CEST 2019
;; MSG SIZE  rcvd: 55
$ dig google-analytics.com
[...]
google-analytics.com.  2       IN      A       0.0.0.0
;; Query time: 2 msec
;; SERVER: 192.168.188.11#53(192.168.188.11)
;; WHEN: Mo Aug 05 21:11:09 CEST 2019
;; MSG SIZE  rcvd: 65
```

the system should use Pi-hole as its DNS server. To be on the safe side, it is a good idea to restart all clients one time. For ex-

ample, you can also use the `sudo pihole logging off` command to disable the syslog in `/var/log/pihole.log`.

If the website fails to load as desired, try disabling Pi-hole’s filter function by selecting *Disable*. The menu gives you the choice of a time out between 10 seconds and up to 5 minutes, an arbitrary time of your choosing, or you can completely disable the filter. Once you disable the filter, the *Active* status top left above the sidebar shows red.

If you urgently need to visit a site classified as problematic, you can add the domain to your *whitelist*. The *blacklist* lets you add more sites you want to block. You can use regular expressions and wildcards when adding domains to the lists.

Double Filter

To investigate the impact Pi-hole has on loading times, we decided to do a comparison test with three German news portals: Spiegel Online, Welt.de, and Golem.de. We used the version 76.0.3809.87 web browser in the plain vanilla version, and then with the uBlock Origin ad-blocking add-on, and finally without an ad blocker, but with the DNS filter courtesy of Pi-hole 1.21.6. Finally, we investigated what a combination of all these filters offered.

The figures in Table 1 show that the dedicated ad blockers accelerate the load action far better than the Pi-hole filter left

Unique Local Addresses

After entering the Pi-hole system as DNS server, it can happen that the loading times of web pages deteriorate noticeably. If so, check the IPv6 settings in the router’s administration interface – in the case of FRITZ!Box, below *Home network* | *Network* | *Network settings* | *IPv6 addresses*. In the *Unique Local Addresses* section, you will usually find the option *Assign Unique Local Addresses (ULA) as the default setting, as long as there is no IPv6 Internet connection*. This option prevents the router from assigning unique local addresses (the IPv6 counterpart to private IPv4 addresses) when an IPv6 address is provided by the Internet provider.

Due to the privacy extensions implemented in IPv6, the IPv6 address – also that of the Pi-hole server – continually changes, and DNSv6 queries then disappear in a black hole. If the loading times are noticeably longer, enable the *Unique Local Addresses (ULA) always assign* option and then restart the Pi-hole server. Ideally, you will want to check this setting before installing Pi-hole. The unique local address of the Pi-hole system can be determined using the `ip -6 addr` command. Pay attention to the `inet6` line, which starts with `fd00` or the prefix you set in the FRITZ!Box configuration. Enter this address in the format `fd00:0:0:0:aaaa:bbbb:cccc:dddd` as your *Local DNSv6 server* in the *IPv6 addresses* section. Finally, you need to update your Pi-hole configuration by typing `pihole -r`; the system will now use the new IPv6 address.



to its own devices. Despite this, you can use a combination of all of these technologies, that is, ad blocker plus Pi-hole, to reduce the loading times by another 10 to 20 percent. The speed benefits turned out to be particularly spectacular for loading pages plastered with ads. The Welt.de site came up six times faster in our lab.

Conclusions

Pi-hole reliably prevents network-capable domestic appliances from opening connections to known trackers and thus transmitting usage data. Conversely, this does not mean that Pi-hole always prevents tracking. For example, if your fridge insists on sending data to the vendor's server, you might need to discover the vendor's address and add it to your blacklist manually.

Having said this, it was clear – after a few days – that smartphones and network-aware multimedia devices were benefiting from the Pi-hole server's privacy filter. The domains most frequently blocked included both the typical ad networks, but also analysis tools that specialize in acquiring user data from mobile devices. And the Pi-hole put an end to the verbosity of a Samsung TV set, as well as a set of Sonos active speakers, without needing any additional configuration overhead.

In terms of the web browsing experience on a PC, Pi-hole offers slightly less than a full-blown ad-blocking add-on like uBlock Origin. The classical ad blocker is more convenient to configure, and it accelerates rendering of the page more than the Pi-hole DNS filter left to its own devices. You can achieve optimum results with a combination of an ad blocker and Pi-hole. This combination improves the loading times of many websites, reducing them to a fraction of the time taken by the unfiltered originals. At the same time, Pi-hole filters out ads and trackers from the data stream, even if the devices or programs themselves do not offer filtering options. ■■■

Info

- [1] Pi-hole: <https://pi-hole.net/>
- [2] DNS Sinkhole: https://en.wikipedia.org/wiki/DNS_sinkhole
- [3] OpenDNS: <https://www.opendns.com>
- [4] Quad9: <https://www.quad9.net>

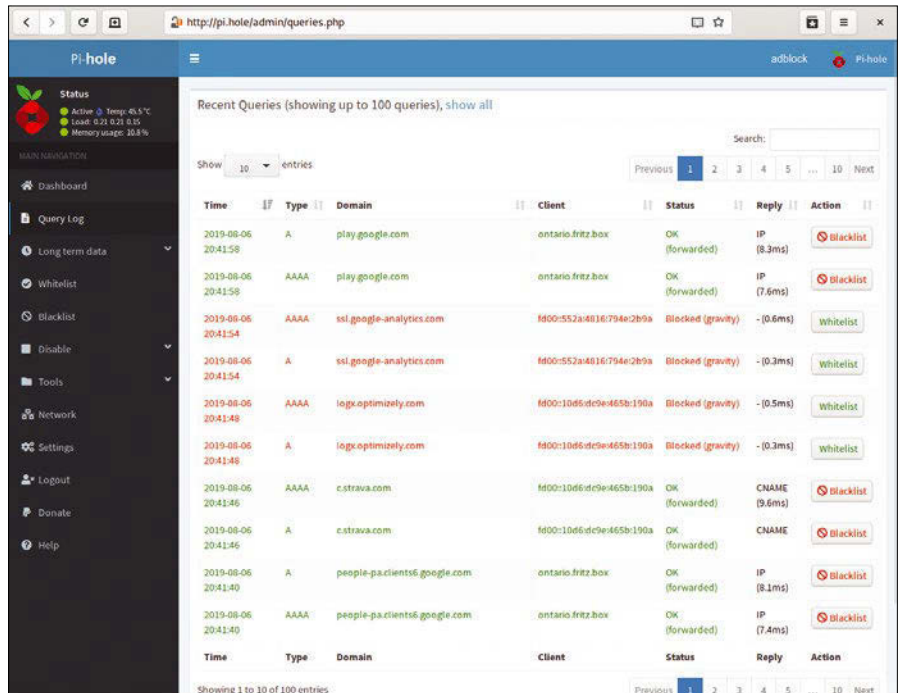


Figure 3: The dashboard for the Pi-hole admin interface delivers information on the ad blocker's status.

Table 1: Loading Times Compared

	Chrome	Pi-hole	uBlock Origin	Pi-hole + uBlock Origin
Version	76.0.3809.87	04.03.01	1.21.6	1.21.6 + 04.03.01
SpOn	7459 (± 194)	3490 (± 690)	1928 (± 64)	1959 (± 122)
Welt	9817 (± 436)	5862 (± 360)	2210 (± 272)	1644 (± 202)
Golem	6778 (± 458)	2619 (± 93)	506 (± 69)	463 (± 38)

All of these times are shown in milliseconds (rounded). Mean values for five rounds of testing are determined using the Chrome Dev Tools.

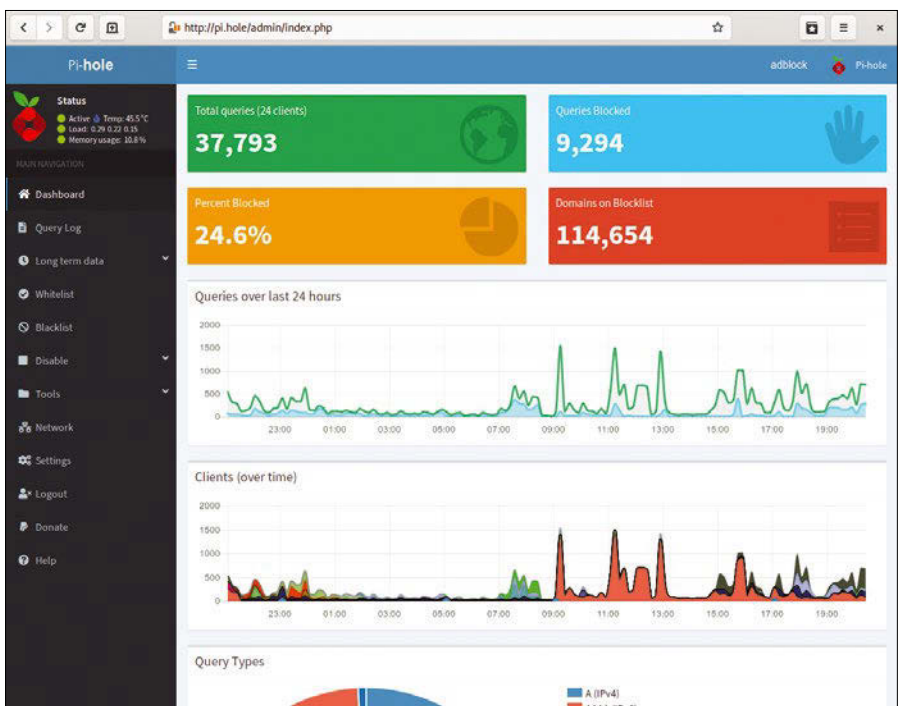


Figure 4: Viewing log term data in the Pi-hole query log.



Killing ads with the LAN-level Privoxy web proxy

Interceptor

Add-on ad blockers can help mitigate the degradation of your browsing experience, but sometimes you need to bring stronger weapons. A filtering web proxy can scrub web traffic to eliminate unwanted ads and scripts. *By Rubén Llorente*

Most people see no ethical issues with the fact that websites display advertisements. Unfortunately, for lots of technical reasons, advertisements are often undesirable, to the point that removing them is sometimes necessary for a satisfactory surfing experience. (See the box entitled “Why Remove the Advertisements.”)

Many users opt to use an ad blocker such as uBlock Origin [1] or Adblock Plus [2] in their web browsers. These blockers, which are installed as browser add-ons, are easy to set up and offer high quality advertisement filtering. However, they only work in the browser they are installed in, they aren’t intended to work with smartphones and tablets, and they require significant duplication of labor if you need to support multiple computers on a local area network (LAN).

An earlier article in this issue described one possible network-based ad-blocking solution: the Pi-hole DNS tool. This article shows how to block ads using the Privoxy proxy server.

A proxy server is a tool that makes connections on behalf of other programs. Under regular operation, you would set up a proxy server in your LAN and configure every web browser in your network to use it. When a proxy-aware browser attempts to access a website, it connects to the proxy and asks the proxy to access the site. The proxy then establishes the connection, downloads the web data, and hands it to the browser that made the request. (Figure 1 shows a simple proxy setup.)

Proxies are useful for content filtering, such as detecting advertisements and sending an ad-free version of the site to the client browser (Figure 2). The most

Author

Rubén Llorente is a mechanical engineer, whose job is to ensure that the security measures of the IT infrastructure of a small clinic are both law compliant and safe. In addition, he is an OpenBSD enthusiast and a weapon collector.

Why Remove the Advertisements?

If advertisements are necessary for the sustainability of free websites, why remove them?

The first reason is that many sites are overloaded with advertisements to the point that they are barely usable. Advertisements mean longer page load times and higher bandwidth consumption. Some advertisements in websites are so invasive that users will spend more time closing ads and pop-ups than reading what they wanted to read in the first place. The user is often confronted by the choice between having a bad user experience or installing an ad blocker.

On top of that, some people have issue with the fact that they are paying an ISP and buying a data plan from them, only to make it possible for somebody else to use a big percentage of their bandwidth to stuff it with advertisements. Users are literally paying for the data channels that serve advertisements to them.

One of the biggest worries regarding advertising is the surveillance capabilities advertisers have. Most often, advertisement agencies use methods to know and register which sites you

have been visiting, for how long, and other details that will help them know which advertisements to show to you in order to try to sell you things. Although trying to find what you like in order to offer it to you is not extremely evil behavior, it can lead to the advertiser having too much information about you. For example, by extrapolating the websites you visit, advertisers can find out details about your political affiliations or health issues.

The biggest concern is, however, security. Advertisements include pieces of code that are forced into your web browser and executed without your permission. Some advertising systems have been compromised by evil entities and corrupted into serving malware instead of regular advertising. Since advertising networks serve a very, very big number of ads to an enormous number of users, a single compromised advertiser can distribute an astounding amount of malware. An ad blocker protects you from the dangers associated with downloading and executing this malicious code.



that has no ad blocker and is not configured to use a proxy, the router takes the connection that would be sent to that website and instead sends it to the proxy. The proxy then knows that your aunt wants to visit that site, fetches it, and serves your aunt a version of the site that is advertisement free. The computer your aunt is running does not get to know that there is a proxy anywhere in the LAN. It is effectively a magnanimous man-in-the-middle (MITM) attack. Figure 3 displays how interception mode works.

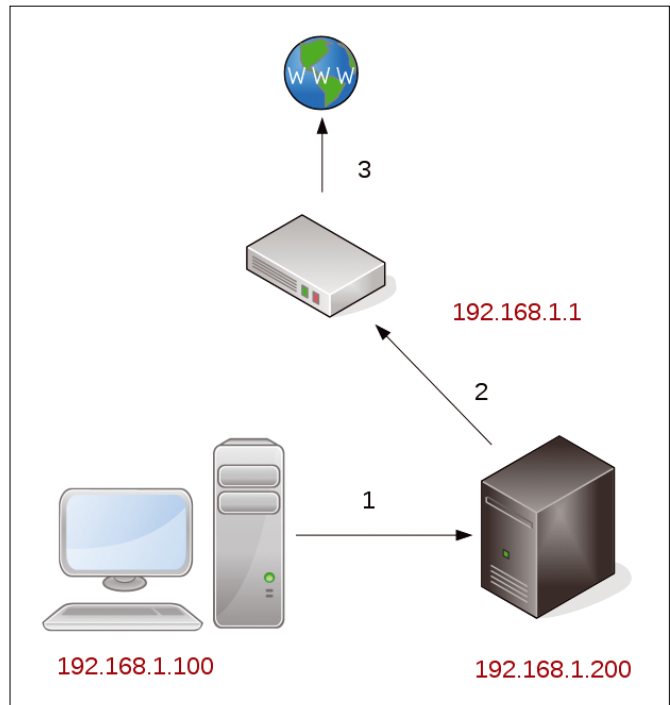


Figure 1: 1) The workstation with IP address 192.168.1.100 connects to the server running a web proxy at 192.168.1.200. 2) The proxy attempts to fetch a website in the name of the workstation. 3) The gateway, a router with internal IP 192.168.1.1, forwards the connection out of the LAN and into the Internet.

common technique is for the proxy to return a 404 HTTP error (*Not Found*) to the client when the

browser attempts to download an advertisement. This approach is beneficial, because it prevents the advertisement from being downloaded and wasting your bandwidth.

The drawback of regular proxy operation is that you still have to configure every device in your network to use the proxy, so while a proxy allows you to maintain all your advertisement filter configuration in one place, making administration easier, it is not a silver bullet that solves all your problems. However, if your router has decent packet filtering capabilities, you can pull quite a neat trick: Use the proxy in *interception mode*.

Interception mode means the router detects web traffic that is trying to bypass the proxy and hands it to the proxy anyway. When your aunt accesses *page-full-of-ads.eu* with a computer

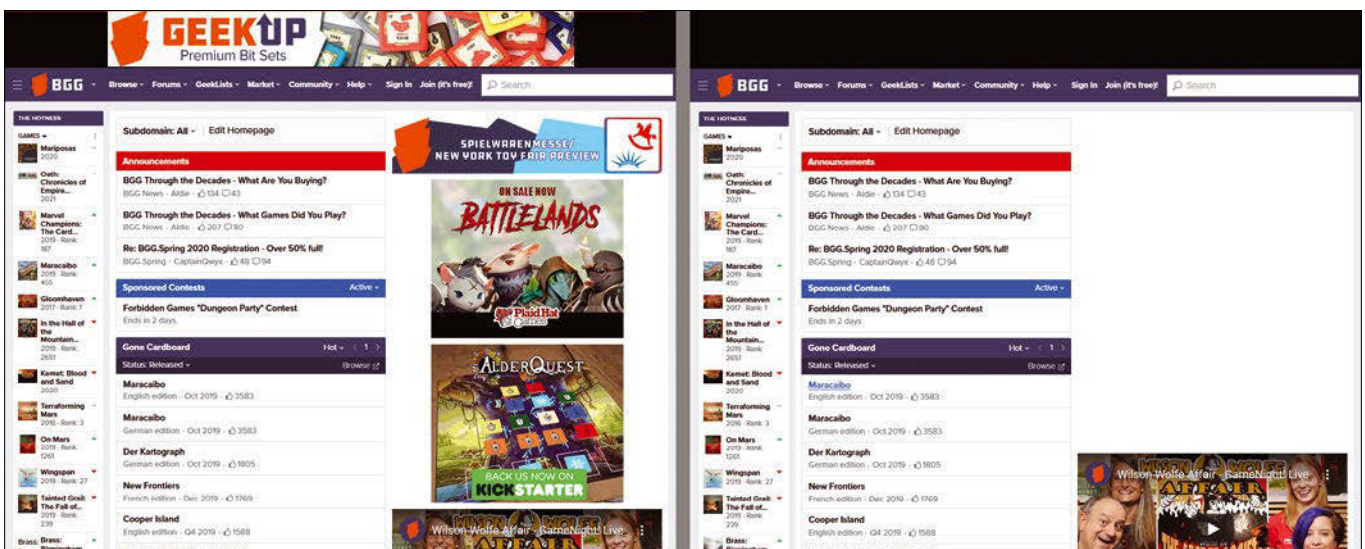


Figure 2: The website (left) comes up faster and looks better when processed by an ad blocker (right).

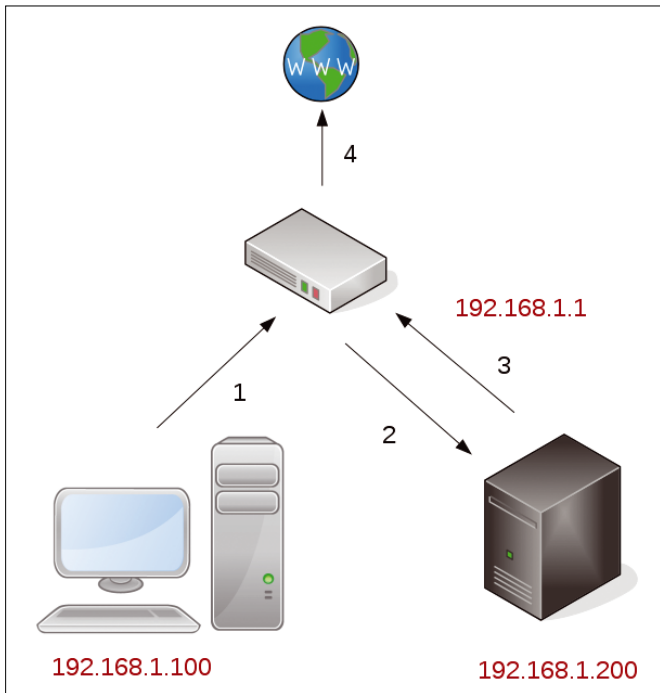


Figure 3: 1) Interception mode: The workstation with IP address 192.168.1.100 attempts to connect to a website located somewhere in the Internet, using no proxy. 2) The router that acts as a gateway for the LAN intercepts the connection and redirects it to the proxy server at 192.168.1.200. 3) The proxy attempts to retrieve the website from the Internet in the name of the workstation. 4) The router forwards the connection to the Internet. Note that the workstation never connects to the external website directly even though it isn't configured to use the proxy.

Installation and Configuration

Privoxy [3], which is a derivative of the discontinued *Internet Junkbuster*, has the ability to block unwanted traffic and also modify traffic that passes through in order to make it safe (for example, disable canvas fingerprinting and other tracking methods in JavaScript code).

In order to get started with Privoxy, you need to install it first. Installation in the Devuan distribution is as simple as:

```
$ su -  
[password]  
# apt-get update  
# apt-get install privoxy
```

By default, Devuan launches services that have just been installed. Since Privoxy has not been configured yet, disabling it is advisable:

```
# /etc/init.d/privoxy stop
```

The configuration of the proxy program is stored in `/etc/privoxy`. Figure 4 shows a list of the configuration files. The user is strongly encouraged to modify only `config`, `user.filter`, and `user.action` and leave the rest of the files alone. The files `default.ac-`

Listing 1: Custom Rules

```
01 # These rules block the listed hosts.  
02 {+block{Advertiser}}  
03 *.infolinks.com  
04 static.criteo.net  
05  
06 # There rules block the listed hosts. The advertisements  
   are replaced by harmless placeholders.  
07 {+block{Advertisement Image} +handle-as-image}  
08 *.amazon-adsystem.com  
09 onlinebookclub.org/images/banners/  
10  
11 # This rule instructs Privoxy to apply a custom filter,  
   called "killscript", to the hosts featured in the list.  
12 # The filter must be defined in /etc/privoxy/user.filter  
13 {+filter{killscript}}  
14 elpais.es  
15 *.elpais.es
```

Listing 2: user.filter Example

```
01 FILTER: killscript  
02 s/<script[^>]*.*?</script>/igs
```

Listing 3: Incorporating the StevenBlack Blacklist

```
01 $ curl -o stevenblacklist.txt  
02 https://raw.githubusercontent.com/StevenBlack/hosts/  
   master/hosts  
03 $ su  
04 [password]  
05 # echo "{block {StevenBlacklist}}" >>  
   /etc/privoxy/user.action  
06 # grep '^0\.\0\.\0\.\0' stevenblacklist.txt |  
   awk '{print ""$2"}' | sort >> /etc/privoxy/user.action
```

`tion` and `default.filter` include the default filters and blocklists and are updated every time Privoxy itself is updated. If the user makes modifications on these files, they will be lost when Privoxy is bumped to a new version.

The `config` file is long and contains the main configuration for the proxy. Fortunately, defaults are good enough for most users. In order to get started, the administrator just has to locate a line that reads

```
listen-address 127.0.0.1:8118
```

and replace it with:

```
listen-address 0.0.0.0:8118
```

The proxy service will now listen on every network interface of the server at port 8118. Some administrators might prefer to use port 8080, which is conventionally used for HTTP prox-

```
devuan@devuan:~$ ls /etc/privoxy  
config          match-all.action      trust  
default.action  regression-tests.action user.action  
default.filter  templates              user.filter  
devuan@devuan:~$
```

Figure 4: Contents of `/etc/privoxy`.

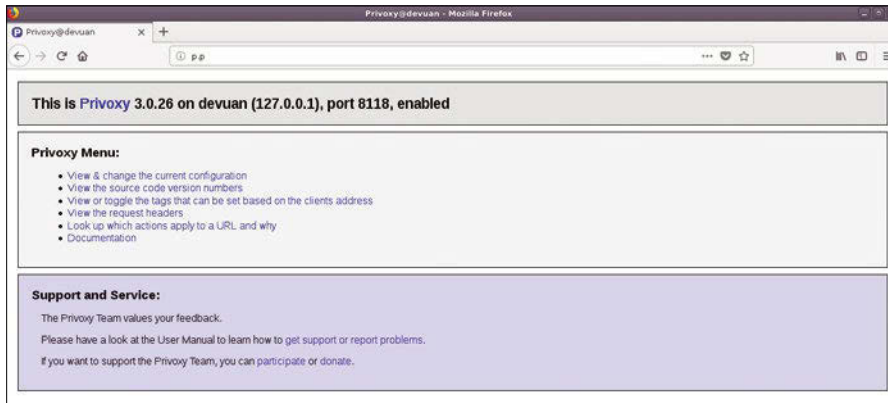


Figure 5: If Privoxy is working, and your browser is configured to use it in direct mode, visiting special website `http://p.p` will take you to Privoxy's information panel.

ies. It is very important that this port is not accessible from the Internet. Most home routers act as firewalls that prevent Internet access to this port, so it is not an issue. The `config` file is very well documented, and you may perform additional changes if needed. For a quick summary of the configuration, go to Privoxy's information panel (Figure 5).

The default blacklist used by Privoxy is functional but not great. The user may add custom items to the blacklist by appending them to `user.action`. The file is well commented and contains examples, and many macros are included. The most basic usage is to append sites and advertisement providers that you don't want to see but aren't included in the default filters. Listing 1 shows some example rules for blocking content. Advertisements that are very graphic in nature can be handled as images. Advertisements that are handled as images are replaced by a harmless, placeholder pattern.

The `user.filter` file is used to define custom filters. Filters are special instructions that replace content in the web pages that the proxy serves to the user. The filtering feature is quite advanced but also very useful, as it allows you to rewrite dangerous JavaScript code before it is delivered to the user. Filters are written using Perl replacement syntax. For example, the filter in Listing 2 deletes any script present in any HTML document that goes through the proxy.

A list with the sites I want to subject to this stern filtering policy must be defined with a custom rule in the `user.action`

Listing 4: Incorporating the Easylist Blacklist

```
01 $ curl -o easylist.txt https://easylist.to/easylist/
    easylist.txt
02 $ su
03 [password]
04 # echo "{block {Easylist}}" >> /etc/privoxy/user.action
05 # sort -u easylist.txt | grep ^\|\.|^$ | grep -v \| |
    sed 's/[|^]//g' >> /etc/privoxy/user.action
```

Listing 5: Configuring a Proxy

```
01 # echo "export \"http_proxy=http://192.168.1.200:8118\"" >
    /etc/profile.d/proxy_configuration.sh
02 # echo "export \"https_proxy=http://192.168.1.200:8118\""
    >> /etc/profile.d/proxy_configuration.sh
```

file (see the box entitled "Populating a Blacklist").

Once the configuration is done, you can start the service. The following command will make your proxy available in the LAN:

```
# /etc/init.d/privoxy start
```

Client Configuration

The easiest way to get a web browser to use a proxy server that is running in your LAN is through the browser's preference options. Most operating systems allow you to configure a preferred proxy that programs will be forced to use [7] [8]. On Devuan, you can configure a

workstation to use the proxy by issuing the commands described at Listing 5.

These commands will make most programs that are installed in the workstation and are capable of using a web proxy take advantage of the Privoxy instance. These commands assume that the proxy server is located at address `192.168.1.200` and that Privoxy listens on port `8118`. Change these settings to suit your needs.

Populating a Blacklist

Many blacklists are available for blocking unwanted sites and content. In addition to stopping advertisers, some blacklists also prevent malware and phishing sites. These blacklists are not published in a Privoxy-friendly format but may still be used with some creativity.

In a previous article [4], I made use of the StevenBlack blacklist [5]. You can incorporate the StevenBlack list into Privoxy by issuing the commands shown in Listing 3. Listing 4 describes how to incorporate the alternative EasyList [6] blacklist.

Listing 6: Example Firewall Rules for an Intercepting Router

```
01 EXTERNAL_INTERFACE=eth1
02 INTERNAL_INTERFACE=eth0
03
04 # Masquerade outgoing traffic.
05 iptables -t nat -A POSTROUTING -o $EXTERNAL_INTERFACE -j
    MASQUERADE
06
07 # Redirect connections that attempt to talk to external
08 # web servers without using the proxy.
09 iptables -t nat -I PREROUTING -i $INTERNAL_INTERFACE -p
    tcp ! -s 192.168.1.200 --dport 80 -j DNAT
    --to-destination 192.168.1.200:8118
10
11 # Permit loopback traffic, allow traffic generated by the
    router itself, and drop connections from the WAN.
12 iptables -A INPUT -i lo -j ACCEPT
13 iptables -A INPUT -m state --state RELATED,ESTABLISHED -j
    ACCEPT
14 iptables -A INPUT -i $EXTERNAL_INTERFACE -j DROP
```

Enable Interception Mode

You can enable interception mode by searching for the `accept-intercepted-requests` parameter in `/etc/privoxy/config` and changing its value from `0` to `1`. This step allows the proxy to process HTTP requests that are intercepted with the help of the router in the network.

Admittedly, most consumer-grade routers lack the ability to perform MITM attacks against devices located in the LAN at the command of the system administrator. There are many ways to intercept a connection headed to one place and hijack it into moving somewhere else (such as your intercepting proxy). I am covering the most natural approach here, which consists of using the router to perform *Network Address Translation* on the connection. In the following example, it is assumed that your router has a netfilter stack with iptables, which is the case for most common Linux distributions. If your router is performing classical masquerading (as most home routers are configured to do), then adding a single DNAT rule will suffice. Listing 6 shows a simple example configuration. The DNAT rule will instruct the router to catch any web request that does not come from the proxy server and force it to go through Privoxy. You may need to change the values of the network interfaces and the IP address.

What About TLS?

How do you successfully perform a MITM attack against a connection that is encrypted to prevent MITM attacks? Most websites nowadays serve their content using HTTPS, with SSL or TLS encryption, even if that content is not sensitive. HTTPS was explicitly designed to prevent MITM actions. Privoxy cannot break HTTPS in order to look at the contents of the websites the client is visiting and take actions on them, such as modifying harmful JavaScript code before sending it to the client, without generating lots of security warnings in the client's web browser.

It is still possible to use Privoxy and an MITM attack to decrypt the traffic, analyze it, process it, and send it to the client without triggering security warnings. However, the way to achieve this is ugly and requires the collaboration of the client.

The process works as follows: The administrator creates a CA certificate for internal use and installs it in the clients whose connections he intends to make go through the proxy. Then a corresponding private key is installed in the intercepting proxy. When a browser tries to access an HTTPS site and the connection is intercepted, the proxy generates a fake certificate on the fly using the key, which is trusted by the client, and reads the request. Then the proxy fetches the website normally over HTTPS, processes its contents, and sends its output to the client browser using the spoofed connection.

This approach is controversial because it breaks the assumption that TLS connections are not modified by any intermediary on the network. Privoxy does not support this sort of interception out of the box, although there is a lot of demand for it, according to the project mailing list. However, Privoxy can be chained up with an HTTPS interception mechanism such as ProxHTTPSProxy [9].

In normal conditions, Privoxy is only capable of making limited filtering when dealing with HTTPS connections, and only if the client is actively using the proxy instead of being intercepted.

Final Considerations

A proxy service can greatly improve the web browsing experience. However, using a proxy introduces a new set of problems.

The first issue is that, in order to visit a website using Privoxy, the proxy has to download it, process it, possibly rewrite its code, and then send the page to the client once it has been fully processed. Most web browsers are designed to download the components of a website once a user attempts to visit it and start displaying these components as they become available. Privoxy, on the other hand, has to download the whole site and process it, and then send it all at once to the browser. The result is that a site might look unresponsive while it is loading.

The default filters and blocklists included with Privoxy are designed not to break websites. That said, they are not as powerful as browser-based blockers such as uBlock Origin. Custom rules may be added in order to suit your needs, but if you do something wrong, you may end up breaking some websites.

It is usually a good idea to use Privoxy with a caching proxy. A caching proxy is a regular proxy that is able to catch requests and keep local copies of the websites the users visit, so it is not necessary to download them every time a user attempts access them. Squid [10] is probably the best known FOSS caching web proxy.

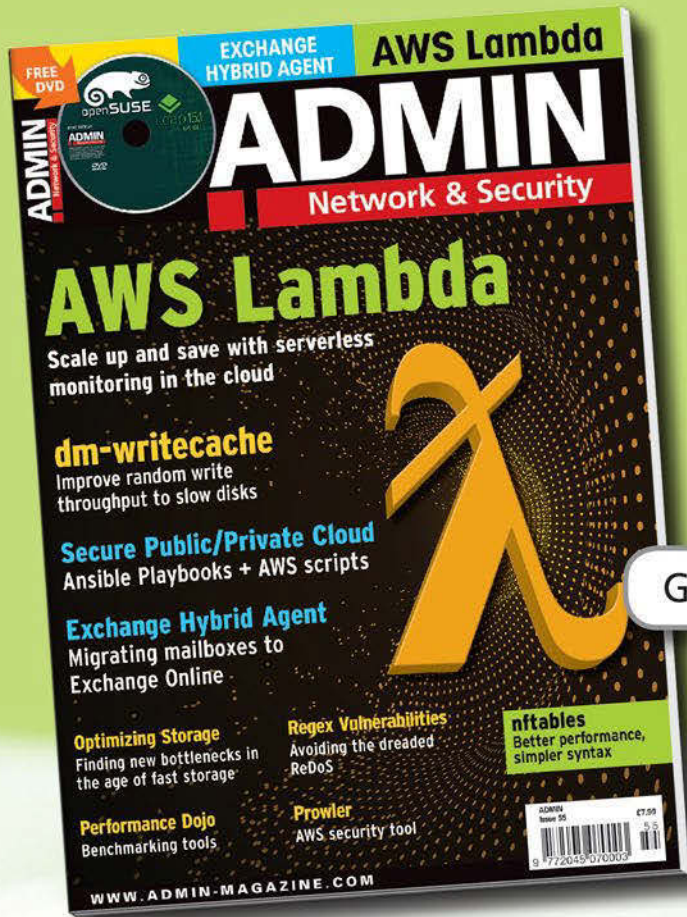
The steps described in this article for implementing a proxy server that is capable of processing websites that use HTTPS in *interception mode* require a lot of work, and it is recommended that you complement this setup with DNS blocking.

Privoxy is very configurable and surprisingly powerful. You can even hack Privoxy into removing EU cookie warnings and performing other clever tricks. For more information on Privoxy, see the project documentation [11]. ■■■

Info

- [1] uBlock Origin: <https://github.com/gorhill/uBlock>
- [2] Adblock Plus: <https://adblockplus.org/>
- [3] Privoxy: <https://www.privoxy.org/>
- [4] "Setting up a local DNS server with Unbound" by Rubén Llorente, *Linux Magazine*, issue 227, October 2019, <http://www.linux-magazine.com/Issues/2019/227/Local-DNS-with-Unbound>
- [5] StevenBlack DNS blacklist: <https://github.com/StevenBlack/hosts>
- [6] EasyList advertisers blacklist: <https://easylist.to/>
- [7] How to configure Android to use a proxy server: <https://hide-ip-proxy.com/configure-proxy-server-android/>
- [8] How to configure Windows 10 to use a proxy server: <https://pureinfotech.com/setup-proxy-server-windows-10/>
- [9] ProxHTTPSProxy: <https://github.com/wheever/ProxHTTPSProxyMII>
- [10] Squid: <http://www.squid-cache.org/>
- [11] Privoxy documentation: <https://www.privoxy.org/user-manual/index.html>

REAL SOLUTIONS for REAL NETWORKS



THE NEW IT

ADMIN – your source for technical solutions to real-world problems.

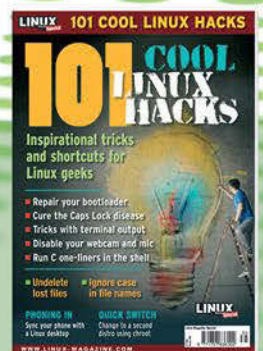
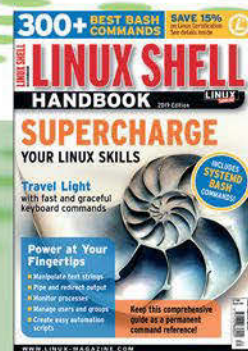
SUBSCRIBE NOW!

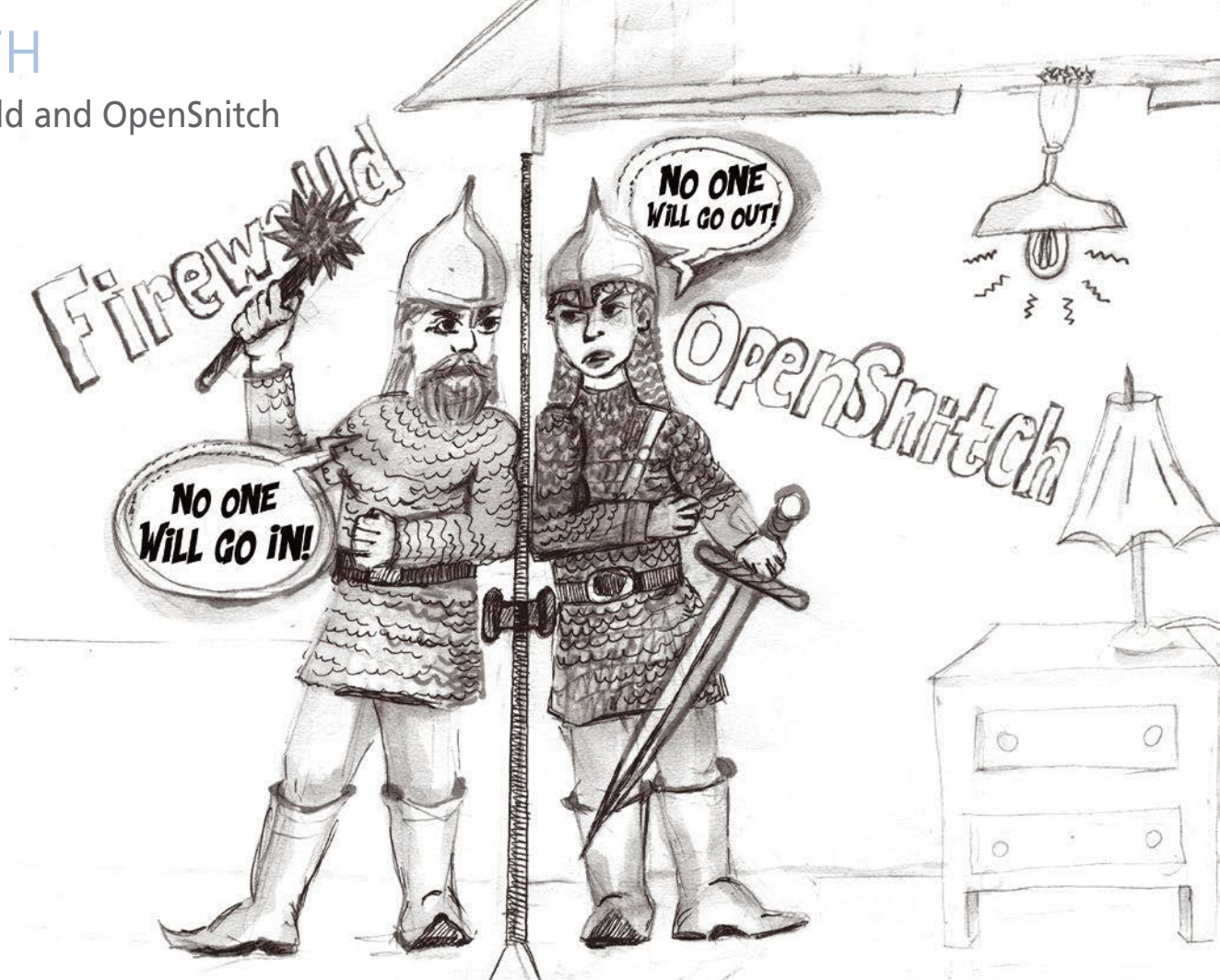
shop.linuxnewmedia.com

GET IT FAST with a digital subscription!



Check out our full catalog:
shop.linuxnewmedia.com





Workshop: Lock down your Linux system with Firewalld and OpenSnitch

DOUBLE PROTECTION

For maximum security, you'd better watch traffic in both directions. This hands-on workshop takes you through the steps of setting up firewalls for outgoing as well as incoming traffic. *By Alexander Tolstoy*

Linux has a great selection of firewalls for securing stand-alone computers or whole networks. Small LANs usually depend on small routers for security. But what if your Linux machine is exposed on the network and you don't have control over the upstream router or gateway? What if you're working behind one of the millions of smaller routers with out-of-date firmware or missing security patches?

If you're really serious about building a perfectly secure, network-attached Linux workstation, you'll need to fortify the system from two directions: with a host-based firewall for managing incoming connections and with a per-application firewall for controlling outbound traffic. This

hands-on workshop will show you how to fortify your system using Firewalld and the OpenSnitch application firewall.

Limiting the Incoming Flow

Network security starts with limiting access to your host from the outer world. One of the most advanced and popular tools for limiting incoming traffic is Firewalld [1], a modern and flexible firewall that replaced the legacy Iptables software a few years ago. Firewalld was introduced as a default network security solution in Fedora 18 and RHEL 7, and consequently, in CentOS 7 as well. The Firewalld firewall is now included in many other mainstream Linux distributions, such as openSUSE and Arch.

Firewalld has many benefits over Iptables and other old-fashioned tools for accessing the *netfilter* feature of the Linux kernel. Previous firewall tools required manual and precise rules for blocking or allowing certain ports and network addresses, which were not trivial and caused a lot of headache once the host's network configuration changed. Firewalld is more flexible, allowing you to add system services to zones and assign a zone to a network interface. A user does not need to know which port is associated with a service because Firewalld comes with over 140 presets for many, if not all, Linux services.

By assigning services to zones, users can create a custom set of rules for vari-

ous use cases and conditions. By default, Firewalld has 9 zones that differ by the level of trust to a network. Several interfaces can be assigned to a zone, but only one zone can be assigned to an interface. Firewalld applies all changes on the fly, so you won't need to restart when changing zones, services, or ports. Although Firewalld has no mechanism of auto-changing zones, you can use the zone system to adapt to changing network conditions. For instance, a laptop connected to a wired Ethernet network would probably use the `en01` interface, and a public Wi-Fi interface might use `wlan0`. You could assign each of these interfaces to different zones: `en01` could be in the `work` zone, which includes SSH access, and `wlan0` will belong to the `public` zone, which does not.

Firewalld (Figure 1) has a couple of graphical front-ends, specifically a GTK3-based GUI, which you will find in Gnome Software, and a Qt5-based interface bundled with Yast in openSUSE and SuSE Linux Enterprise. For this article, I will use the command-line utility known as `firewall-cmd`, which works everywhere (Figure 2).

Check to see if the firewall is running:

```
$ sudo firewall-cmd --state
```

If it is not, enable and launch it with:

```
$ sudo systemctl enable firewalld
&& sudo systemctl start firewalld
```

Say you want to open TCP port 80 to allow access to your network's web server from remote hosts:

```
$ sudo firewall-cmd
--add-port=80/tcp
```

If no zone is specified, the preceding command applies to the current zone, which should be `public` according to the Firewalld defaults. If you are unsure, you can check the default zone with:

```
$ sudo firewall-cmd
--get-default-zone
```

By default, all network interfaces are assigned to the default zone. You can find out what zones your current network interface has been assigned with the following command:

```
$ sudo firewall-cmd
--get-active-zones
```

Assume the current zone is `public`. The following command would allow you to view the zone configuration:

```
$ sudo firewall-cmd
--zone=public --list-all
```

The output will probably list `dhcpv6-client` as the only allowed service. Most of the other parameters for `public` are empty by default, which means that nothing else is allowed. Not a good

zone for further experiments in our sandbox. To transition the current network interface to another zone, enter:

```
$ sudo firewall-cmd --zone=trusted
--change-interface=en01
```

Note that changing zones changes the set of services that will be allowed. To tailor the zone for the needs of your network, you might wish to add services to the zone. For a list of available services, enter:

```
$ sudo firewall-cmd --get-service
```

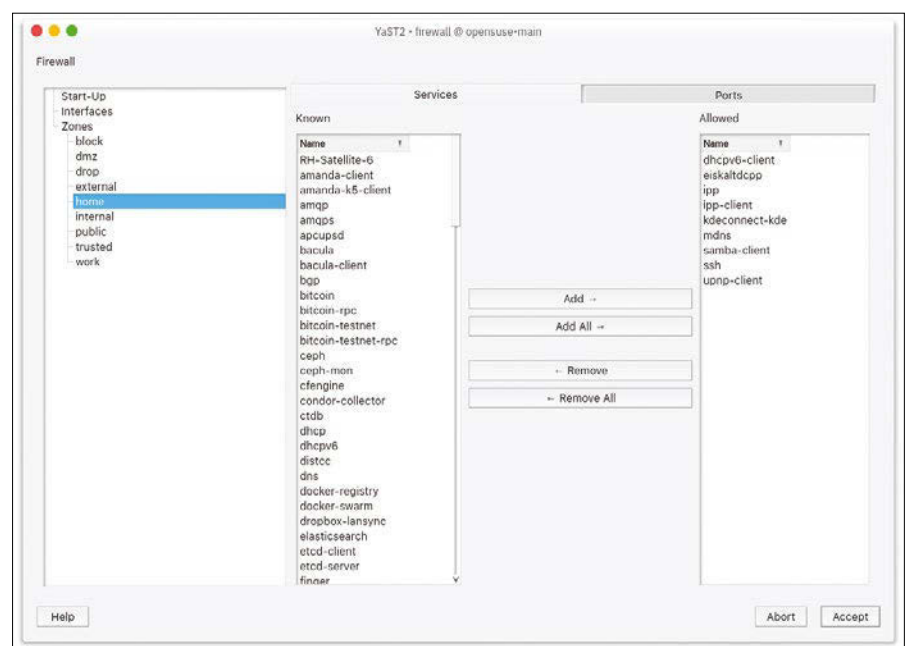


Figure 1: By default, Firewalld is very distrustful, but you can whitelist services that you want to allow.

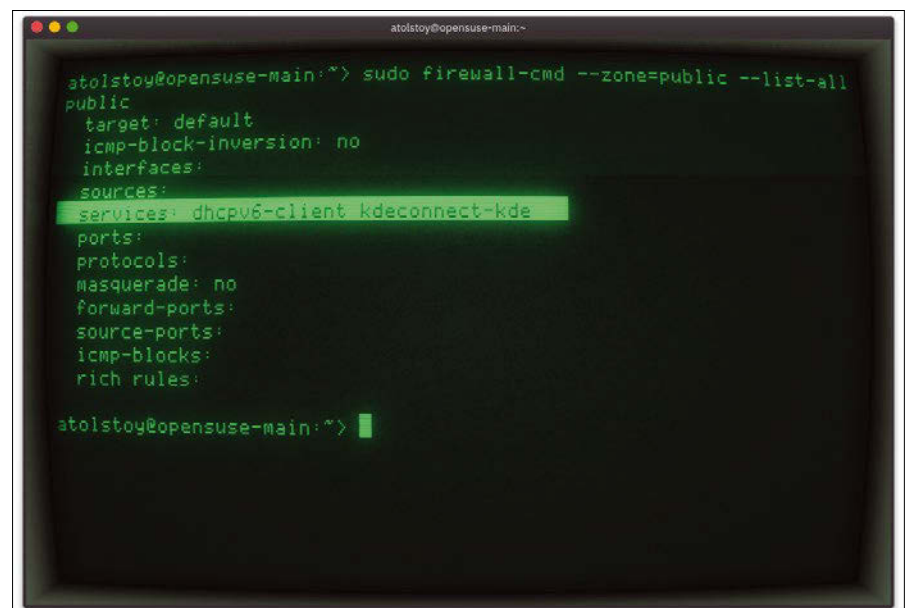


Figure 2: Firewall-cmd is an easy and powerful command-line tool.

You can also examine the contents of `usr/lib/firewalld/services` for a list of available services. Suppose I want to add the IPP service to the home zone for enabling remote printing to the CUPS server via the Internet Printing Protocol:

```
$ sudo firewall-cmd \
--zone=home --add-service=ipp
```

Other use cases require accessing your home PC from the outer world for the sake of remote administration, granting access to BitTorrent downloads, and other common scenarios.

Masquerading

On small home LANs, a home router usually performs network address translation, but it is also possible to set up a Linux computer as a router and then configure Firewalld to perform port forwarding and masquerading for incoming connections.

A router would need two network interfaces: one for the Internet (`eno0`) and another for the internal LAN (`eno1`).

To assign different zones to the two interfaces, enter:

```
$ sudo nmcli c mod eno0 \
connection.zone internet
$ sudo nmcli c mod eno1 \
connection.zone lan
$ sudo firewall-cmd \
--get-active-zones
```

Now that the two connections belong to different Firewalld zones, you can start masquerading. The following commands enable the `masquerade` feature for the internet zone and then forward all remote SSH connections from port 22 to port 2222 of another machine (192.168.88.11) inside the internal LAN:

```
$ sudo firewall-cmd \
--zone=internet --add-masquerade
$ sudo firewall-cmd \
--zone=internet --add-forward-port= \
port=22:proto=tcp:toport= \
2222:toaddr=192.168.88.11
```

So far all the changes to the Firewalld configuration are temporary and will be lost after the Firewalld service is restarted. To make the changes permanent, add the `--permanent` parameter to every `firewall-cmd` command that changes something. It is also possible to save the current setup by converting it to the permanent configuration:

```
$ sudo firewall-cmd \
--runtime-to-permanent
```

A reverse action is sometimes also required: changes made with the `--permanent` parameter are saved but not applied immediately (which is not obvious). To make permanent changes work for the current session, use the command:

```
$ sudo firewall-cmd --reload
```

Firewalld also has some dedicated features for handling outgoing connections (use `--direct` and `--add-rule` options), but another tool called OpenSnitch offers a more user-friendly solution for managing outgoing traffic.

Controlling Outgoing Traffic

For security and privacy reasons, many users wish to know what outgoing connections the applications you use and websites you visit want to establish from your machine. Various apps and APIs might try to launch outgoing connections from your computer, including

cloud storage clients, weather widgets, and big data collectors.

OpenSnitch [2] is designed to track all outgoing network connections. It is an open source project inspired by Little Snitch, an application firewall for Macs. The value of OpenSnitch is that it reveals all such background network activity and lets you explicitly allow or block each connection (Figure 3).

OpenSnitch is a work in progress, which is why it is not yet packaged for easy installation in most Linux distributions. Despite that, the software is stable enough and fits well into typical home or office usage scenarios. The steps for installing OpenSnitch are the same for any Linux flavor. If you're working in a recent Ubuntu version, the commands would look like Listing 1.

You will probably need to comment out the first line in the `ui/requirements.txt` file as long as the plain `grpcio` component (not the one with `-tools`) is only needed for Python2 and not for the newer Python3. Then complete the installation with:

```
$ make && sudo make install
```

OpenSnitch provides a Systemd service that you will need to activate upon startup:

```
$ sudo systemctl enable \
opensnitchd && \
sudo systemctl start opensnitchd
```

Fire up the OpenSnitch graphical front-end with:

```
$ opensnitch-ui
```

A new icon should appear in the system tray area. After any process initiates a

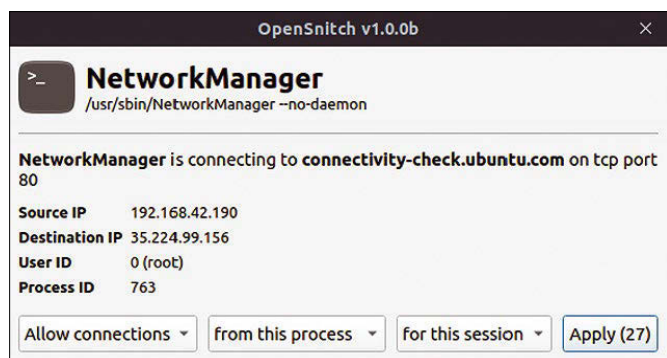


Figure 3: OpenSnitch reveals background network activity and lets you allow or deny each connection.

Listing 1: Setting Up OpenSnitch

```
$ mkdir -p ~/go/bin && export GOPATH=~/go
$ export PATH=$PATH:$GOPATH/bin
$ sudo apt-get install git libnetfilter-queue-dev \
libpcap-dev protobuf-compiler python3-pip golang-go
$ go get github.com/golang/protobuf/protoc-gen-go
$ go get -u github.com/golang/dep/cmd/dep
$ $GOPATH/src/github.com/golang/dep/install.sh
$ python3 -m pip install --user grpcio-tools
$ go get github.com/evilsocket/opensnitch
$ cd $GOPATH/src/github.com/evilsocket/opensnitch
```

connection to a TCP or UDP port, OpenSnitch will throw up a warning window. You will have 15 seconds to review the source and target IP addresses, the process name, port ID, and user ID and make a decision whether you allow or deny the connection. By default, if time runs out and you do nothing, OpenSnitch will allow the connection. Of course, in many situations, you simply need more time to decide what to do with an OpenSnitch warning. Extend the time to any other length by editing the `~/.opensnitch/ui-config.json` file.

Although OpenSnitch may look a bit paranoid at first, remember that it was designed to distrust everything. Frequent warnings indicate that OpenSnitch is in the training mode and it's up to you to decide if you will allow or block each connection once, for the current session, or forever.

Using the Firefox web browser for daily surfing reveals that the `systemd-resolved` service consumes UDP port 53 for DNS name resolution, Mozilla browser integration uses Python3.x to communicate with `extensions.gnome.org` on TCP 443, Ubuntu's version of NetworkManager runs a custom connectivity checker using TCP 80 and that's almost it – nothing really bad or suspicious. If you allow these typical connections from

the very beginning, you will for the most part be able to surf the web without further annoyances.

OpenSnitch will turn up a lot more when you are dealing with desktop integration APIs, weather services, online accounts, remote cloud connectors, and other services. If you run Skype, Dropbox, Google Drive, Windows apps via Wine, or any proprietary software, it always makes sense to keep OpenSnitch up and running. Even if the extended time period is not enough and you prefer to apply OpenSnitch actions once or for this session only, it is easy to trigger the alarm again:

```
$ sudo systemctl restart \
opensnitchd
```

The restart only applies to the per-application firewall and does not interfere with other network processes.

Saving your response as *forever* creates an appropriate JSON file under `/etc/opensnitchd/rules`. Just as with Iptables rules, you can modify OpenSnitch rules and even add your own.

OpenSnitch will help you understand how various applications and system services use the network. Many of these services are perfectly normal; the key is to look for rare or unusual or

unfamiliar services that might be behaving strangely. Nevertheless, the offending service does not have to be that exotic – you'll find examples of misbehaving Gnome Shell extensions and KDE Plasmoids that somehow want to connect to remote hosts, and you definitely have the right to know what is happening.

OpenSnitch is not just safeguarding software; it diligently collects statistics and keeps track of past incidents. Right-click on the application's icon in the tray and select *Statistics*. A new window will appear with several tabs for viewing network connection statistics (Figure 4). By default, statistics display on the *General* tab as a table with timestamps, processes, destinations, protocols, and actions. This tab, along with others (Hosts, Processes, Addresses, Ports and Users), has a small button near the top-right corner of the window that exports the current view to CSV format for further usage. With the help of *Statistics*, it is possible to analyze past events that you may not have noticed at the time they occurred.

Conclusions

OpenSnitch may not be a complete clone of Little Snitch – it misses a lot of nice details, like placing connections over an eye-catching geographic map, but it does its job faithfully.

Setting up Firewalld and OpenSnitch for controlling incoming and outgoing connections is an uncomplicated task compared to the task of configuring old-school firewalls. Both applications have well-designed graphical front-ends and let you escape the command line completely if you prefer to point and click. ■■■

Info

[1] Official Firewalld website:
<https://firewalld.org>

[2] OpenSnitch GitHub page: <https://github.com/evilsocket/opensnitch>

Author

Alexander Tolstoy is a long-term Linux enthusiast and a tech journalist. He never stops exploring hot new open source picks and loves writing reviews, tutorials, and various tips and tricks. Sometimes he faces a bitter truth delivered by the inhuman fortune | cowsay command that he once thoughtlessly put in `~/.bashrc...`

OpenSnitch Network Statistics						
General						
Version	Status	Uptime	Rules	Connections	Dropped	
1.0.0b	running	0:11:57	3	202	0	
Time	Action	Process	Destination	Protocol		
2019-12-23 18:15:46	allow	/usr/sbin/NetworkManager	connectivity-check.ubuntu.com:80	tcp		
2019-12-23 18:15:15	allow	/lib/systemd/systemd-resolved	192.168.42.129:53	udp		
2019-12-23 18:10:46	allow	/lib/systemd/systemd-resolved	192.168.42.129:53	udp		
2019-12-23 18:10:46	allow	/lib/systemd/systemd-resolved	192.168.42.129:53	udp		
2019-12-23 18:10:46	allow	/usr/sbin/NetworkManager	connectivity-check.ubuntu.com:80	tcp		
2019-12-23 18:09:27	allow	/lib/systemd/systemd-resolved	192.168.42.129:53	udp		
2019-12-23 18:09:27	allow	/lib/systemd/systemd-resolved	192.168.42.129:53	udp		
2019-12-23 18:09:27	allow	/lib/systemd/systemd-resolved	192.168.42.129:53	udp		
2019-12-23 18:09:27	allow	/lib/systemd/systemd-resolved	192.168.42.129:53	udp		
2019-12-23 18:09:26	allow	/lib/systemd/systemd-resolved	192.168.42.129:53	udp		
2019-12-23 18:09:25	allow	/usr/lib/firefox/firefox	secure-us.imrworldwide.com:443	tcp		
2019-12-23 18:09:25	allow	/lib/systemd/systemd-resolved	192.168.42.129:53	udp		
2019-12-23 18:09:25	allow	/lib/systemd/systemd-resolved	192.168.42.129:53	udp		
2019-12-23 18:09:25	allow	/usr/lib/firefox/firefox	ca3.chci.com:443	tcp		

Figure 4: Statistics are everything. See the whole outgoing network activity log within a single tab.

Solving the river crossing puzzle with Go

Safe Passage



How does a ferryman transport a wolf, a goat, and a cabbage across a river in a boat that can only carry the ferryman plus one item at a time, without the wolf eating the goat or the goat eating the cabbage while left together back on shore?

Mike programs the solution in Go. *By Mike Schilli*

Whether it is a wolf, a goat, and a cabbage, or one of the many variations on this scenario, keeping the participants safe during passage is part of the classic river crossing logic puzzle [1]. Humans find a solution to these brainteasers by trial and error. Computers, on the other hand, determine the solution by exploring trees of interlinked state sequences, eventually arriving at the desired final state by following one of many different available paths [2].

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.



Before crossing the river, the ferryman, the wolf, the goat, and the cabbage are initially on the west bank. The ferryman can only take one of the three candidates into the boat and cross over to the east bank. During the crossing, he has to take care not to leave the wolf and goat alone on the bank, because the wolf would eat the goat. The same applies to the goat and the cabbage, as goats like to eat cabbage.

The riddle seems unsolvable at first. For example, once the ferryman has

crossed the river with the goat and returns on an empty run, the only choice he has is between the wolf and the cabbage as the next candidate. However, neither will get along with the goat on the other bank. The trick is that the ferryman can also carry passengers on the return journey, thus ensuring that he never leaves a dangerous combination (goat/cabbage or wolf/goat) unattended.

Ferry Algorithm

In order to solve this teaser with an algorithm, the program has to model the problem as a sequence of states that assign the candidates to either the west or the east bank. In the initial state, the ferryman, wolf, goat, and cabbage are all sitting on the west bank, and the east bank is empty. In a possible second state, the goat sits on the east bank, while the wolf, cabbage, and ferryman are on the west bank after the ferryman has completed his return journey.

Each state maps the two banks, with each bank accommodating a number of candidates. From a given state, the system can switch to one or more subse-

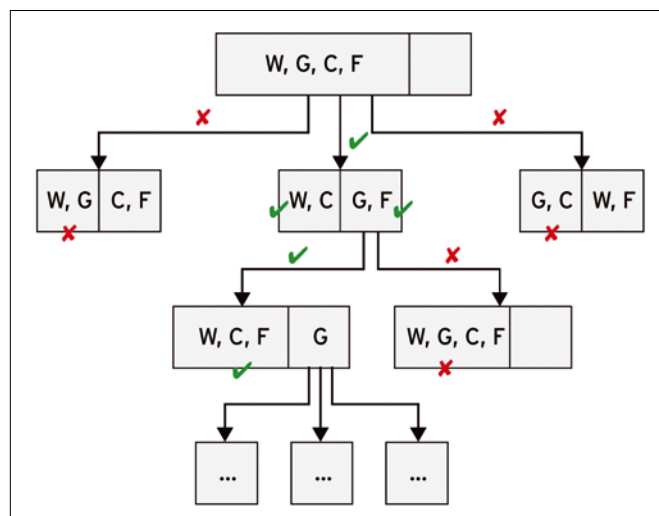


Figure 1: The algorithm moves from the initial state (on top) through all possible subsequent states and discards impermissible ones.

Listing 1: Program Start

```
01 $ go build solver.go passenger.go sort.go sort.go shore.go state.go
02 $ ./solver
```

Listing 2: passenger.go

```
01 package main
02
03 type Passenger int
04
05 const (
06     Goat Passenger = iota
07     Wolf
08     Cabbage
09     Ferryman
10 )
11
12 func (p Passenger) String() string {
13     return []string{"Goat", "Wolf",
14         "Cabbage", "Ferryman"}[p]
15 }
```

quent states, some of which are not allowed, for example, because the wolf and goat would be left unattended on the same bank.

Starting from the initial state, the algorithm tries out all possible subsequent states to determine whether they are permissible. If a state checks out, the algorithm searches for further subsequent states on this basis, and so the merry dance continues. If the program at some point finds itself in a state in which all the candidates are sitting on the east bank, the puzzle is solved. The program then outputs the recorded path that led to this state, and the astounded user can now marvel at a solution to the problem.

Depth or Breadth First

In the diagram in Figure 1, the algorithm starts with the state at the top, where all participants (W = wolf, G = goat, C = cabbage, F = ferryman) are located in the left part of the rectangle (i.e., on the west

bank). If the algorithm then moves to the next node bottom left, it finds a state where the cabbage and ferryman are on the east bank. On the left bank, you have the wolf and the goat – but this is impermissible according to the rules, so the algorithm discards this state (shown with a red X) and naturally doesn't progress to any subsequent states from this dead end.

There are two strategies for traversing the states in this tree structure: breadth first and depth first. Breadth first discovers all direct subsequent states of a state before it drills deeper into their successors. Depth-first, on the other hand, drills as deep as it can, visiting children and grandchildren, before it visits siblings at the same level.

In this river crossing puzzle, the goal is to find a solution with as few steps as possible, so breadth first is the right approach, since depth first might find a much too complicated solution.

Once the algorithm is determined, the programming can begin. For the sake of clarity, the code is divided into several listings [3]. If you want to start right away, launch the problem solver with the instructions in Listing 1, which compiles

the code without any additional external libraries (i.e., with plain vanilla Go).

Listing 2 models the passengers and the ferryman using integer values defined by `const`. The `=iota` assignment tells the compiler to assign the integer values `0`, `1`, ... to the constants `Goat`, `Wolf`, and so on. In order to be able to convert them back into strings for the user's entertainment, the `String()` method on the `Passenger` type defines a slice of strings. Given one of the previously defined constants, the function reaches into the slice to pull out the corresponding string element, because elements in Go's string slices are handily indexed from `0` onward, just like the `=iota`-defined integer constants. In general, a `String()` method in Go is used to convert self-defined types into strings as soon as they are found in a string context – for example, if `Printf()` wants to output types as strings with `"%s"`.

Next, Listing 3 defines `Shore` as a data structure to represent a river bank including its methods. The `passengers` array stores the waiting passengers within the `Shore` structure starting in line 8. All elements are of the `Passenger` type – that is, they are represented as named integer constants defined in Listing 2 (e.g., `"Wolf"`). Both Listing 2 and Listing 3 define the same `main` package; therefore they can both reference each other's data structures.

Listing 3: shore.go

```
01 package main
02
03 import (
04     "fmt"
05     "sort"
06 )
07
08 type Shore struct {
09     passengers []Passenger
10 }
11
12 func (s Shore) String() string {
13     sort.Sort(Passengers(s.passengers))
14     return fmt.Sprintf("%s", s.passengers)
15 }
16
17 func (s Shore) Copy() Shore {
18     c := Shore{}
19     c.passengers = make([]Passenger,
20         len(s.passengers))
21     copy(c.passengers, s.passengers)
22     return c
23 }
24
25 func (s *Shore) Add(p Passenger) {
26     s.passengers = append(s.passengers, p)
27 }
28
29 func (s *Shore) Remove(p Passenger) {
30     result := []Passenger{}
31     for _, passenger := range s.passengers {
32         if passenger != p {
33             result = append(result, passenger)
34         }
35     }
36     s.passengers = result
37 }
38
39 func (s *Shore) Move(p Passenger,
40     t *Shore) {
```

Listing 3: shore.go (continued)

```

41 s.Remove(p)
42 t.Add(p)
43 }
44
45 func (s Shore) Has(p Passenger) bool {
46     for _, passenger := range s.passengers {
47         if passenger == p {
48             return true
49         }
50     }
51     return false
52 }
53
54 func (s Shore) IsValid() bool {
55     if s.Has(Ferryman) {
56         return true
57     }
58     if s.Has(Wolf) && s.Has(Goat) {
59         return false
60     }
61     if s.Has(Goat) && s.Has(Cabbage) {
62         return false
63     }
64     return true
65 }

```

Deep Copies

Go won't automatically copy the deeper parts of a nested data structure. To create a deep copy, the programmer has to manually define the necessary steps, and, for example, explicitly copy arrays that reside within the structure. The `Copy()` method starting in line 17 of Listing 3 acts on a given `Shore` object and first initializes a new, empty object. It then uses `make` to allocate a passenger array of the same length in the copy, and then uses the built-in `copy()` function to copy the elements from the original to the copy, after which it returns the latter to the caller. With this `Shore` helper function, the caller can easily duplicate game states and build new ones from existing ones by inflicting gradual changes.

To assign new passengers to a particular `Shore` object, to remove them again, to move them to a new shore side, or to query whether a passenger is stationed on a particular shore, Listing 3 provides the methods `Add()`, `Remove()`, `Move()`, and `Has()` in lines 25 to 52. For example, a state object can later select one of its shore sides and use `west.Has(Wolf)` to

retrieve a Boolean response (`true/false`) to the question as to whether the wolf is present.

The implementation shows a disadvantage of choosing an array as the data structure for storing passengers. To see if a certain passenger is present, a `for` loop has to run through all elements until it either finds them or returns `false` at the end if the search is unsuccessful. Removing a passenger actually tears a hole in the array, which a `for` loop then has to laboriously patch again by rebuilding the array without the removed passenger as shown in line 31. Alternatively, a map would provide faster access, but it has the disadvantage that the data stored in it is unsorted and has to be sorted later for display purposes.

The `IsValid()` method starting in line 54 checks whether the passengers pres-

ent on a shore side are in compliance with the game rules or contradict them in any way. It more or less implements the algorithm's business logic. If the ferryman is present on one side, this is a valid state, because even wolf and goat get along just fine as long as there is a supervisor present. But if the ferryman is missing in this case, `IsValid()` in line 59 returns `false`, because the wolf would devour the goat without further ado. The same applies to the goat and the cabbage, when the method returns `false` if they're residing on a bank together without the ferryman present. On the other hand, if everything is okay, `IsValid()` returns `true` at then end.

Sorting with Go

To enable the algorithm to easily determine later whether two bank objects have identical passengers (to detect already-seen states), it compares their string representations generated by `String()` in line 12,

character by character. Since the original order of the arrays is lost after manipulating game states, line 13 sorts them according to passenger numbers. The `Passenger` elements are in fact plain vanilla integers, but the strict type-enforcing Go compiler refuses to sort them with the standard integer sorting function, `sort.Ints()`. This is why Listing 4 steps in to teach Go how to sort an array of `Passenger` values. It defines a `Passengers` (note the plural) type plus three functions that Go requires the programmer to define in order to sort arbitrary data types.

The first function is named `Len()` and specifies the length of the array to be sorted. Listing 4 simply determines this using Go's built-in `len()` function. The second function is `Swap()` and shows the compiler how to swap two elements at the index positions `i` and `j` during sorting. This is easily done in Go, because it understands the syntax `a, b = b, a`. Thirdly, the sort needs a `Less()` function, which returns a true value if its first parameter is smaller than the second. With the present array of integers, a numerical comparison of the two elements using `p[i] < p[j]` gives the correct results.

With this tool in Listing 4, Listing 3 can now call `sort.Sort(Passengers(...))` on an array of `Passenger` types, which the function then sorts in-place [4].

From State to State

Listing 5 defines the game states (i.e., the situation on both banks), with the `State` data type. Go does not support classes, but it offers object orientation with structures and methods operating on the data defined in the structure.

Listing 4: sort.go

```

01 package main
02
03 type Passengers []Passenger
04
05 func (p Passengers) Len() int {
06     return len(p)
07 }
08 func (p Passengers) Swap(i, j int) {
09     p[i], p[j] = p[j], p[i]
10 }
11 func (p Passengers) Less(i, j int) bool {
12     return p[i] < p[j]
13 }

```

It is important to note the difference between writing and reading methods. A normal “receiver” (the object preceding the function name after the `func` keyword, as in the `IsValid()` function in line 13) gives the function read-only access to the data. The function can write, but only into a copy of the object, which disappears after the function has terminated.

For persistent writes, you need to use a pointer receiver, as shown, for example, in `Move()` from line 25. No matter whether `Move()` is a normal receiver or a pointer (as indicated by a `*` preceding the receiver in the function definition), the invocation of `state.Move()` by the caller looks completely identical. This means that programmers have to be very careful. If the pointer definition is missing by mistake, no errors occur – but nothing is written correctly, which (and this has been scientifically proven) leads to much wailing and gnashing of teeth on the part of the programmer.

Every state object holds two shore objects `west` and `east`, as well as a pointer `prev` to the state from which it was derived. If the algorithm later stumbles upon a solved state, the path to the target can be traced backwards using this pointer chain. The ferryman’s location determines on which shore the next change can happen. Line 26 defines the direction as “from west to east,” but reverses `from` and `to` in line 29, if it finds that the ferryman isn’t on the west bank, but on the east bank instead.

Bosses and Workers

A state is considered permissible if confirmed by `IsValid()` from line 13 in

Listing 5: `state.go`

```
01 package main
02
03 import (
04     "fmt"
05 )
06
07 type State struct {
08     west Shore
09     east Shore
10     prev *State
11 }
12
13 func (s State) IsValid() bool {
14     return s.west.IsValid() &&
15         s.east.IsValid()
16 }
17
18 func (s State) String() string {
19     return fmt.Sprintf("%s | %s",
20         s.west.String(),
21         s.east.String(),
22     )
23 }
24
25 func (s *State) Move(p Passenger) {
26     from := &s.west
27     to := &s.east
28     if to.Has(Ferryman) {
29         from, to = to, from
30     }
31     from.Move(p, to)
32 }
33
34 func (s State) IsFinished() bool {
35     return len(s.west.passengers) == 0
36 }
37
38 func (s State) Copy() State {
39     d := State{}
40     d.west = s.west.Copy()
41     d.east = s.east.Copy()
42     return d
43 }
44
45 func (s State) Successors() []State {
46     startShore := s.east
47     if s.west.Has(Ferryman) {
48         startShore = s.west
49     }
50     results := []State{}
51
52     for _, passenger :=
53         range startShore.passengers {
54         candidate := s.Copy()
55         candidate.Move(passenger)
56         if passenger != Ferryman {
57             candidate.Move(Ferryman)
58         }
59
60         if candidate.IsValid() {
61             results = append(results, candidate)
62         }
63     }
64
65     return results
66 }
```

Listing 5. It examines both of its shore objects, `west` and `east`, and returns an okay if both their `IsValid()` functions say so. The string representation of the `State` object is calculated by the object’s `String()` method from line 18 onwards. In turn, it delegates the hard work to the two bank objects, and then, acting as a higher-level authority, combines the two partial results into a string with a pipe sign as the separator. A copy of a state is created by `Copy()` in line 38. The function makes copies of the two shores and again delegates the hard work to the individual shore objects.

The final game state is reached when `IsFinished()` returns a true value starting in line 34. To determine this state, the method simply checks whether there is any candidate left on the west

bank. If that’s true and there was no cheating, then the ferryman and all passengers should all be safe and sound on the east side.

A state object’s most important task is to find subsequent states so that the solver can later build the search tree. The `Successors()` method in line 45 tries to send the ferryman to the other bank either alone or with a passenger waiting on the same bank. It blindly tries all possibilities and then checks with `IsValid()` in line 60 to see whether a valid state has been created this way. If so, it appends it to the result list that the method returns to the calling solver.

Solution

All the solver in Listing 6 now needs to do is create the start state in `main()`, as-

sign the ferryman and all passengers to the west bank of the river, and call `solve()` in line 8. There, a dynamically

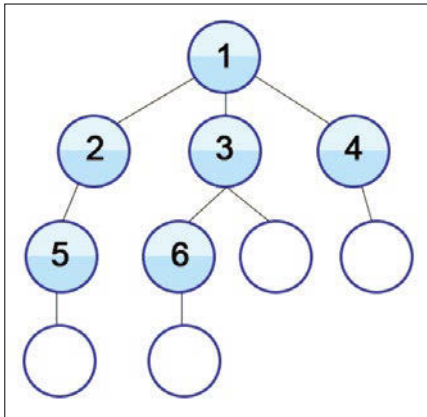


Figure 2: A breadth-first search examines the tree level by level and finds the shortest path to the solution. MRE, CC BY-SA 3.0

maintained `todo` list in the form of an array of state objects determines states that still need to be examined. If one of them turns out to be the target state, `IsFinished()` in line 25 returns a true value. If that's the case, the `for` loop starting in line 27 then compiles the path to this end state by tracing back the states' `prev` pointers. They guide the loop all the way to the start state, which has an uninitialized `prev` pointer (i.e., it has a value of `nil` in line with the Go standard). Line 32 builds an array from this inverse sequence using `append()` in each step to insert new elements at the beginning of the existing array, resulting in a new array.

Initially, the `todo` list only contains the start state with all participants on the west bank. The `for` loop from line 13 uses slice arithmetic to remove the last

element of the `todo` array in line 17 on every round. At the end of the loop, `Successors()` searches for possible successors of the current state and inserts them at the beginning of the `todo` slice. This creates a queue where new elements are added from the left and picked from the right (first in, last out).

This procedure causes a breadth-first search [5] in the search tree (Figure 2). It scans the nodes layer by layer until it finds the target state. If we were to use a stack instead of a queue (first in, first out), for example, by appending new states to the right of the array and also fetching them from there, this would cause depth-first behavior in the tree. The algorithm would not proceed layer by layer, but would always advance immediately to the most deeply nested paths. It might find a solution faster –

Listing 6: solver.go

```

01 package main
02
03 import (
04     "errors"
05     "fmt"
06 )
07
08 func solve(state State) ([]State, error) {
09
10     seen := make(map[string]bool)
11     todo := []State{state}
12
13     for len(todo) > 0 {
14         // pop off last element
15         lastidx := len(todo) - 1
16         s := todo[lastidx]
17         todo = todo[:lastidx]
18
19         // prevent cycles
20         if _, ok := seen[s.String()]; ok {
21             continue
22         }
23         seen[s.String()] = true
24
25         if s.IsFinished() {
26             path := []State{}
27             for cs := &s; cs.prev != nil;
28                 cs = cs.prev {
29                 c := cs.Copy()
30                 path = append([]State{c}, path...)
31             }
32             path = append([]State{state}, path...)
33             return path, nil
34         }
35
36         for _, succ := range s.Successors() {
37             // insert new element at end
38             succ.prev = &s
39             todo = append([]State{succ}, todo...)
40         }
41     }
42
43     return []State{},
44         errors.New("Can't solve")
45 }
46
47 func main() {
48     var state State
49
50     state.west.Add(Wolf)
51     state.west.Add(Cabbage)
52     state.west.Add(Ferryman)
53     state.west.Add(Goat)
54
55     solution, err := solve(state)
56     if err != nil {
57         panic(err)
58     }
59
60     fmt.Printf("Solution:\n")
61
62     for _, step := range solution {
63         fmt.Printf("[%s]\n", step)
64     }
65 }

```

but not necessarily the shortest option, and often a very convoluted one.

To prevent the solver from getting stuck and, for example, sending the ferryman back and forth without any cargo (therefore generating an infinite number of subsequent states in the process), line 20 checks whether a proposed subsequent state has previously been processed. To do this, it checks the seen map (a hash table) to discover whether the state's string representation already exists as a key. If so, the continue statement in line 21 ignores this entry and triggers the next round in the for loop.

Figure 3 shows the solver's output, which found a solution in seven moves. First, the ferryman takes the goat to the east bank, before coming back empty. The ferryman then takes the wolf to the east bank, but takes the goat back. When he reaches the west bank, he

```
$ ./solver
Solution:
[[Goat Wolf Cabbage Ferryman] | []]
[[Wolf Cabbage] | [Goat Ferryman]]
[[Wolf Cabbage Ferryman] | [Goat]]
[[Cabbage] | [Goat Wolf Ferryman]]
[[Goat Cabbage Ferryman] | [Wolf]]
[[Goat] | [Wolf Cabbage Ferryman]]
[[Goat Ferryman] | [Wolf Cabbage]]
[[] | [Goat Wolf Cabbage Ferryman]]
$
```

Figure 3: The algorithm solves the problem in seven moves.

grabs the cabbage and leaves the goat there. He takes the cabbage to the east bank, where the wolf is waiting but is no threat to the cabbage. Now the ferryman only has to cross back without cargo and then take the goat to the east bank. The whole crew has then arrived safely without any incidents.

Even More

Once programmed, the solver can also be expanded if required. How about an additional passenger who is rather mellow and won't threaten anyone, for example, a salmon? Quickly added to Listing 2's const construct as Salmon and wired into the main() function in Listing 6 with state.west.Add(Salmon), the solver will find a solution for this modified problem as well, although it will be a bit more complicated. Figure 4

```
$ ./solver-salmon
Solution:
[Goat,Wolf,Cabbage,Ferryman,Salmon | ]
[Wolf,Cabbage,Salmon | Goat,Ferryman]
[Wolf,Cabbage,Ferryman,Salmon | Goat]
[Wolf,Cabbage | Goat,Ferryman,Salmon]
[Wolf,Cabbage,Ferryman | Goat,Salmon]
[Wolf | Goat,Cabbage,Ferryman,Salmon]
[Goat,Wolf,Ferryman | Cabbage,Salmon]
[Goat | Wolf,Cabbage,Ferryman,Salmon]
[Goat,Ferryman | Wolf,Cabbage,Salmon]
[ | Goat,Wolf,Cabbage,Ferryman,Salmon]
$
```

Figure 4: After adding a salmon to the passenger list, you can still find a solution, although it is a little more complex.

shows the nine steps necessary for the entire crew to safely arrive on the eastern shore.

In this case, the ferryman first takes the goat, then the salmon, and then the cabbage. But now he cannot return without a cargo to the wolf waiting on the west bank, because the goat would eat the cabbage. Instead, he takes the goat back, crosses with the wolf, and then returns without cargo to pick up the goat on the west bank for the final crossing. A human puzzle fan would have tinkered with this for a while, but for the algorithm it's just another routine task, solved in a fraction of a second. ■■■

Info

- [1] River crossing puzzle: https://en.wikipedia.org/wiki/River_crossing_puzzle
- [2] "Classic Computer Science Problems in Python": <https://www.manning.com/books/classic-computer-science-problems-in-python>
- [3] All listings for this article: <ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/232/>
- [4] In-place algorithms https://en.wikipedia.org/wiki/In-place_algorithm
- [5] Breadth-first search: https://en.wikipedia.org/wiki/Breadth-first_search



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



The sys admin's daily grind: lshw

Make Room!

In order to avoid complaints from his children, Charly prefers to use lshw instead of a screwdriver to analyze his home firewall PC's hardware details. *By Charly Kühnast*

For many years, a small, fanless industrial PC with two Gigabit Ethernet interfaces has served as a firewall in my home store-room. The hardware is not very powerful, but it is just about sufficient for forwarding network packets and a few iptables rules.

I seem to recall that this device has 2GB RAM – but whether in the form of a 2GB module or as two 1GB modules, I really don't know. How many RAM slots are there anyway, and what is the maximum amount of RAM I could use if I decided to upgrade? I would like to find that out without taking the firewall off the Internet and removing its case, because that always provokes disrespectful comments from my kids: "What kind of availability is that, Dad?"

Hardware Lister, lshw, [1] is a reliable tool for answering questions about hardware. It elicits extensive information from the system about every installed hardware component – usually more than I ever wanted to know.

In daily practice, I usually misuse lshw to find out whether I'm on a physical server or a virtual machine (VM). As always, many roads lead to Rome, but it's hard to get there faster than with lshw:

```
$ lshw -c system | grep -i product
product: VMware Virtual Platform
[...]
```

Ah, a VM. Sometimes you see KVM or VirtualBox, and so on, depending on the virtualization platform.

But back to my small firewall and the actual purpose of lshw. For example, typing

```
lshw -C cpu
```

gives you all the information that lshw finds about the installed CPU. In this

H/W path	Device	Class	Description
/0/0		memory	64KiB BIOS
/0/4/5		memory	24KiB L1 cache
/0/4/6		memory	512KiB L2 cache
/0/£		memory	2GiB System Memory
/0/£/0		memory	2GiB DIMM SDRAM Synchronous
/0/£/1		memory	DIMM [empty]
/0/£/2		memory	DIMM [empty]
/0/£/3		memory	DIMM [empty]

Figure 1: lshw's diagnosis is that there are four DIMM banks in Charly's firewall containing just one lonely 2GB module.

```
Maximum Memory Module Size: 4096 MB
Maximum Total Memory Size: 16384 MB
Installed Size: 2048 MB (Double-bank Connection)
Enabled Size: 2048 MB (Double-bank Connection)
Installed Size: Not Installed
Enabled Size: Not Installed
Installed Size: Not Installed
Enabled Size: Not Installed
Installed Size: Not Installed
Enabled Size: Not Installed
```

Figure 2: Potential to expand: Charly's firewall computer can take up to 16GB RAM.

case, it is an Intel Atom with a 32-bit core and two logical CPUs. My question about the RAM is also answered by lshw. The command

```
lshw -short -C memory
```

shows that there are four DIMM banks, of which only one is occupied by a 2GB module (Figure 1).

Despite the wealth of information, there is one little thing that lshw can't help me with: It doesn't tell me the maximum amount of RAM I can use. For this, I have to dig a little deeper into the toolbox and pull out dmidecode. The following command gives me the desired results:

```
$ dmidecode -t memory | grep size -i
```

With this, I discovered that the maximum expansion capacity is four DIMMs of 4GB each (Figure 2). Now, let me just check what I have available in spare RAM. ■■■

Info

[1] lshw: <https://ezix.org/project/wiki/HardwareLiSter>

Author

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



DrupalCon

MINNEAPOLIS 2020
MAY 18-22

Minneapolis Convention Center
events.drupal.org/minneapolis2020

We've announced our stellar lineup of curated sessions featuring top minds in open source digital experience.

Solidify your plans. Register **today** to start your amazing DrupalCon journey!

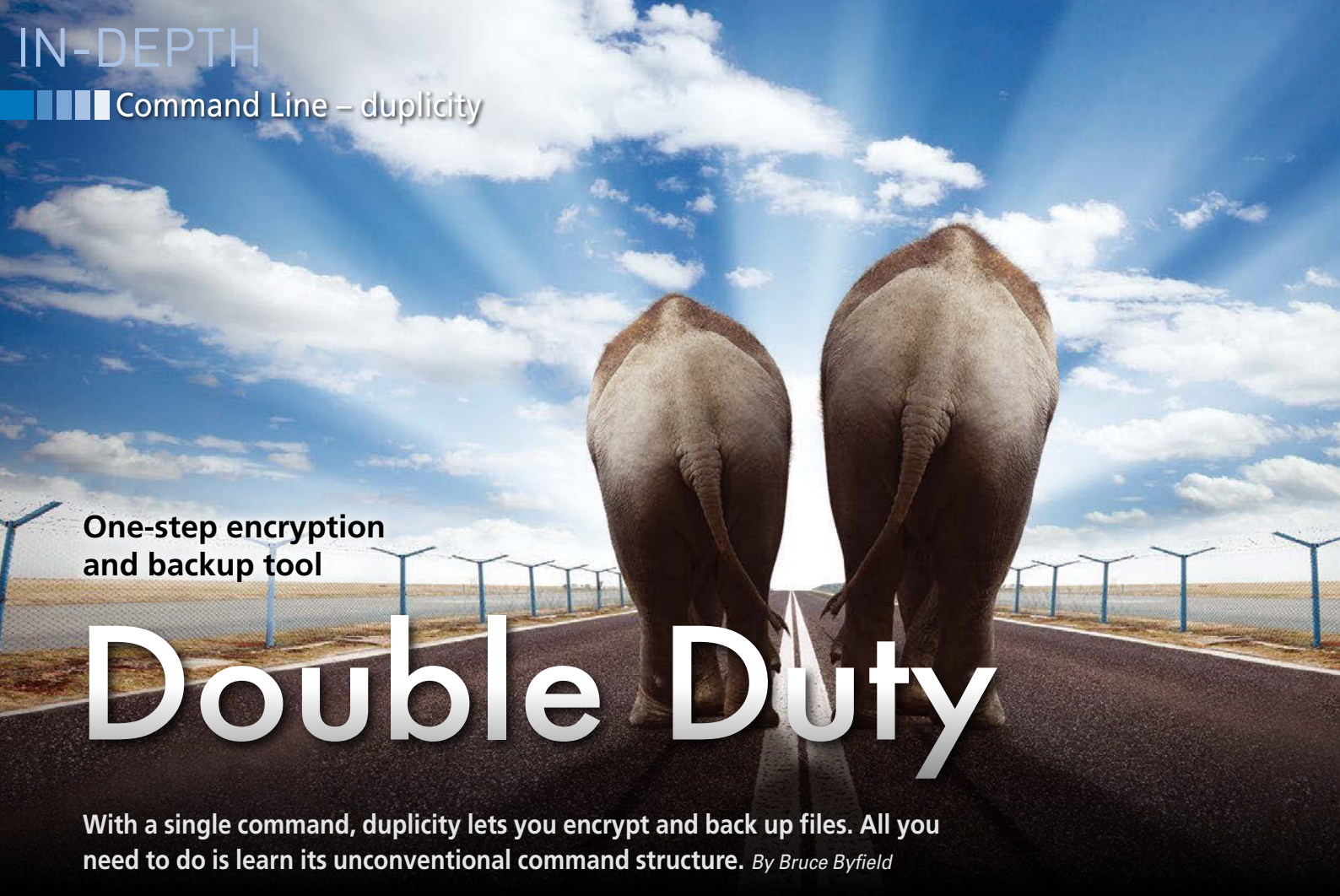
Network, learn and be inspired.



**Important
Dates**

Regular Registration Ends - March 31
Hotel Booking Deadline - April 7





One-step encryption and backup tool

Double Duty

With a single command, duplicity lets you encrypt and back up files. All you need to do is learn its unconventional command structure. *By Bruce Byfield*

Despite its name, duplicity [1] is not a command to enable dishonesty. Instead, duplicity is one of those modern command-line tools that combines more than one function in the same application. Instead of encrypting files in a separate operation and then backing them up, duplicity does both in a single step. When it comes to using duplicity, its only limita-

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 <http://prenticepieces.com/>.

tion is a somewhat eccentric command structure.

Using GnuPG for encryption, duplicity backs up directories and files on a local or remote server. Although sources to back up are expressed as directory paths, targets for the backup files must be listed as a URL, not a path. For example, a local target directory must be identified as `file:///usr/local/backup` rather than `/backup`. (Note that the three forward slashes in the target URL are not an error: Two are for the URL, and the third is for the path from root.) By default, each archive is placed in a separate directory unless you use the `--allow-source-mismatch` option.

duplicity supports backups to local drives (including mounted external drives), FTP, SFTP/SCP, Rsync, WebDAV,

Google Docs, HSI, and Amazon S3. duplicity's man page does not detail how to set up all these various targets, but detailed instructions and examples are available online, particularly for those that require additional libraries, such as Google Drive, which requires PyDrive [2], and Amazon S3, which requires *python-boto* [3]. Some targets also take unique options. Regardless of the targets, after the first creation of a backup, later backups will be incremental, affecting only parts of files that have changed since the last backup (Figure 1). Remember that directories containing a backup display in a file manager, but the backup archives do not since they are encrypted. You will need to use duplicity to list the backup archives (see the Actions section).

```
bb@nanday:~$ duplicity /home/bb/fonts/Cantarell file:///usr/local/home/bb/duplicity/cantarell
Local and Remote metadata are synchronized, no sync needed.
Last full backup date: none
GnuPG passphrase for decryption:
Retype passphrase for decryption to confirm: █
```

Figure 1: A local backup: duplicity detects that the source and target are connected, that this is the first backup, and defaults to a full backup. Since a GnuPG key is not defined as an environmental variable, duplicity then requests a GnuPG key. After the backup, a summary of statistics is given that includes the size of the backup files.

Lead Image © Jakub Gajda, 123RF.com


```

gpg (GnuPG) 2.2.12; Copyright (C) 2018 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
(1) RSA and RSA (default)
(2) DSA and Elgamal
(3) DSA (sign only)
(4) RSA (sign only)
Your selection? █

```

Figure 2: Before encrypting, `duplicity` requires a GnuPG key.

Backups created by `duplicity` will preserve original directories and files, including permissions and symbolic links, although not hard links. In addition, the man page warns against including `/proc`, the directory that displays system information in a virtual filesystem, because of the likelihood of causing a crash.

In order to encrypt, `duplicity` requires a GnuPG encryption key. If you do not already have a key, you can familiarize yourself with the basic concepts and the procedures for creating one by reading an online how-to [4]. When you are ready, you can generate a key with the command:

```
gpg --full-generate-key
```

You can use `--gen-key`, a less verbose option, if you are familiar with `gpg`, but `--full-generate-key` is easier for novices, since it gives more explanations, and, if nothing else, provides moderately secure defaults (Figure 2).

Actions

`duplicity` uses actions (aka sub-commands) to refine the basic command. You can do a basic backup with nothing more than the following structure:

```
duplicity SOURCE-PATH TARGET-URL
```

For example:

```

duplicity /home/tt 🔍
scp://TARGET-UID/backup

```

However, you have the option of adding an action after the basic command. For

example, when creating a backup, you can specify `full` rather than the default `incr` (incremental), even if nothing has changed in the backup. If you specifically request `incr`, `duplicity` will switch to `full` if anything goes wrong. If a previous backup attempt has failed or stopped before finishing, you should run `cleanup` before the next attempt to see if any files with errors are in the backup. If so, you will need to use the `--force` option to delete them.

To edit an archive, you can use the `remove-older-than TIME` action, specifying the time in a number of different formats, including: `YYYY/MM/DD`; an interval after which all files will be deleted, using `s`, `m`, `h`, `D`, `W`, `M`, or `Y` to set the interval; or else simply `now`. Similarly, `remove-all-but-n-full COUNT` indicates how many backup sets to keep – for example, a count of 1 will leave only the latest backup set. Both actions need to be used with `--force` to delete files.

During housekeeping, you can use `list-current-files` to see what files are currently backed up in an archive (Figure 3). For a quicker response time, this information is read from `duplicity`'s signature files rather than the archive, which may mean that a corrupted file is not detected. To check the state of files directly, you can use `verify`, with a verbosity level of 4 or higher (see the Selected Options section for more information) to see a separate message for each altered file. Alternatively, use `collection-status` to see the backup's status, a summary of the sets that the backup contains, and the number of tar files in each.

To restore files, simply reverse the source and the target:

```
duplicity scp://TARGET-UID/backup/* 🔍
/home/tt/
```

If a file or directory already exists, then `duplicity` will not overwrite unless the `--force` option is used.

Selected Options

`duplicity`'s options are all in long form, rather than being a single character. A large number specify what to include or exclude. `duplicity` includes two matching sets of options that begin with `--exclude` or `--include`, which are completed by specifics. For example, `--exclude-shell_pattern PATTERN` and `--include-shell_pattern PATTERN` are completed with a string of characters. Should either match a directory, then all the directory's subfiles are affected. Alternatively, you can use the `regex` completion. The completion `device-files DEVICE` can be used on exterior devices. Still other completions are `filelist`, `if-present FILENAME`, or, for `--exclude`, `--exclude-other-filesystems FILESYSTEM`. Another separate option for excluding files is `--extra-clean`, which saves space by not backing up things like the signature files for old backups.

Other options modify `duplicity`'s basic behavior. With `--encrypt-key KEY`, you can specify which GnuPG key to use. If you are not sure whether you need a full backup, you can use `--full-if-older-than TIME`, which will only do a full backup if the last one was before a certain time. When restoring, you can use `--rename OLD-PATH NEW-PATH` to change the directory where the files are written; otherwise, the archived files will be restored to their original location, overwriting any files there.

If you receive errors, you might try again with `--num-retries NUMBER` or even `--ignore-errors`. Should you want to add two backups to the same direc-

```

bb@nanday:~$ duplicity list-current-files file:///usr/local/home/bb/duplicity/cantarell
Local and Remote metadata are synchronized, no sync needed.
Last full backup date: Sun Dec 22 14:37:39 2019
Thu Apr 21 18:24:42 2016 .
Thu Apr 21 18:24:42 2016 Cantarell-Bold.ttf
Thu Apr 21 18:24:42 2016 Cantarell-BoldOblique.ttf
Thu Apr 21 18:24:42 2016 Cantarell-Oblique.ttf
Thu Apr 21 18:24:42 2016 Cantarell-Regular.ttf

```

Figure 3: Encrypted archives can only be detected from within `duplicity`.

tory, you can use `--allow-source-mismatch` or else wait for `duplicity` to prompt you to use that option. In truly desperate circumstances, you can use `--force`, either in anticipation or when prompted – although some data might be lost if you do. Perhaps, though, you may prefer to receive advance warning by testing what you are about to do with the addition of `--dry-run`, which will tell you what will happen without performing any action.

When using most of these options, you might want to adjust the verbosity level with `--verbosity LEVEL`. Like verbose actions in other commands, `duplicity`'s verbosity option adjusts the level of feedback you receive. In verbosity's case, this feedback includes what warnings are displayed. `duplicity` has nine levels of verbosity. Not all are named, but those with names are Error (0), Warning (2), Notice (4), Info (8), and Debug (9). Named levels may be displayed or set by name rather than number or by their initial letter. The default is Notice, which means that only notices, warnings, and errors are displayed. However, should you run into difficulties, you might want to increase the verbosity in the hopes of displaying information that can help you solve your difficulties. For example, if verbosity is set to Debug, you can see the full details of what a command does. The alternative is to look through the log for ignored errors – which is both inconvenient and inefficient, since no error summary is given at the end.

`duplicity`'s default protocol is SFTP/SCP/SSH, defaulting to SFTP, because it has fewer shell quoting problems than SCP. You can specify that SFTP/SCP look for an `FTP_PASSWORD` environmental variable, or prompt the user if the variable cannot be found. In addition, you can switch from the default SFTP protocol with `--use-scp`. If necessary, you can pass SSH options with:

```
--ssh-options oopt1=PARAMETER1 ?
          oopt2=PARAMETER2
```

Depending on the backup's location, you may also need other options. Using FTP, you may need to experiment with `--ftp-passive` and `--ftp-regular` if you are having trouble making connections. Similarly, when using Amazon S3, you may decide against the default `--s3-use-new-style` in favor of `--s3-european-buckets`. In addition, you may get faster uploads with `--s3-unencrypted-connections`, but at the cost of an insecure connection: Your archive will still be encrypted, but an ob-server could see the name of the bucket, your access key, and any increment dates and their sizes, as well as the fact that you are using `duplicity`.

An Efficient Command Structure

If `duplicity` has a fault, it is a lack of compression options. I am sure that I am not the only one who would like to have control over the archive's size. However, perhaps it is just as well, since compression and encryption

might result in too many things that could go wrong.

Still, by combining two related functions in a single command, `duplicity` remains one of the premier backup applications for Linux. Depending on your setup, you may take several tries to archive with `duplicity`. But, once you do, the chances are that you will use the same handful of command structures repeatedly, regardless of whether you are backing up a home system or a large network. Among the dozens of available backup tools, `duplicity` remains one of the most versatile and reliable.

In fact, the project site proposes replacing tar with `duplicity`, using a revised archival format [5]. The proposal cites tar's incompatibility with some filesystems, and its inelegant handling of long file names. The proposal does not seem to have reached the development stage, but it suggests the project's long-term goal is to improve Linux backup even more than it has already done. ■■■

Info

- [1] duplicity: <http://duplicity.nongnu.org/>
- [2] Google Drive: <https://blog.xmatthias.com/duplicity-google-drive/>
- [3] Amazon S3: <https://easyengine.io/tutorials/backups/duplicity-amazon-s3/>
- [4] GnuPG How-To: <https://www.gnupg.org/howtos/en/GPGMiniHowto.html>
- [5] Replacing tar: http://duplicity.nongnu.org/new_format.html



LinuxFest Northwest

April 24 - 26, 2020 • Bellingham Technical College

LFNW2020: “Be Excellent To Each Other”

1800+ attendees • 100+ sessions • 40+ vendors
Open hardware • Social events • Job fair • World famous raffle
All free to attend



<https://lfnw.org>



LinuxFest Northwest (est. 2000), an annual Open Source event co-produced by *Bellingham Linux Users Group* and the *Information Technology department at BTC*. LFNW features presentations and exhibits on free and open source topics, as well as Linux distributions & applications, InfoSec, and privacy; something for everyone from the novice to the professional!



Making an online archive of all your bookmarked pages

Preserve Your Favorite Pages

If you have a large collection of bookmarked pages, it's worth protecting! With the right scripts, you can create an archive so you never lose access to all your favorite web pages. *By Marco Fioretti*

The World Wide Web is so embedded in our lives that we often forget how ephemeral it is. Search engines can *find* content, but they can't ensure that the content is easily accessible over time [1]. Book-

marks are useful for storing information about websites, but you never know when the page will change or go offline. The only way to be sure you have permanent access to web content is to archive a copy of the page on your own system.

- A searchable index of all bookmarks, usable from any device
- A link to a personal, automatically archived copy of the page referenced in the bookmark

Because no single tool provides all the functionality I need, I have created my own solution using the Shaarli [5] bookmark manager, the ArchiveBox [6] self-hosted web archive tool, and a couple of custom scripts to glue it all together.

Shaarli and ArchiveBox

Shaarli describes itself as a “minimalist... database-free bookmarking service.” Figures 1 and 2 show the Shaarli main window and picture wall with the default theme.

Shaarli needs a web server configured to serve PHP pages over HTTPS connections. Once that requirement is satisfied, just upload the Shaarli files where the web server can find them, and then load the Shaarli home page in your browser. Set a user name and password and then click on *enable the REST API*. Drag the Shaarli bookmarklet to your browser's toolbar. The API setting makes it possible to export bookmarks, and the bookmarklet opens the Shaarli pop-up window shown in Figure 3.

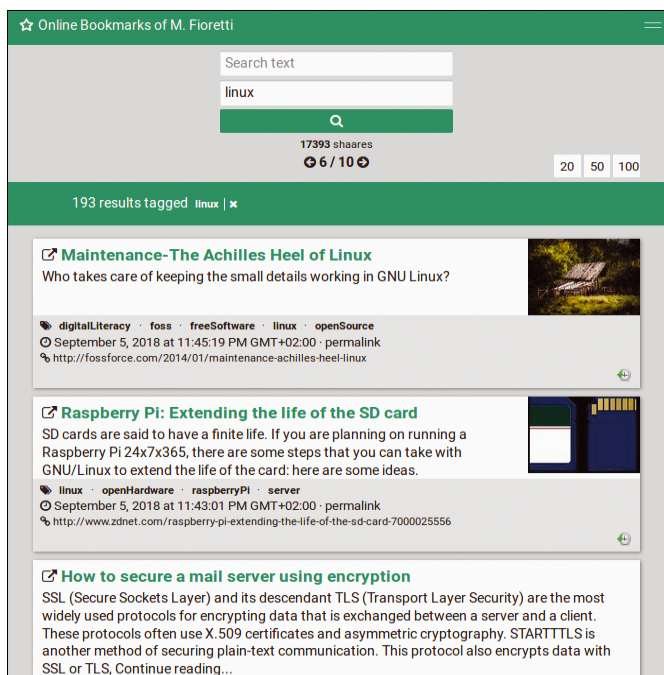


Figure 1: The Shaarli bookmark manager, as it appears to anonymous users.

I used to archive web pages with the Scrapbook extension for Firefox [2]. Today, you can do the same thing with add-ons like WebScrapBook [3] or SingleFile [4], but none of these ready-made solutions offer the scale, flexibility, and ease-of-use I need for my personal web archive. I need a solution that can handle thousands of bookmarks and has the following features:

Lead Image © Roman Motizov, 123RF.com

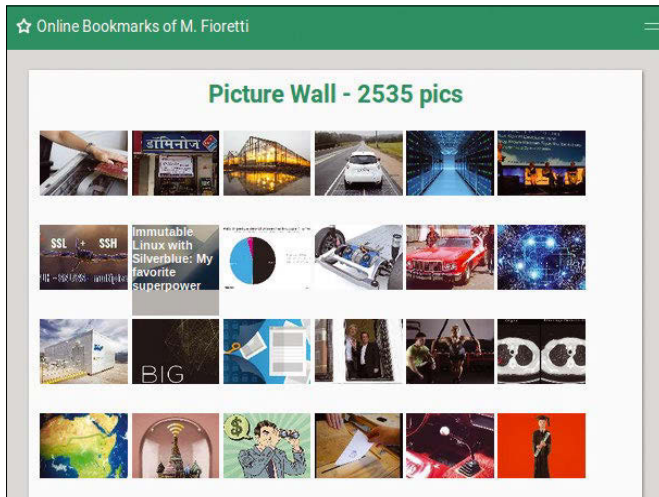


Figure 2: Shaarli can also display bookmarks as a snapshot gallery.

ArchiveBox is a Python-based, command-line front end to the `wget` and `youtube-dl` downloaders, the `pywb` “web recorder,” and a headless version of the Chromium browser. To use ArchiveBox, just give it all the URLs you wish for it to archive. ArchiveBox will save copies of the web pages in several formats, including PDF (with JavaScript and multimedia), a single-file WAR archive, and graphic versions. ArchiveBox also generates one index per page of all the formats it archives (see Figure 4), as well as a general index.

You can install ArchiveBox on Linux in several ways, all well documented on the website, but its dependencies may be a problem. To work well, ArchiveBox needs relatively recent versions of all dependencies. If all the necessary binary packages are not available in the standard repositories of your Linux distribution, the whole process may take more time than you can afford. The alternative to installing all these dependencies separately is to use the official Docker container for ArchiveBox. The script described in this article uses the Docker version, but it will work with native installations of ArchiveBox with changes to one or two lines of code.

Gluing It All Together

Shaarli stores bookmarks as one huge, encoded string inside one of its own source files, Listing 1 is a wrapper script for the `shaarli` client [7] that lets you export Shaarli bookmarks in JSON format.

Line 3 of Listing 1 saves the current date in the `$D` variable. Line 4 loads the Python virtual environment in which the Shaarli client needs to work. Line 5

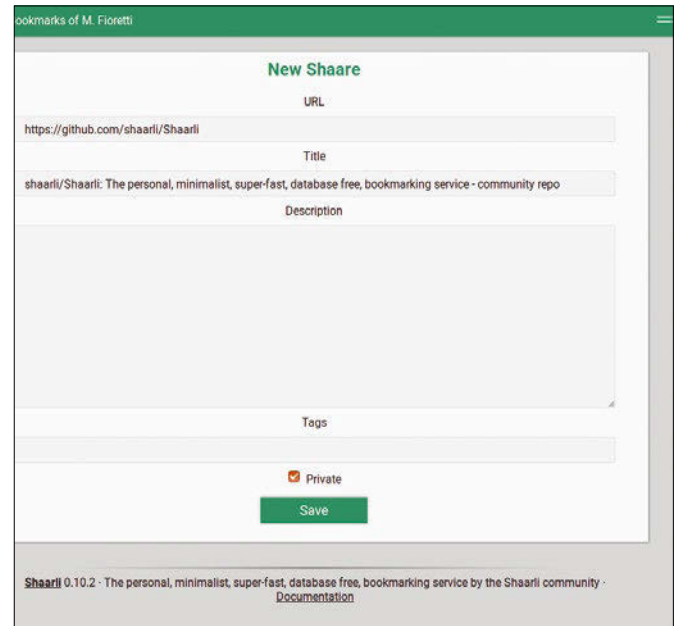


Figure 3: The Shaarli pop-up window to add and tag new bookmarks.

Listing 1: The Shaarli Bookmarks Export Script

```
01 #!/bin/bash
02
03 D=`date +%Y%m%d%H%M%S`
04 source /root/scripts/virtualenvs/shaarli3/bin/activate
05 shaarli -c /root/scripts/shaarli.conf -f json get-links --limit all >
   $D-shaarli.json
06 exit
```



Figure 4: ArchiveBox lists all of one web page’s archived formats: HTML, PDF, screenshot, and more.

runs the client, telling it to obtain a configuration file from `shaarli.conf` and save all the bookmarks in JSON format [8]. The result is a file with a name like `20190915113030-shaarli.json` that contains one huge JSON string. A snippet of the file looks similar to the following:

```
{ "id": 2109, "url": "https://example.com/some/web/page/", "shorturl": "7IewwA", "title": "Web Page Title", ... }
```

This snippet shows all you need to know about the Shaarli JSON format: All the variables of each bookmark ("id", "url", "title", etc.) are stored as key-value pairs, enclosed in double quotes, and separated by colons.

The key element used to integrate Shaarli with ArchiveBox is the unique numeric identifier ("id" in the snippet above) that Shaarli assigns to each bookmark. The internal Shaarli link to edit bookmark number 2109 would have the format `?edit_link=2109`.

The Shaarli code that displays those strings appears in the template file `shaarli/tpl/default/linklist.html`:

```
<a href="?edit_link={$value.id}" f1f1 title="{\$strEdit}"><i class="fa f1f1 fa-pencil-square-o edit-link"></i></a>
```

The task is to add a *LOCAL COPY* link

(Figure 5) that points to a folder with the same name as the ID value (Figure 6):

```
<a href="?edit_link={$value.id}" title="{\$strEdit}"><i class="fa fa-pencil-square-o edit-link"></i></a><a href="http://example.com/webarchive/{$value.id}/" target="_blank">LOCAL COPY</a>
```

Because the icon to edit bookmarks is displayed only after the user has logged in, placing the extra code right after the code for the edit icon makes sure that the links to each *LOCAL COPY* are visible only to logged in users.

At this point, the script in Listing 2 is all you need to actually archive bookmarks inside folders that Shaarli can link to automatically.

The main phases of the `shaarlibox` script in Listing 2 are as follows:

1. Set or load configuration variables (lines 3 to 8).
2. Create a "sandbox" for the container to run into, and make it writable to everybody (lines 10 and 12).
3. Load ID number and URLs of all bookmarks (line 15 and 16).
4. Archive all those URLs, one at a time, with the official Docker image for ArchiveBox (the loop in lines 20 to 71).
5. Move the main logfile to the parent folder and exit (lines 72 and 73).

The sandbox is the `$ARCHIVEBOX_HOME`, which is

recreated from scratch every time the script runs. Making that folder world-writable (line 12) is necessary because the container creates folders and files with user IDs.

`$SHAARLIHOME` is a web-accessible folder where the archived copies are saved. You can give it any value – as long as it corresponds to the `example.com/webarchive` string you added to the Shaarli template file.

`$ADDED_BOOKMARKS` in line 18 counts how many bookmarks have been archived. Lines 67 to 70 terminate the script as soon as that counter equals the value of `$BOOKMARKSLIMIT` set in line 5. I strongly suggest that you set this variable to a very low value, say 10 or 20, the first time you run the script. In this way, you can quickly get an idea of how much time and space it would take to archive everything.

The main logfile (`$LOG`) has a unique prefix, corresponding to the date and hour when the script starts (lines 6 and 7). Finally, `$SHAARLI_JSON` is the file generated by the `shaarli` backup script in Listing 1, which is passed as the first argument to `shaarlibox`.

The pipeline of text-processing commands in Phase 3 work as follows: `first`, `perl`, `grep`, and `cut` extract the `id` and `url` fields of all bookmarks from the JSON file. Their final output is then sorted in reverse numerical order (placing the most recent bookmarks first), using the `id` numbers as keys. To see what the options of each command mean, please consult the correspond-

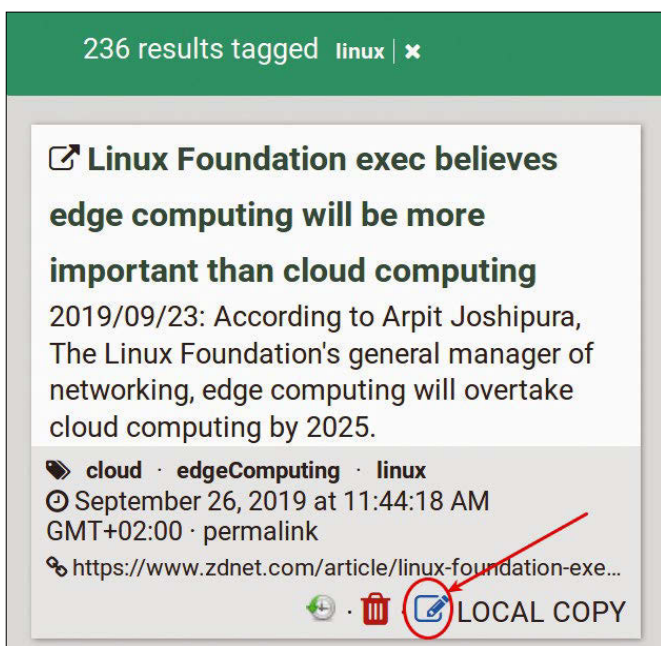


Figure 5: The *Edit* icon right below each Shaarli bookmark, and the added link to its *LOCAL COPY*.

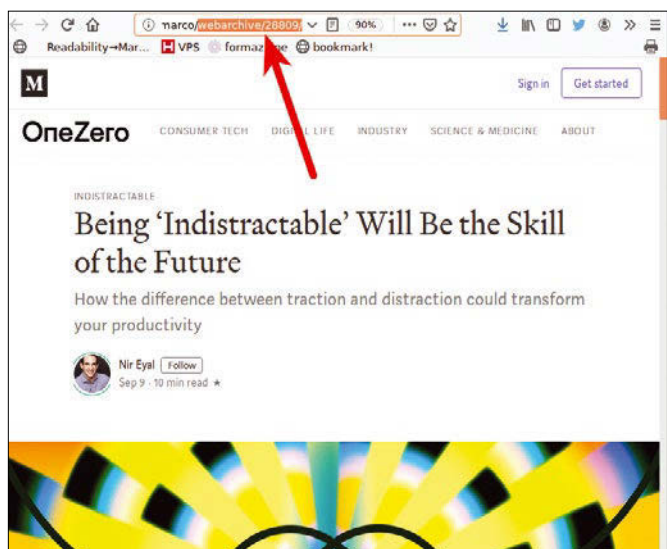


Figure 6: A web page archived with a local URL that matches its Shaarli identifier.

ing man pages. The final result, which is saved in the file `newest_bookmarks_first.csv`, has this format (the URLs are trimmed for clarity):

```
https://www.zdnet.com/article/2
linux-foundation.../|296
https://www.linux.com/audience/2
devops/...|295
https://www.linuxuprising.com/2
2019/08/...|294
```

The main loop (lines 20 to 71) reads that file, one line at a time, saving the current ID and URL values in `$BOOKMARKNUM` and `$CURRENT_BOOKMARK` (lines 22 and 23). All the `printf` statements in the loop write what is happening to one main log file (`$LOG`) for later analysis and debugging. Lines 26 to 28 check if `$SHAARLIHOME` already contains a subfolder called `$BOOKMARKNUM`, which means that the bookmark was already archived in a previous run, so nothing else happens. Otherwise, the script writes the `$CURRENT_BOOKMARK` inside the file `url_list.csv` (line 31) that will serve as the input for ArchiveBox.

You might wish to modify the script to write, say, 10 or 20 URLs at a time inside `newest_bookmarks_first.csv` and then fetch them all with just one call of the ArchiveBox container. Calling that container once per bookmark slows the script down. However, if the script downloads just one bookmark per container call, with `$BOOKMARKSLIMIT` set to a very low value, it is much quicker to check if you are using the best combination of options for ArchiveBox before letting it loose on all your bookmarks.

The actual archiving happens in line 33. This command, which was copied from the documentation, downloads the official container image of ArchiveBox and all its dependencies and runs it with Docker. My only additions are the optional settings `WGET_USER_AGENT` and `FETCH_WARC`. The first tells the `wget` download utility inside the container to identify itself as some desktop browser whenever it visits a website. This step is necessary, because some websites will refuse to serve pages to automatic downloaders. The `FETCH_WARC` setting prevents ArchiveBox from creating WARC archives of each bookmark, to save some disk space.

Just like Shaarli, ArchiveBox gives to each URL it processes a unique numeric identifier (which I decided to call `$ARCHIVEBOXFOLDER`) and saves all the copies in a folder with the same name, inside the

local archive directory. Since only one URL is processed every time, whatever subfolder there is in archive is named for the numeric identifier, and finding its value is the task of line 45.

Listing 2: shaarlibox

```
01 #! /bin/bash
02
03 ARCHIVEBOXHOME=/root/archiveboxsandbox
04 SHAARLIHOME='/var/www/html/webarchive'
05 BOOKMARKSLIMIT=10
06 NOW=`date +%Y%m%d%H%M`
07 LOG="$NOW-shaarlibox.log"
08 SHAARLI_JSON=$1 # shaarli bookmarks in JSON format
09
10 rm-rf $ARCHIVEBOXHOME
11 mkdir -p $ARCHIVEBOXHOME
12 chmod 777 $ARCHIVEBOXHOME
13 cd $ARCHIVEBOXHOME
14
15 perl -e 'while (<>) {s/"id": (\d+?),.*?"url": "([^\"]+)"/\nLINK|$2|$1\n/g; print}'
    $SHAARLI_JSON | \
16 grep ^LINK | cut '-d|' -f2- | sort -t '|' -k 2 -n -r > newest_bookmarks_first.csv
17
18 ADDED_BOOKMARKS=1
19
20 while IFS= read -r line
21 do
22 BOOKMARKNUM=`echo $line | cut '-d|' -f2`
23 CURRENT_BOOKMARK=`echo $line | cut '-d|' -f1`
24 printf "SB: \nSB: \nSB: %7.7s bookmark %9.9s : %s;\n" $BOOKMARKNUM
    "$ADDED_BOOKMARKS/$BOOKMARKSLIMIT" $CURRENT_BOOKMARK >> $LOG
25 printf "SB: %7.7s edit: https://bookmarks.zona-m.net/?edit_link=%s\n\n"
    $BOOKMARKNUM $BOOKMARKNUM >> $LOG
26 if [ -d $SHAARLIHOME/$BOOKMARKNUM ]
27 then
28 printf "SB: %7.7s already archived: %s;\n" $BOOKMARKNUM $CURRENT_BOOKMARK >>
    $LOG
29 else
30 printf "SB: %7.7s now archiving to : %s\n" $BOOKMARKNUM
    "$SHAARLIHOME/$BOOKMARKNUM" >> $LOG
31 echo $CURRENT_BOOKMARK > url_list.csv
32
33 cat url_list.csv | docker run -i -v $ARCHIVEBOXHOME:/data nikisweeting/
    archivebox env WGET_USER_AGENT="Mozilla/5.0 (Windows NT 10.0; Win64; x64)
    AppleWebKit/537.36 (KHTML, like Gecko) Chrome/73.0.3683.75 Safari/537.36" env
    FETCH_WARC=False /bin/archive &> archive2shaarli.$BOOKMARKNUM.log
34
35 COREFILES=`find . -type f -name "core.*" | perl -e 'while (<>)
    {print if m/core\.(\d+$/)} | wc -l`
36 if [[ "$COREFILES" -gt "0" ]]
37 then
38 printf "SB: %7.7s WARNING! CORE(s) in: %s\n" $BOOKMARKNUM
    "$SHAARLIHOME/$BOOKMARKNUM" >> $LOG
39 rm -f archive/*core.* ; rm -f archive/core.*
40 fi
41 printf "\n#####\n\n" >> $LOG
```


When ArchiveBox archives a web page inside `$ARCHIVEBOXFOLDER`, the `index.html` file inside that directory has that page's URL if the archive process

succeeds; otherwise it is set to *Not yet archived*.... If such a string is found in the `index.html` file, the `$FAILURE` flag is set to 1 (line 47). When that happens,

the code in lines 49 to 55 creates a new folder in `$SHAARLIHOME/$BOOKMARKNUM` and saves the ArchiveBox logfile inside it, in HTML format. In this way, if I click on the *LOCAL COPY* link for the bookmark inside Shaarli, I will see what happened (Figure 7), even if I missed it in the general `$LOG` file.

If `$FAILURE` is zero, it means that ArchiveBox succeeded, and all the versions of the current bookmark are saved and indexed inside `archive/$ARCHIVEBOXFOLDER`. So, aside from logging the event, all that is left to do is (lines 56 to 66) to move `archive/$ARCHIVEBOXFOLDER` to `$SHAARLIHOME/$BOOKMARKNUM` (which is where the *LOCAL COPY* link in Shaarli points), fix some relative links of the `index.html` file (lines 59 and 60, see below for details), increment the `$ADDED_BOOKMARKS` counter, and remove all the auxiliary files left by ArchiveBox (line 65). This last step is necessary because if the next run of ArchiveBox found those files, it would create extra folders, which would break the simple command used in line 45 to find `$ARCHIVEBOXFOLDER`.

The first time ArchiveBox runs, or believes it is running, it creates one `static` folder, at the same level as the `archive` folder, where it puts all the icons and CSS files used by the indexes of each page. Therefore, all the links to those resources inside each `index.html` file have the form `../..static/`. Since I needed the `static` folder to be inside `$SHAARLIHOME`, all those strings must change from `../..static` to `../static`, which is the purpose of the Perl command in line 59.

General Set Up and Maintenance

The script in Listing 2 makes one ArchiveBox container call per bookmark. This makes it noticeably slower, but much simpler and much more future-proof. As is, if you ever want to replace ArchiveBox with another archiver, you only have to rewrite two lines of code (33 and 45)! Besides, if ArchiveBox processed 10 or 20 bookmarks in each run it would be more complicated to match them with their Shaarli identifiers, and much easier to fill your hard drive, or miss downloading errors.

Due to lack of space, I can only briefly discuss three possible improvements

Listing 2: shaarlibox (Continued)

```

42 cat archive2shaarli.$BOOKMARKNUM.log >> $LOG
43 printf "\n#####\n" >> $LOG
44
45 ARCHIVEBOXFOLDER=`find archive -type d | grep / | cut -d/ -f2 | sort | uniq`
46
47 FAILURE=`grep -c '<title>Not yet archived...</title>'
archive/$ARCHIVEBOXFOLDER/index.html`
48 if [[ "$FAILURE" -ge "1" ]]
49 then
50 printf "SB: %7.7s failed to archive %s\n" $BOOKMARKNUM $CURRENT_BOOKMARK >> $LOG
51 mkdir $SHAARLIHOME/$BOOKMARKNUM
52 echo "<html><title>Failed to archive $CURRENT_BOOKMARK</title></
head><body><pre>" > $SHAARLIHOME/$BOOKMARKNUM/index.html
53 cat archive2shaarli.$BOOKMARKNUM.log >> $SHAARLIHOME/$BOOKMARKNUM/index.html
54 echo '</pre></body></html>' >> $SHAARLIHOME/$BOOKMARKNUM/index.html
55 rm -rf archive/$ARCHIVEBOXFOLDER
56 else
57 printf "SB: %7.7s moving copy from : archive/$ARCHIVEBOXFOLDER to
$SHAARLIHOME/$BOOKMARKNUM\n" $BOOKMARKNUM >> $LOG
58 mv archive/$ARCHIVEBOXFOLDER $SHAARLIHOME/$BOOKMARKNUM
59 perl -pi.bak -e 's/\.\.\.\.\.\./static\/\.\.\.\.\./g'
$SHAARLIHOME/$BOOKMARKNUM/index.html
60 rm $SHAARLIHOME/$BOOKMARKNUM/index.html.bak
61 mv archive2shaarli.$BOOKMARKNUM.log $SHAARLIHOME/$BOOKMARKNUM/
62 fi
63 printf "SB: %7.7s see local copy: http://zona-m.net/marco/webarchive/%s\n"
$BOOKMARKNUM $BOOKMARKNUM >> $LOG
64 ((ADDED_BOOKMARKS++))
65 rm -rf sources static index.html index.json robots.txt
66 fi
67 if [[ "$ADDED_BOOKMARKS" -gt "BOOKMARKSLIMIT" ]]
68 then
69 break
70 fi
71 done < newest_bookmarks_first.csv
72 mv $LOG ../
73 exit

```

```

[*] [2019-10-01 10:53:17] Parsing new links from output/sources/stdin-1569927197.txt...
> Adding 1 new links to index (parsed import as Plain Text)
[*] [2019-10-01 10:53:17] Saving main index files...
> /data/index.json
â`s /data/index.json
> /data/index.html
â`s /data/index.html
[â-¶] [2019-10-01 10:53:17] Updating content for 1 pages in archive...

[+] [2019-10-01 10:53:17] "http://www.cablemodem.com"
http://www.cablemodem.com
> /data/archive/1569927197
> title
Failed: Unable to detect page title
Run to see full output:
cd /data/archive/1569927197;
curl http://www.cablemodem.com | grep

```

Figure 7: The error page generated by the script when ArchiveBox cannot save a bookmark.

that will strengthen this bookmark archiving system: privacy, backups, and disk space. Shaarli bookmarks can be all private by default, but the archive itself will be private only if you password-protect the `$SHAARLIHOME` folder at the web server level. For backups, instead, the only specific information you need is which Shaarli files to back up [9].

Using the options in Listing 2, archiving about 2,100 bookmarks created over 230K files on my server, requiring more than 23GB. You can save lots of space, however, by choosing an efficient archive format and using a tool like `rdfind` [10] to replace duplicate files with hard links.

Long-term maintenance of the whole system is relatively simple, but necessary. On the ArchiveBox side, you might need to update the `shaarliibox` script if new versions use different command-line options or directory structures. With Shaarli, whenever you upgrade to a new version, you will need to manually patch the template.

Final Thoughts and Advice

I have shown how this integrated bookmarking and archiving system is both flexible and, at least on the archival side, backward-compatible. Porting the system to a different archiver is quite simple, as is importing bookmarks previously archived with *other* systems. Replacing Shaarli with another bookmarking tool is more difficult, but it is still possible.

The last thing I want to share about archiving web pages is that some issues cannot be solved by coding. No automatic archival procedure for *old* book-

marks can travel back in time. If a page is 10 years old, archiving it today has quite a different value and meaning, than archiving it on the day it was published. And if an archived page has a dynamic *Breaking News* box, that box will likely display different breaking news every time you load it, not the news of the day when it was archived.

Another big issue is completeness. ArchiveBox works very well, but the only way to be 100 percent sure that you did archive usable copies of *all* your bookmarks is to check all those copies yourself, one by one. A problem could arise because a website might generate error pages that ArchiveBox does not recognize. Another issue is that websites might deliberately serve misleading information to clients that they do not recognize, or that do not execute JavaScript automatically. For instance, Medium.com served articles to my browser without problems, but when I tried to access the page through ArchiveBox, a message told me that the page did not exist.

I later discovered that if I loaded the archived copy in a text-only browser, it

Author

Marco Fioretti (<http://stop.zona-m.net>) is a freelance author, trainer, and researcher based in Rome, Italy, who has been working with Free/Open Source software since 1995, and on open digital standards since 2005. Marco also is a board member of the Free Knowledge Institute (<http://freeknowledge.eu>).



would display the whole article. It turns out the error message was caused by a JavaScript file. When I renamed that file and reloaded the archived copy, it rendered almost exactly as the original page. ■■■

Info

- [1] “Indeed, it seems that Google IS forgetting the old Web”: <http://stop.zona-m.net/2018/01/indeed-it-seems-that-google-is-forgetting-the-old-web/>
- [2] Scrapbook: A Firefox extension for personal Web archives and more: <https://www.techrepublic.com/blog/linux-and-open-source/scrapbook-a-firefox-extension-for-personal-web-archives-and-more/>
- [3] WebScrapBook: <https://github.com/danny0838/web ScrapBook>
- [4] SingleFile: <https://addons.mozilla.org/en-US/firefox/addon/single-file/>
- [5] Shaarli: <https://github.com/shaarli/Shaarli>
- [6] ArchiveBox: <https://archivebox.io>
- [7] Shaarli CLI client: <https://github.com/shaarli/python-shaarli-client>
- [8] Shaarli client installation: <https://github.com/shaarli/python-shaarli-client/blob/master/docs/user/installation.rst>
- [9] How to backup a Shaarli instance: <https://shaarli.readthedocs.io/en/latest/guides/backup-restore-import-export/>
- [10] Rdfind: <https://rdfind.pauldreik.se/>



MakerSpace

Using the curses library to view IoT data Old School

When you need some quick graphical output, the old school curses library can save you some time and effort. *By Pete Metcalfe*

Many projects require a lot of time building colorful web pages or custom graphical user interfaces (GUIs). In a number of cases, however, only a simple text interface is required, especially for remote connections into a Raspberry Pi when you just want a quick system update.

In this article, I review a 1980s technology called `curses` [1], which lets you create text-based applications without requiring X windows or web services. In one example I look at C and Python apps that simulate Raspberry Pi sensor data, and in two examples, I output large text presentations and dynamic bars in Python [2].

Python Curses

The `curses` module is standard in Python and includes features such as:

- dynamic screen positioning
- ASCII box-drawing characters
- basic color support
- window and pad objects

As a first example, I create an interface with a color background, header, footer, and dynamic text that simulates sensor data to a Raspberry Pi (Figure 1; Listing 1).

The first step in a `curses` app is to define a main screen object (`stdscr`, line 4). The next step is to enable color and create some color pairs (lines 8-11). Color pair 3 sets the background to blue

Listing 1: `curses_text.py`

```
01 import curses , time, random
02
03 # create a curses object
04 stdscr = curses.initscr()
05 height, width = stdscr.getmaxyx() # get the window size
06
07 # define two color pairs, 1- header/footer , 2 - dynamic text, 3 - background
08 curses.start_color()
09 curses.init_pair(1, curses.COLOR_RED, curses.COLOR_WHITE)
10 curses.init_pair(2, curses.COLOR_GREEN, curses.COLOR_BLACK)
11 curses.init_pair(3, curses.COLOR_WHITE, curses.COLOR_BLUE)
12
13 # Write a header and footer, first write colored strip, then write text
14 stdscr.bkgd(curses.color_pair(3))
15 stdscr.addstr(0, 0, " " * width, curses.color_pair(1) )
16 stdscr.addstr(height-1, 0, " " * (width - 1), curses.color_pair(1) )
17 stdscr.addstr(0, 0, " Curses Dynamic Text Example" ,curses.color_pair(1) )
18 stdscr.addstr(height-1, 0, " Key Commands : q - to quit " ,curses.color_pair(1) )
19 stdscr.addstr(3, 5, "RASPBERRY PI SIMULATED SENSOR VALUES" ,curses.A_BOLD )
20 stdscr.refresh()
21
22 # Cycle to update text. Enter a 'q' to quit
23 k = 0
24 stdscr.nodelay(1)
25 while (k != ord('q')):
26     # write 10 lines text with a label and then some random numbers
27     for i in range(1,11):
28         stdscr.addstr(4+ i, 5, "Sensor " + str(i) + " : " ,curses.A_BOLD )
29         stdscr.addstr(4+ i, 20, str(random.randint(10,99)) ,curses.color_pair(2) )
30         time.sleep(2)
31         k = stdscr.getch()
32
33 curses.endwin()
```

Lead image © Andrey Kiselev, 123RF.com

(line 14). Lines 15-18 use color pairs and the screen size (with height and width, defined in line 5) to add a header and footer strip.

The `stdscr.nodelay` command allows the program to cycle until the `stdscr.getch()` call returns a key stroke. The simulated Rasp Pi values refresh every 10 seconds until the `q` key is captured (line 25), after which, the terminal settings are returned to normal (`curses.endwin()`) and the program exits.

This simulated Raspberry Pi example only took about 30 lines of code, which is significantly less than if an equivalent web application were used.

C curses Example

For the C example, I used a Raspberry Pi and the previous Python example. Before you begin, you need to install the curses library:

```
sudo apt-get install libncurses5-dev
```

The curses syntax is similar between C and Python, but not identical (Listing 2). For example, in Python the `addstr` command includes a color pair reference. In C this is not supported, so an on/off attribute (`attron/attroff`) toggles the color pair. However, C does support a formatted string write command, `mvprintw` (lines 38 and 42).

To compile and run the program, enter:

```
gcc -o c1 c1.c -lncurses
./c1
```

With the exception of the caption in the top line, the C and Python output look identical.

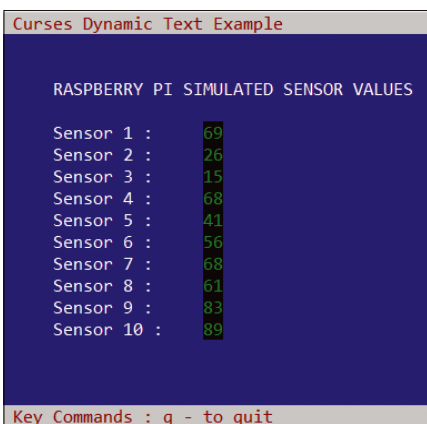


Figure 1: Curses dynamic text example.

Listing 2: c1.c

```
01 /* c1.c - Basic Curses Example */
02
03 #include <curses.h>
04 #include <stdlib.h>
05 #include <unistd.h>
06
07 int main(void)
08 {
09     int row, col, k;
10 // Create a curses object and define color pairs
11     initscr();
12     getmaxyx(stdscr, row, col);
13     start_color();
14     init_pair(1, COLOR_RED, COLOR_WHITE);
15     init_pair(2, COLOR_GREEN, COLOR_BLACK);
16     init_pair(3, COLOR_WHITE, COLOR_BLUE);
17     curs_set(0);
18     noecho();
19     nodelay(stdscr, TRUE);
20 // Write a header and footer, first write colored strip, then write text
21     bkgd(COLOR_PAIR(3));
22     attron(COLOR_PAIR(1));
23 // Create a top and bottom color strip
24     for (int i = 0; i < col; i++) {
25         mvaddstr(0, i, " ");
26         mvaddstr(row-1, i, " ");
27     }
28     mvaddstr(0, 0, " Curses C Dynamic Text Example");
29     mvaddstr(row-1, 0, " Key Commands: q - to quit");
30     attroff(COLOR_PAIR(1));
31     mvaddstr(2, 5, "RASPBERRY PI SIMULATED SENSOR VALUES" );
32     refresh();
33 // Cycle with new values every 2 seconds until a q key (133) is entered
34     while (k != 113)
35     {
36         attroff(COLOR_PAIR(2));
37         for (int i = 0; i < 10; i++) {
38             mvprintw((4+i), 5, " Sensor %d : ", i);
39         }
40         attron(COLOR_PAIR(2));
41         for (int i = 0; i < 10; i++) {
42             mvprintw((4+i), 20, "%d", rand() %100);
43         }
44         k = getch();
45         sleep(2);
46     }
47     endwin();
48     exit(0);
49 }
```

FIGlet for Large Custom Text

To generate large custom text for presentation, you can use the FIGlet library [3], which has an extensive selection of “fonts” created with standard ASCII characters. The FIGlet library is installed in Python with the command:

```
pip install pyfiglet
```

An example executed from the Python shell shows how it works:

```
>>> import pyfiglet
>>> value1 = pyfiglet.figlet_format(
    "12.3", font = "starwars" )
>>> print(value1)
  _ _ _
 /_ | |_\ |_\ \
 | | ) | | _ ) |
 | | / / | _ <
 | | / / _ _ ) |
 | | | | ( _ ) _ /
```

By combining curses with FIGlet, you can create some simple Raspberry Pi interfaces. A little bit of trial and error might be required to get a FIGlet font [4] that matches your requirements. I found that the `starwars` and `doom` fonts worked well for dynamic text and the `small` font was good for headings.

Listing 3 shows the code that generates the FIGlet large text in Figure 2. In this example, a `get_io()` function generates random numbers; for a true Rasp Pi project, this function would return sensor values.

Curses Windows

The examples so far have been based on a main screen object, but the curses library also supports windows. Windows are useful because they support border outlines, and text can be cleared from and written to windows



Figure 2: FIGlet large text.

Listing 3: bigtxt.py

```
01 import curses, time
02 import pyfiglet, random
03
04 def get_io(): # Get a random value. Tweek later with real data
05     global value1
06     testvalue = str(random.randint(100,1000)/10) + " C"
07     value1 = pyfiglet.figlet_format(testvalue, font = "starwars" )
08
09 # Create a string of text based on the Figlet font object
10 title = pyfiglet.figlet_format("Raspi Data", font = "small" )
11
12 stdscr = curses.initscr() # create a curses object
13 # Create a couple of color definitions
14 curses.start_color()
15 curses.init_pair(1, curses.COLOR_YELLOW, curses.COLOR_BLACK)
16 curses.init_pair(2, curses.COLOR_GREEN, curses.COLOR_BLACK)
17
18 # Write the BIG TITLE text string
19 stdscr.addstr(1,0, title,curses.color_pair(1) )
20 stdscr.addstr(8,0, "Sensor 1: GPIO 7 Temperature Reading" ,curses.A_BOLD)
21
22 # Cycle getting new data, enter a 'q' to quit
23 stdscr.nodelay(1)
24 k = 0
25 while (k != ord('q')):
26     get_io() # get the data values
27     stdscr.addstr(10,0, value1,curses.color_pair(2) )
28     stdscr.refresh()
29     time.sleep(2)
30
31     k = stdscr.getch()
32
33 curses.endwin()
```

without affecting the main screen object. The syntax to create a curses window object is:

```
mynewwindow = curses.newwin(
    height, width, begin_y, begin_x)
```

The code for Figure 3,

```
# define a win1 window object
win1 = curses.newwin(9, 44, 6, 4)
# write text inside the window object
win1.addstr(8,0, "
```



Figure 3: Large text in two curses windows.

Listing 4: bar.py

```

01 import curses
02
03 bar = ' ' # with reverse video a space will show up
04 value1 = 10 # in real life this needs to be scaled
05
06 stdscr = curses.initscr()
07 curses.curs_set(0) # don't show the cursor
08
09 stdscr.addstr(1,3, "Python Curses Bar")
10 stdscr.refresh()
11 # Define windows
12 win1 = curses.newwin(3, 32, 3, 2)
13 win1.border(0) # add a border
14 # a horizontal bar 10 characters wide
15 win1.addstr(1, 1, bar * value1,curses.A_REVERSE )
16 win1.refresh()
17
18 # Wait for a key press then exit
19 stdscr.getch()
20 curses.endwin() # restore the terminal to original settings
    
```

```

"Sensor 1: Temperature Reading",
curses.A_BOLD)
value1 = pyfiglet.figlet_format(
"23 C", font = "doom") win1.addstr(
0,0,value1,curses.color_pair(2) )
    
```

produces two curses windows. The

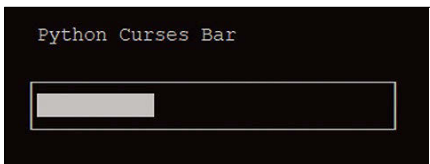


Figure 4: A simple curses bar.

major difference in the code is that information within a window is addressed by the window object instead of the background screen object.

Dynamic Bars

Simple progress or indicator bars can be created by a curses window with a border (Listing 4). The bar itself is generated by writing a space character with inverse video (Figure 4).

After I had the basic curses bar working, I was able to use what I learned in the earlier examples to create Raspberry

ported in a variety of programming languages. I focused my curses work here on C and Python, but I've also had good success in Lua.

I only had one presentation issue and that was with line drawing characters like borders when I used Putty (a Windows-based SSH program). To fix this issue, I changed the Putty *Window | Translation* setting to use the *VSCII* character set; then, everything looked good. ■■■

Info

- [1] Python curses: <https://docs.python.org/3/library/curses.html>
- [2] Code for this article: <ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/232/>
- [3] pyfiglet: <https://github.com/pwaller/pyfiglet>
- [4] FIGlet fonts: <http://www.figlet.org/examples.html>

Author

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

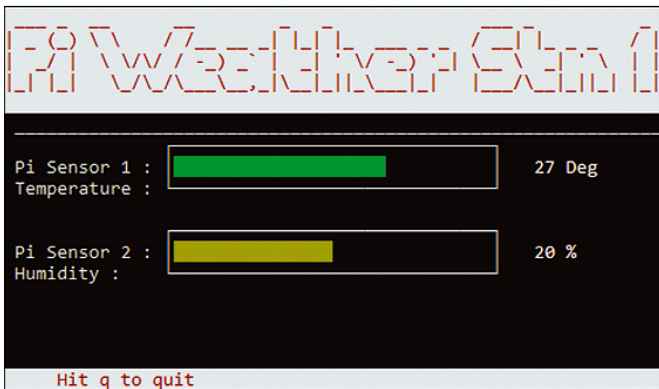
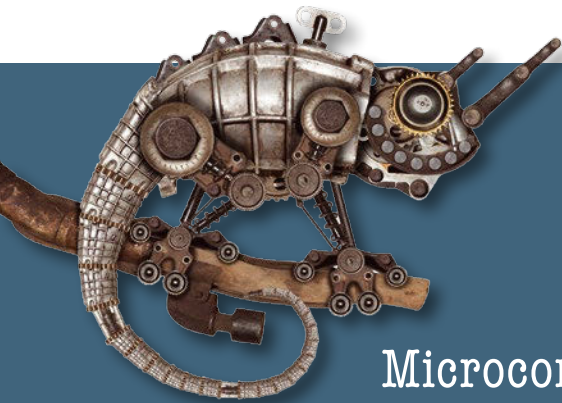


Figure 5: Raspberry Pi sensor data presented as bars.

Pi interfaces that included color and large text (Figure 5).

Summary

If you are looking for a quick and easy way to present data, without the bother of a GUI, curses is a great solution. Curses is sup-



MakerSpace

Microcontroller programming with BBC micro:bit Pocket-Size Programming

Designed for students, the BBC micro:bit, in conjunction with MicroPython and the Mu editor, can help you get started with microcontroller programming. *By Roland Pleger*

The idea of developing a microcontroller for schools dates back to 2012. In 2016, in cooperation with the University of Lancaster and several dozen industry partners, the British Broadcasting Corporation (BBC) delivered on this concept with the BBC micro:bit [1]. Although developed for seventh grade students, the BBC micro:bit offers an introduction to microcontroller programming for users of any age.

You can purchase the micro:bit individually for \$14.95 or as the micro:bit

Go Bundle (which includes batteries, a battery holder, and USB cable) for \$17.50 from Adafruit [2].

The microcontroller and its components fit on a 5x4cm board (Figure 1). The 32-bit processor, an ARM Cortex-M0, runs at a clock speed of 16MHz. The micro:bit offers 16KB of RAM and a 256KB flash memory. In principle, a Bluetooth Low Energy (BLE) radio is also available. See Table 1 for more specifications.

If you connect a micro:bit to a Raspberry Pi via a micro-USB cable, the Raspberry Pi will identify the micro:bit as a USB stick with a size of 64MB. Using the `lsusb` command will reveal that the micro:bit's USB interface component is an LPC1768 chip from NXP. Listing 1 reveals that the micro:bit has a device name of `ttyaACM3`.

The micro:bit can be powered via USB or battery, using the battery holder (shown in Figure 1 on the left) with two AAA zinc or alkaline batteries plugged into the top connector. Optionally, you can use the 3V pad at the bottom, but do so with caution. The micro:bit's voltage range is 1.8-3.6V.

The memory-hungry Bluetooth module cannot be addressed as a standard BLE in MicroPython, but a radio connection between modules should at least work.

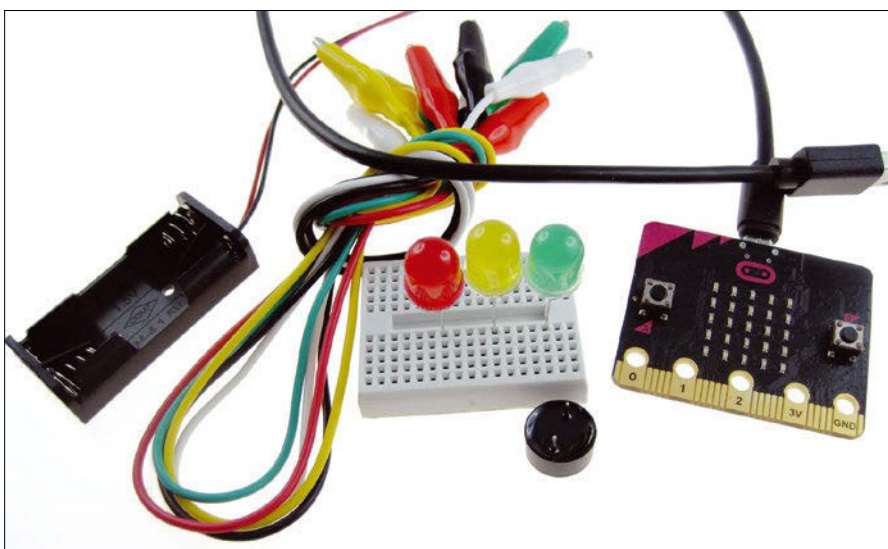


Figure 1: The micro:bit with connecting cables, a breadboard, and components.

Table 1: micro:bit Specifications

25 individually programmable LEDs (5x5 matrix)
Three-axis accelerometer
Three-axis magnetometer
Light and temperature sensors
Two programmable buttons
Micro USB socket for power supply and data transfer
Plug contact for external 3V power supply
Five contacts for crocodile clips (ground, 3V, and three I/O)
Further inputs and outputs via special connectors

Listing 1: micro:bit Device Name

```
$ lsusb
[...]
Bus 002 Device 019: ID 0d28:0204 NXP LPC1768
[...]
$ dmesg | grep tty
[...]
[17860.723466] cdc_acm 2-1.2.1:1.1: ttyACM3: USB ACM device
```

MicroPython

While the micro:bit understands various programming languages, this article focuses on MicroPython, the version of Python optimized for running on micro-processors.

If you want to program microcontrollers on register level, you won't get far with MicroPython, but then neither is the micro:bit the tool for this task. Instead, an Arduino [3] would be the better choice, which, unlike the micro:bit, offers circuit diagrams and board layouts.

The micro:bit's operating system constantly monitors the USB memory. If it detects a hex file in USB memory, the system transfers it to the internal memory as a byte sequence and executes it.

MicroPython comes with a very simple filesystem that allows programs to write files to the microcontroller and read them from there. However, even attaching data to existing files is too much for the system. The size limit is 30KB. Since the files are stored in internal memory, they can only be addressed by the running program, but not via the USB interface.

Mu

To program the micro:bit, you use the Mu editor [4], a simple Python editor. Mu recognizes the micro:bit at startup if it is connected to a computer via a USB

cable. Mu will prompt you to select the *BBC micro:bit* mode.

Due to the integrated libraries, the resulting programs are only executable on the micro:bit. This means that run-time errors can only be analyzed and output on the micro:bit. As soon as you press the *Check* button, the Mu editor searches for syntax errors. For example, in Listing 2, the word *hello* in the second line is

syntactically incorrect. The solution would be to comment out the line or quote the string (Figure 2).

Pressing the *Flash* button converts the program to hex code and transfers it to the micro:bit where it is executed immediately. You can restart the program via the reset button, which is just like switching on the micro:bit again by applying the supply voltage.

The code from Listing 2 terminates with a run-time error message in the Mu editor:

```
division by zero
```

The micro:bit outputs the error code as a ticker on its 5x5 LED matrix. You can look up the error code in the Mu editor by pressing the *REPL* (which stands for Read-Evaluate-Print-Loop) button (Figure 2).

A Simple Project

Listing 3 shows the code for a prank that delivers a high-pitched squeak when it's dark; if you turn on the light to find the sound source, the system remains silent.

Listing 2: Syntax Error

```
import microbit
print(hello)
print(1/0)
```

Listing 3: Squeak in the Dark

```
01 import microbit
02 import music
03 import random
04
05 BrightTrig = 50
06
07 while True:
08     br = microbit.display.read_light_level()
09     if br < BrightTrig:
10         microbit.sleep(800 * (1+random.randrange(10)))
11         music.pitch(900, 80 * (1+random.randrange(5)))
```

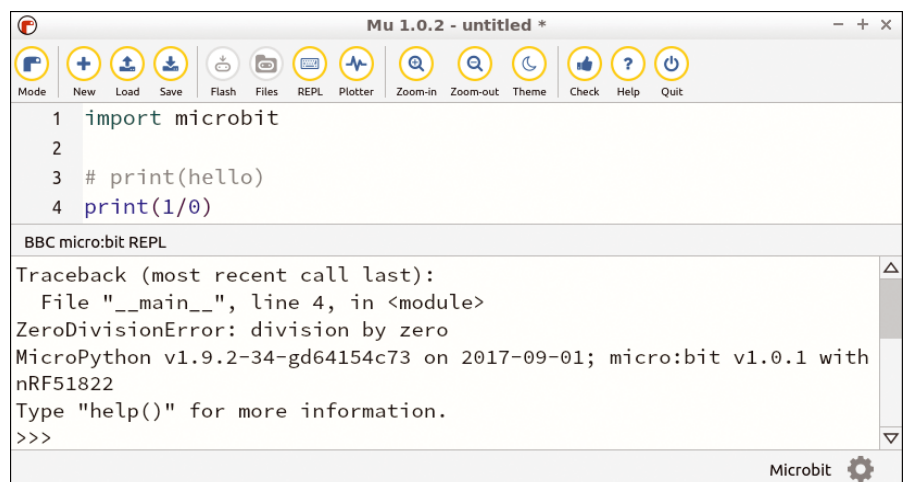


Figure 2: A MicroPython program with a run-time error message in the Mu editor.

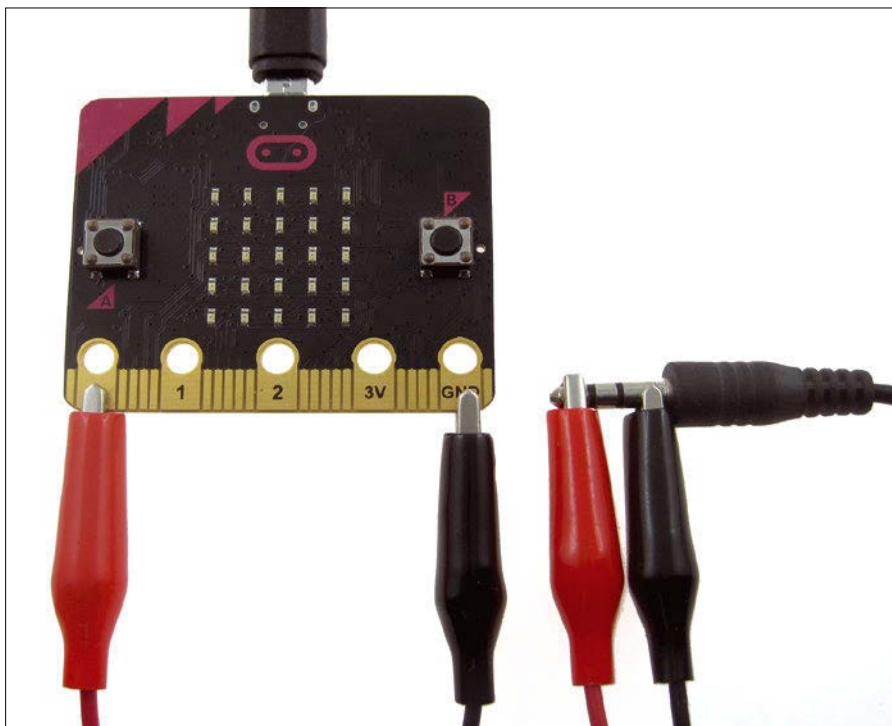


Figure 3: Crocodile clips can be used to connect a jack plug to the BBC micro:bit.

Lines 1-3 import the required Python libraries. The micro:bit not only makes its LEDs light up, it can also use them like photodiodes. Depending on the brightness, the `microbit.display.read_light_level()` command returns a value between 0 and 255. At a value of 50, it is already quite dark (line 5).

The continuous loop introduced by `while True` stores the measured brightness in the variable `br`. If it is greater than the threshold value `BrightTrig`, nothing happens. Otherwise, the program calls `microbit.sleep` (a wait command) that lasts at least 800 milliseconds, depending on the random number `random.randrange(10)`.

The command `music.pitch` generates a square wave signal of 900Hz at output 0. Here too, a random generator controls the duration.

By connecting a small loudspeaker to the micro:bit contacts 0 and GND, the prank begins. Crocodile clips connect a jack plug to an amplifier's line-in input (Figure 3). The micro:bit's power is just enough to connect a ceramic speaker (the small black cylinder shown in Figure 1). You can also use a buzzer (i.e., a ceramic loudspeaker with a built-in tone generator). Then you just need to switch the output on

and off again instead of modulating it at 900Hz.

MicroPython Commands

Table 2 lists the most important MicroPython commands for communicating

with the micro:bit's sensors. There are commands for the matrix display (Figure 4), the buttons, and the digital input and output.

The acceleration sensor also serves as a position sensor when the module is at rest. In order not to have to deal with the orientation angles of its three axes, there is the `microbit.accelerometer.current_gesture()` command. It understands the orientations up, down, left, right, face up, face down, and shake.

A magnetometer with three axes is also available. In theory, it is sensitive enough to detect the direction of the Earth's magnetic field – in practice, it is difficult to do so. The temperature sensor is similar: It also measures the processor temperature. However, under no circumstances should you heat up the board with a hair dryer – be content with measuring the ambient temperature.

The `microbit.pin0.is_touched()` command connects the digital input via a 10m ohm pull-up resistor. The micro:bit uses this to detect if there is too high of a resistance to ground. Typically, one finger is sufficient for this, while another finger simultaneously touches the micro:bit's ground cable

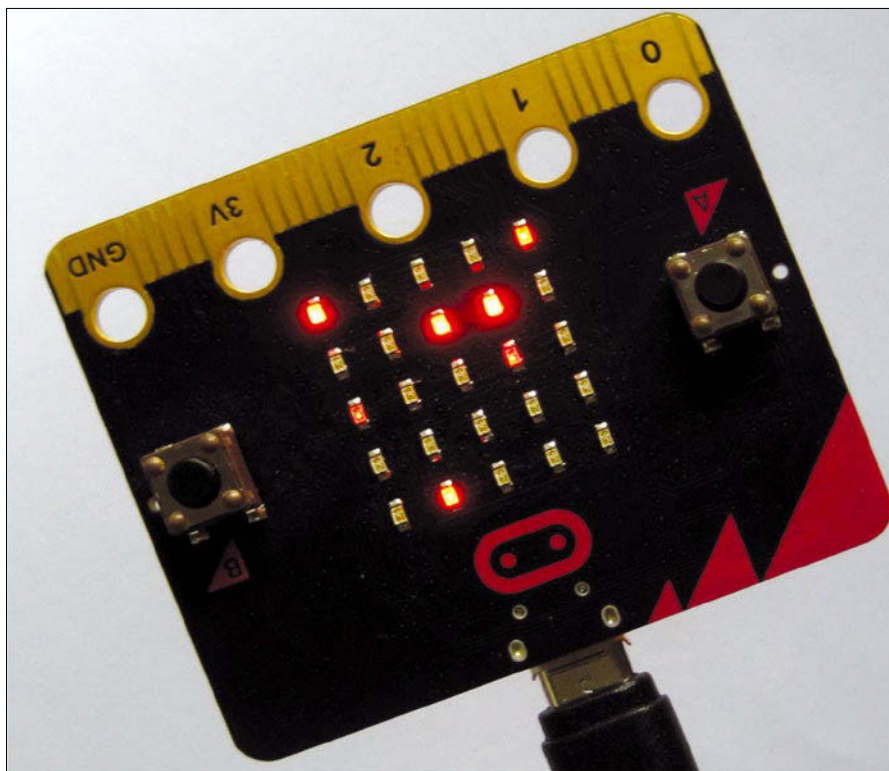


Figure 4: The LEDs in the micro:bit's matrix display can be actuated at different brightness levels.

(GND, the large contact at bottom right on the board).

Interface

Once flashed, the microcontroller no longer accepts commands. Instead, it switches to communication and outputs its information serially via the USB connection. In Figure 5, the connected micro:bit passes the brightness value to Mu's REPL command window via the print() command. If you close the program, you get direct access to the data.

You already know the device name for the serial interface, in our case tty-ACM3, as shown in Listing 1. The commands in Listing 4 redirect the micro:bit's output to the terminal window. The cat command returns the brightness value as a decimal number; od -x returns it as a hexadecimal number (Figure 6).

Hex Files

The Mu editor converts the program code into a hex file. If you disconnect the

micro:bit before flashing, the Mu editor enables access to the file and asks where to store it.

The hex file consists of an 8KB MicroPython interpreter. MicroPython is a reduced Python instruction set, which feels like real Python in its basic functions. Additionally, the file contains the program's byte sequence, which the MicroPython interpreter then executes. Finally, the hex file contains a compressed version of the actual program text including comments.

Table 2: micro:bit Commands

Function	Effect
microbit.display.show(Image.HAPPY)	Displays a smiley face on the LED matrix
microbit.display.show(Image.SAD)	Displays a frowning face on the LED matrix
microbit.display.set_pixel(1, 2, 9)	Switches the LED at position (1,2) to maximum brightness
microbit.button_a.is_pressed()	Checks whether button a is pressed
microbit.button_a.was_pressed()	Checks whether the button was pressed after switching on, or the last request
microbit.button_a.get_presses()	Counts how many times the button was pressed after the last request
microbit.accelerometer.get_x()	Reads the x-coordinates of the position sensor
microbit.pin0.read_digital()	Reads the digital input at pin 0
microbit.pin0.write_digital(1)	Sets the digital input at pin 0
music.play(music.RINGTONE)	Plays a tune if a speaker is connected to pin 0

The command is always preceded by the name of the matching MicroPython library.

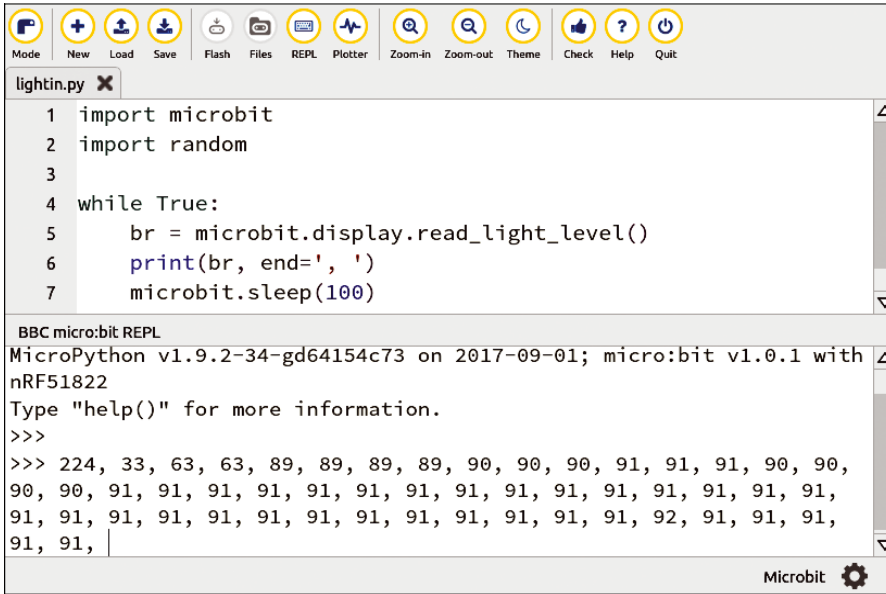
IT Highlights at a Glance

Too busy to wade through press releases and chatty tech news sites? Let us deliver the most relevant news, technical articles, and tool tips – straight to your Inbox.

Linux Update • ADMIN Update • ADMIN HPC

Keep your finger on the pulse of the IT industry.

Admin and HPC: <https://bit.ly/HPC-ADMIN-Update>
 Linux Update: <https://bit.ly/Linux-Update>



```

lightin.py
1 import microbit
2 import random
3
4 while True:
5     br = microbit.display.read_light_level()
6     print(br, end=', ')
7     microbit.sleep(100)

BBC micro:bit REPL
MicroPython v1.9.2-34-gd64154c73 on 2017-09-01; micro:bit v1.0.1 with nRF51822
Type "help()" for more information.
>>>
>>> 224, 33, 63, 63, 89, 89, 89, 89, 90, 90, 90, 90, 91, 91, 91, 90, 90,
90, 90, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91,
91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91, 91,
91, 91,

```

Figure 5: Outputting the brightness in the Mu REPL command window.

Contrary to what the name might suggest, the hex file is initially a text file: The address and bytes are written out as text (Listing 5). The file therefore shrinks as soon as the system converts the ASCII text into binary numbers. This explains why a hex file of more than 500KB fits into the micro:bit's memory, which has a capacity of 256KB – including the MicroPython interpreter. In the end there are about 8KB of physical memory for your own python scripts.

Flashing the micro:bit via a connected computer is a more elegant solution. The micro:bit logs on to the computer's filesystem as a USB stick, so you just need to copy the hex file to it. If the micro:bit recognizes the format, it converts the content into a byte sequence and immediately writes it to its flash memory. Otherwise, the file remains in USB memory, just like on a normal memory stick.

Although flashing always overwrites the old data, the data is retained when the power supply is disconnected. The micro:bit executes the program once it has been loaded, as soon as you resupply power.

Outlook

While the micro:bit's MicroPython software is documented in detail [5], the hardware is not. The outputs are probably short-circuit proof; they possibly also limit the current to 10 or 90mA. However, there are no reliable statements about this. For this reason, you should only connect light emitting diodes via a protective resistor.

The microcontroller has a wide range of internal interfaces; it can handle I2C, SPI, LIN, and UART, as well as USB and USB OTG. Many of the 23 inputs/outputs are also routed to the outside, but only as simple PCB plug contacts, which can only be used with a special connector.

Calliope Mini [6], a project managed by Germany's Calliope gGmbH, describes itself as a further development of the micro:bit. Calliope Mini circuit diagrams and software are freely available as open hardware or open source, and there are also several manuals as open educational resources.

Calliope Mini's circuit board has additional sensors like a microphone and loudspeaker, as well as four I/O contacts for crocodile clips (instead of the micro:bit's three I/O contacts). The two modules do not differ in terms of memory size. The Calliope developers route all other interface contacts to the outside via an easily accessible pin headers. This might be a more reliable solution than the extension board required for the micro:bit. On the other hand, the community and its comments on the Internet are much more for micro:bit than for other MicroPython-based educational boards.

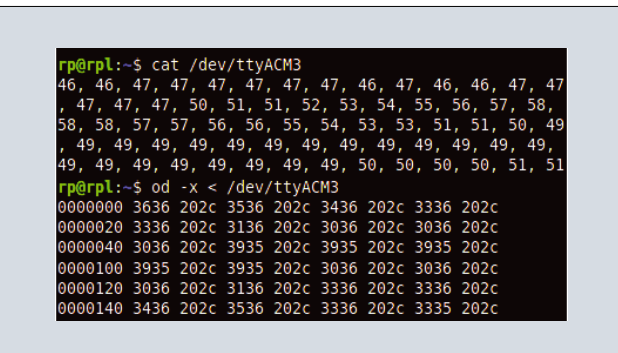
Conclusions

The BBC micro:bit offers an introduction to microcontroller programming and teaches how to use modern high-level languages like Python.

However, the MicroPython interpreter eats up memory and impacts speed. If you are looking to push a microcontroller to its limits, Arduino and its programming environment are a better option. ■■■

Info

- [1] BBC micro:bit: <https://www.microbit.org>
- [2] BBC micro:bit Go Bundle: <https://www.adafruit.com/product/3362>
- [3] Arduino: <https://www.arduino.cc>
- [4] Mu editor: <https://codewith.mu>
- [5] MicroPython documentation: <https://microbit-micropython.readthedocs.io/en/latest/index.html>
- [6] Calliope: <https://calliope.cc/en/idee/ueber-mini>



```

rp@rpl:~$ cat /dev/ttyACM3
46, 46, 47, 47, 47, 47, 47, 47, 46, 47, 46, 46, 47, 47,
47, 47, 47, 50, 51, 51, 52, 53, 54, 55, 56, 57, 58,
58, 58, 57, 57, 56, 56, 55, 54, 53, 53, 51, 51, 50, 49,
49, 49, 49, 49, 49, 49, 49, 49, 49, 49, 49, 49,
49, 49, 49, 49, 49, 49, 49, 50, 50, 50, 50, 51, 51
rp@rpl:~$ od -x < /dev/ttyACM3
0000000 3636 202c 3536 202c 3436 202c 3336 202c
0000020 3336 202c 3136 202c 3036 202c 3036 202c
0000040 3036 202c 3935 202c 3935 202c 3935 202c
0000100 3935 202c 3935 202c 3036 202c 3036 202c
0000120 3036 202c 3136 202c 3336 202c 3336 202c
0000140 3436 202c 3536 202c 3336 202c 3335 202c

```

Figure 6: Data output produced by the micro:bit in a terminal window.

Listing 4: Redirecting Output to the Terminal

```

$ cat /dev/ttyACM3
$ od -x < /dev/ttyACM3

```

Listing 5: Sample Hex File

```

:020000040000FA
:1000000000400020218E01005D8E01005F8E010006
:1000100000000000000000000000000000000000E0
[...]
```

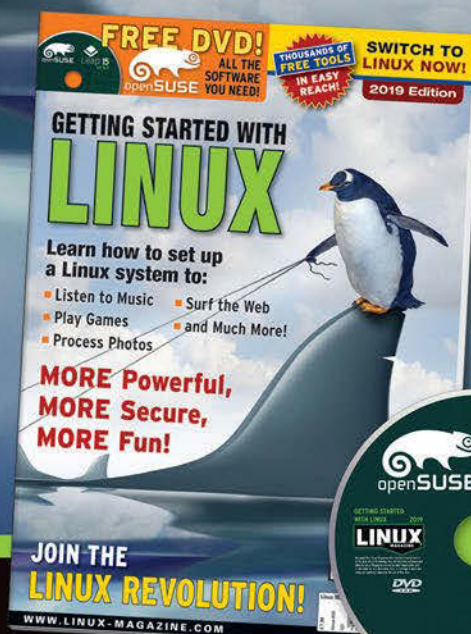

Harness the power of Linux!

Want your friends and colleagues to make the switch to Linux?

This single issue shows beginners how to:

- install Linux
- download and install free software for your Linux system
- play games
- create documents and spreadsheets
- process photos
- play music and videos
- and much more!

GETTING STARTED WITH **LINUX**



ORDER ONLINE:
shop.linuxnewmedia.com/specials



MakerSpace

Audio Builders Workshop
teaches soldering basics

Soldering 101

When it comes to DIY maker projects, if you don't know how to solder, your options are limited. Audio Builders Workshop remedies this with two kits, plus workshops, to give you hands on soldering experience. *By Bruce Byfield*

In the maker movement, the ability to solder is one of the great dividing lines – rather like compiling your own kernel in Linux programming. Just as a programmer who has not compiled a kernel is often limited, unless you know how to solder, the work you can do is limited, as well as the do-it-yourself (DIY) kits you can assemble. To help bridge this divide, Audio Builders Workshop (ABW) [1] has developed two kits to teach soldering basics: a metronome that emits a regular number of beats per minute to mark the musical frequency of sound, and a low pass filter that controls sound levels. Armed with these kits, ABW has held soldering workshops around Boston. In the last year or so, it has also started making the kit available to other interested groups and holding workshops from Anaheim to Frankfurt.

ABW is a special interest group of musicians, educators, and tinkerers in the Boston chapter of the Audio Engineering Society (AES), sponsored by Analog Devices, Inc. and Mouser Electronics. ABW was founded by Owen Curtain, who explains, “I held the first Audio Builders Workshop as a way to learn more about product development. When 50 people showed, I decided to continue with more events. Within a

year, we had held lectures on operational amps, a Compressor Hackathon, and a DSP lecture series. With small groups, we also built microphones, DI boxes, and microphone preamps.” This work is carried on by a small group of regulars plus volunteers recruited for specific events.

Brewster LaMacchia, a core ABW participant, notes that the group has “been focused on DIY rather than specific open source thinking. While the source of the hardware is there, I don't think there's an open hardware license on most of the commercial DIY company offerings.” However, for all practical purposes and by any definition, the kits can be described as open hardware even without the formality of a license. Moreover, the build instructions are released under a Creative Commons Attribution-ShareAlike 4.0 International (CC BY-SA 4.0) license.

According to LaMacchia, the kits arose out of the realization when holding workshops (Figure 1) that “many people were interested but had never soldered before and were hesitant to spend a few hundred dollars on something that looked overwhelming to someone who's never touched a soldering iron. We thought a low cost kit would be a good way to get started, but most of those are cheap kits with very



Figure 1: One of ABW's soldering workshops held in the Boston area.

limited build information. So the idea became 'let's offer a low cost kit with extensive build documentation and hold group learn-to-solder events to introduce people to building their own audio gear'."

Chris Kinkaid, another ABW regular, adds, "We've seen how excited first-time DIYers get about building audio gear. This year ABW has been invited to speak and workshop at half a dozen different events, but it's been tough because this is a labor of love and we all have day jobs as engineers and teachers. But we are currently navigating how we can take on more of these events and grow the mission of ABW. As educators and engineers, we know how empowering it is to understand how the tools we use every day work, and we want to help others in that discovery."

The Kits

Both of ABW's current kits (Figure 2) are musically oriented, because many AES members are either musicians or in-

involved in music production. "They're probably more fun than blinky-light ones," LaMacchia adds, referring to similar kits on the market. He goes on to explain that the metronome is built using 555-based circuits (one with three 5K Ohm resistors), a common choice for hundreds of beginner projects. The metronome includes a line-out jack, so it can be used with other equipment, while the low pass filter can be built into a guitar pedal. Neither kit includes a case, although one could easily be made with a 3D printer or by following the build instructions and modifying a standard plastic case.

Besides the kits, users will need a 9V battery, and a soldering kit that includes a soldering iron, solder, wire strippers, needle nose pliers, and wire cutters. The build instructions also suggest that users read some preliminary online instructions on the basics of soldering, such as MightyOhm's solder comic [2]. The kits are designed for users to gain experience soldering,

rather than to teach them from the very beginning. However, such preliminaries would presumably be covered in a workshop, and the build instructions include useful items such as basic safety precautions. Slightly more advanced topics, such as reading resistor color codes and how to orient capacitors and diodes, are given in the build instructions [3]. To further simplify the process, the kits' circuit boards are labeled to reduce the chance of error.

The build instructions are similar to the highest level of verbosity in some command-line tools: likely to be annoying to experts, but full of valuable information for those who need it. No assumption of previous expertise is made, so the result is a primer on basic electronic concepts. The primer is enhanced by high-resolution images of resistors and other hardware and sharp close-up images of routine tasks like bending leads with needle nose pliers.

After this basic information, the build instructions get down to assembly, starting with an overview, and giving hints that can help orient users as they work. The instructions close with detailed troubleshooting suggestions. The result is some of the best technical documentation I have seen in 25 years of writing instructions. Moreover, in the unlikely event that a solo user needs more help, they can access the notes for workshop leaders [4] or ask for help on Facebook [5].

LaMacchia estimates that an expert could assemble one of the kits in 20 minutes, largely because of the labels on the PCBs. However, he adds, "we find that most people who have never touched a soldering iron leave with a working metronome in 1.5 to 2 hours. The low pass filter takes slightly longer."

More importantly, of course, those who complete a kit (Figure 3) leave with soldering experience. "In terms of feed-

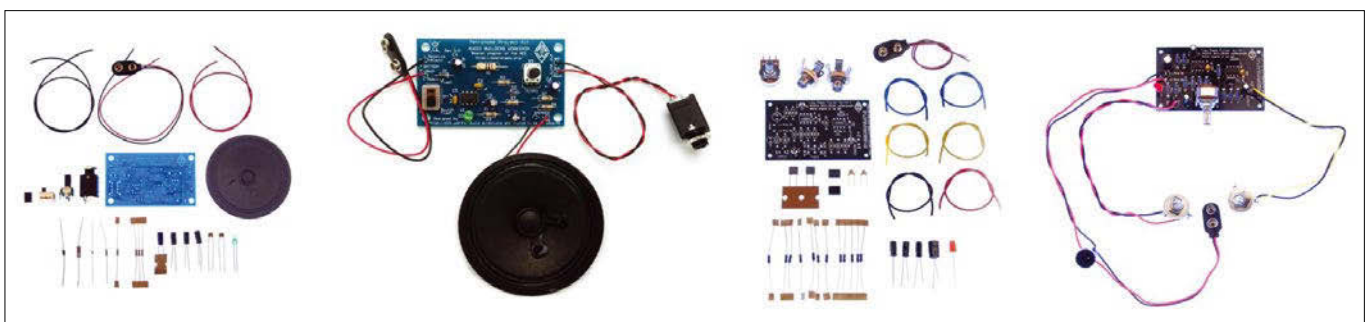


Figure 2: The two kits laid out: the metronome (left) and the low pass filter (right).

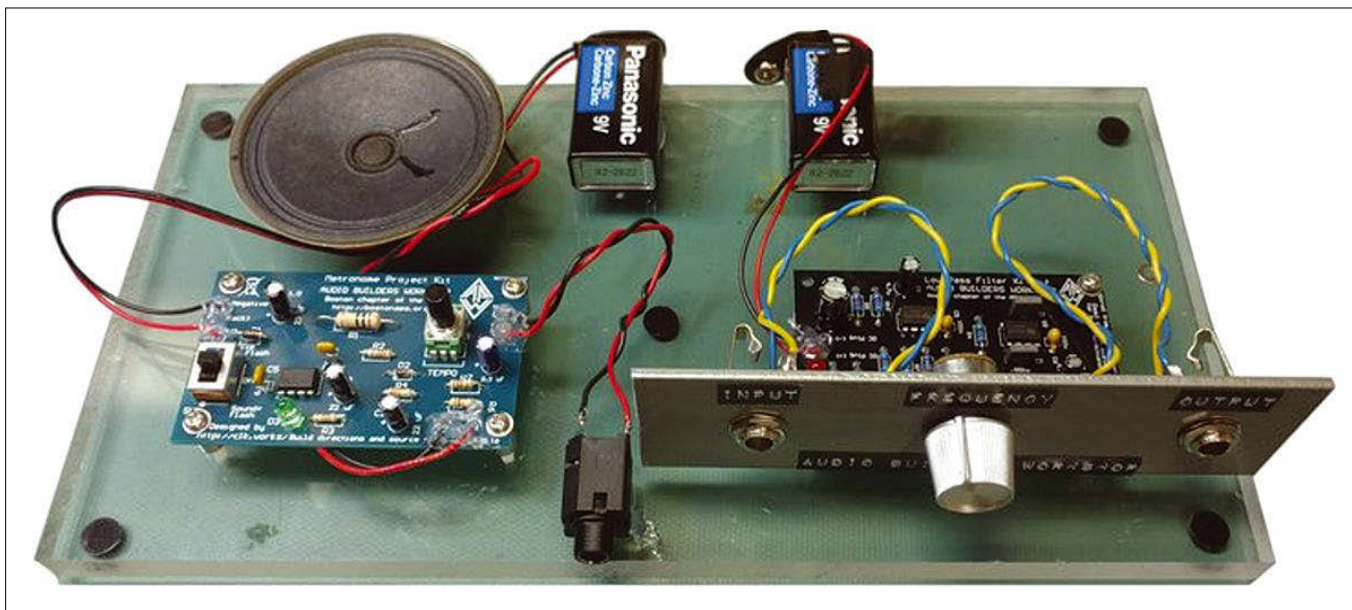


Figure 3: The two assembled kits mounted on a plastic board. The metronome is on the left.

back,” LaMacchia says, “it’s been absolutely fantastic. We have had so many people walk in with a ‘I’m not sure I can do this’ outlook and when they hear that tick-tick-tick of the metronome are literally jumping up and down saying, ‘I made this and it works!’ Particularly in the groups, there’s a lot of high-fiving and selfies with working hardware going on. As an instructor at these events, it’s really rewarding.”

Future Plans

Having created the kits and workshops, ABW is starting to distribute both more widely. “We now have a group of volunteers in New York City that are working to replicate what we’ve done here in Boston,” LaMacchia says. “They held one even this fall and more are in the works. We’re also reaching out to other AES sections, as well as educators.” One especially successful off-shoot is Pathfinders Outdoors, run by Buddy Lee Dobberten, which runs build workshops for veterans with PTSD [6].

In addition, after a token crowdfunding campaign, the kits are now being sold on Crowd Supply [7], a crowd-

funding site that specializes in open hardware.

In 2019, ABW has spent most of its time developing the kits. In the near future, it hopes to return to a breadboard kit it developed in 2018. The idea is to have a kit full of audio parts that people could use as they see fit – “something like the old Radio Shack 100-in-1 kit, brought in to the open source/sharing age,” says LaMacchia. “We hope to get back to that project, as well as some other ideas that we have for related low-cost ways to introduce people to DIY audio electronics. In the longer term, we want to have some offerings for digital signaling processing. That topic is more complex, but we’ve had a lot of interest from the AES members about some software equivalent to the ‘learn to solder concept’ – that is, a way to ease people with no background in to it.”

Perhaps ABW could consider the option of shipping its kits with soldering kits and cases to offer more complete experiences to users – and to spare them the annoyance of having to wait while they order additional supplies. However, so far as the goal of teaching soldering,

ABW succeeds outstandingly, providing a detailed, hands on experience with concrete results. It’s the sort of knowledge that both makers and supporters of open hardware need for their movements to succeed. ■■■

Info

- [1] Audio Builders Workshop: <https://www.audiobuildersworkshop.com/>
- [2] Solder comic: https://mightyohm.com/files/soldercomic/FullSolderComic_EN.pdf
- [3] Build Instructions: <https://www.audiobuildersworkshop.com/builds>
- [4] Workshop Leader instructions: https://drive.google.com/drive/folders/1L2jAS1JjbPwS_V37auWXbAwm-jV8qoW5y
- [5] Facebook: <https://www.facebook.com/groups/AudioBuildersWorkshop/>
- [6] Pathfinders Outdoors: <https://www.pathfindershism.org>
- [7] Crowd Supply page: <https://www.crowdsupply.com/audio-builders-workshop/abw-metronome-and-low-pass-filter-kits>

If you're a gamer, your desktop can get very complicated – with dozens of games from multiple platforms scattered about in different menus. Wouldn't you rather gather everything up in a single place? GameHub is a promising tool that manages all your Steam, GOG, and Humble Bundle games from a single interface. This month we try out GameHub and study whether the promising concept of a universal game interface is ready for the world.

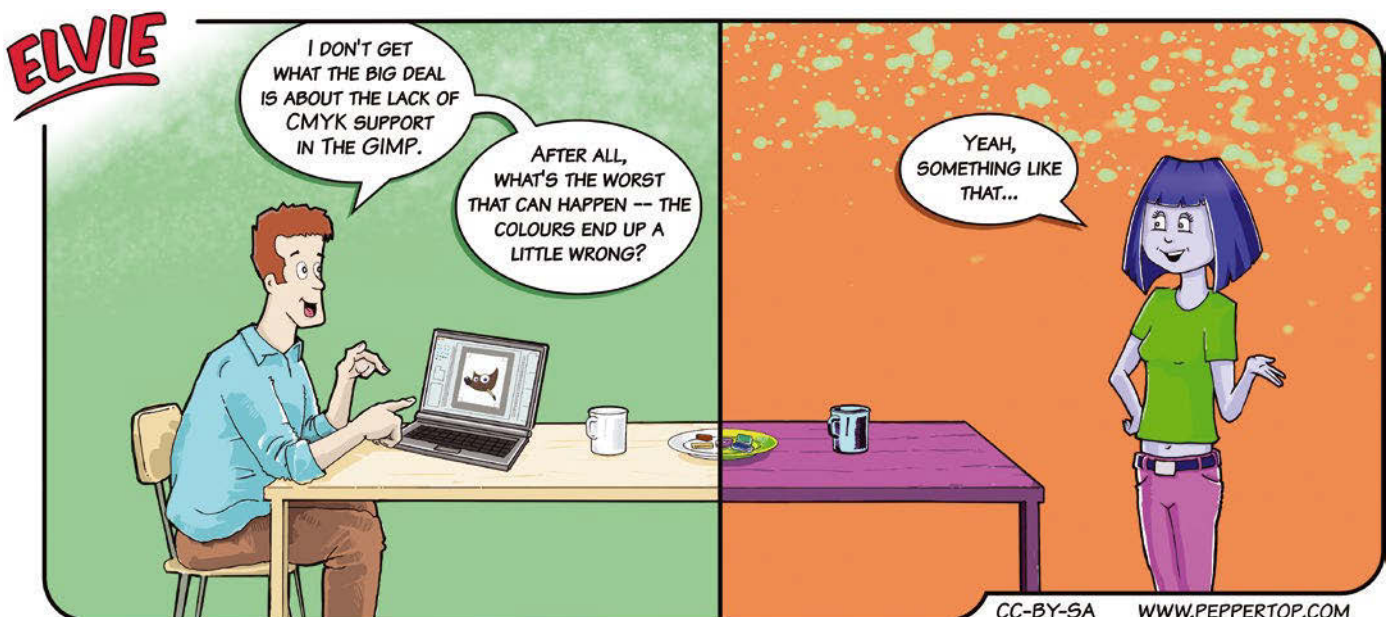
Also in this month's LinuxVoice, we show you how to set up a simple yet secure communication channel with a free tool called Dead Simple VPN, and our tutorial series continues with lessons on Bash Readline functions and the PeerTube video sharing platform.



Image © Olexandr Moroz, 123RF.com

LINUXVOICE ▶

Doghouse – RISC-V Summit	67
<i>Jon "maddog" Hall</i>	
While attending the second summit on RISC-V architecture, maddog was blown away by the level of openness and collaboration.	
GameHub	68
<i>Christoph Langner</i>	
Organize your games by bringing them all together into a single library.	
Dead Simple VPN	74
<i>Christoph Langner</i>	
With a single command, Dead Simple VPN builds a secure VPN connection.	
FOSSPicks	78
<i>Graham Morrison</i>	
Graham looks at TreeSheets, rare, McFly, b2, DrumGizmo, A/B Street, Xmonk, and much more!	
Tutorials – PeerTube	86
<i>Marco Fioretti</i>	
Self-host your videos without the limitations embedded in YouTube and similar platforms.	
Tutorials – Readline	92
<i>Paul Brown</i>	
Readline provides a rich set of tools for moving around quickly on the command line.	



CC-BY-SA WWW.PEPPERTOP.COM

Shop the Shop

shop.linuxnewmedia.com

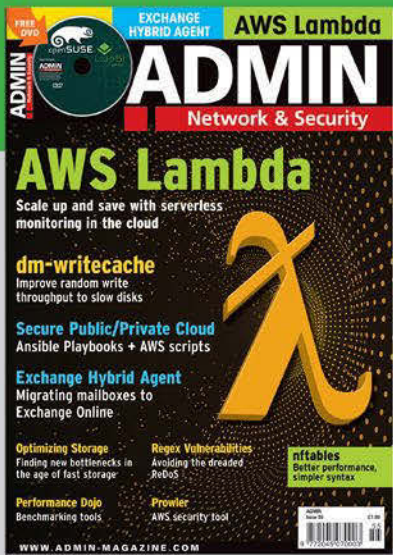
Want to subscribe?

Searching for that back issue you really wish you'd picked up at the newsstand?

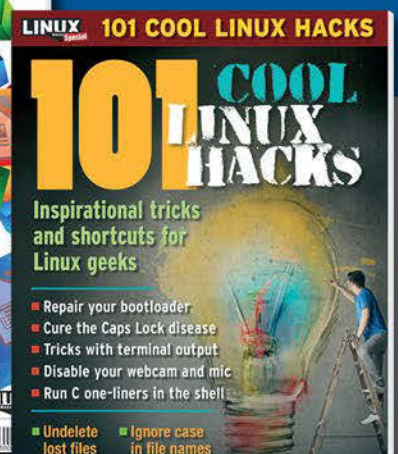
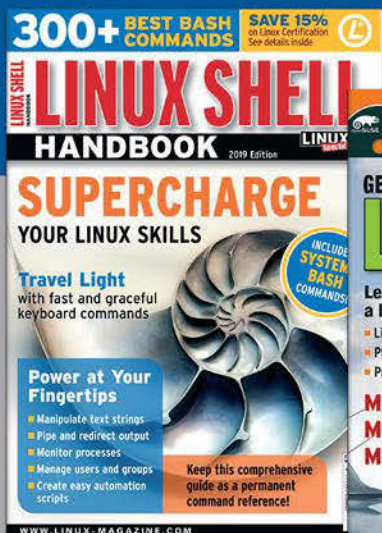
Discover the past and invest in a new year of IT solutions at Linux New Media's online store.

▶▶ shop.linuxnewmedia.com ◀◀

DIGITAL & PRINT SUBSCRIPTIONS



SPECIAL EDITIONS



MADDOG'S DOGHOUSE

While attending the second summit on RISC-V architecture, maddog was blown away by the level of openness and collaboration. BY JON "MADDOG" HALL

Views on the 2019 RISC-V Summit

About a year ago, I wrote an article on the RISC-V architecture and how it seemed to be interesting. I also pointed out that it seemed to be moving along very fast, but there were naysayers who were questioning whether it would get off the ground. I advised the naysayers to wait a year before offering their opinions.

On December 9-12, 2019, I attended the second RISC-V Summit in San Jose, California. There were a little under 3,000 attendees, a few of whom I had worked with over the years at Digital Equipment Corporation, on the Linux kernel project, or other similar projects and companies.

Quite frankly, I was blown away.

While there is much that still has to be done, there was a lot finished including some basic chips produced that were running Linux. There was much additional work being done on software emulators and FPGAs, which created excitement for me. You had chip designers designing extensions and passing those off to operating system and compiler engineers, who would then try out the design with the compilers and operating systems to see if the extensions were useful. This was collaboration that I had never experienced on this level, at this speed, and with this much openness before.

Everyone was talking about being "open," and stressing the sharing of technical information back and forth. It was exciting ... electrifying.

For those of you who are not familiar with RISC-V, it is a movement to have an "open" Reduced Instruction Set Computer (RISC) that was started at the University of California, Berkeley, the same place that the Berkeley Software Distribution (BSD) of Unix started. Trying to learn from the previous decades of instruction set design, they wanted a clean design that would be efficient, compact, extendable, and free of licensing fees to allow both research and development of new processors. Many companies, universities, and even individuals have joined this organization.

The first pleasant thing that happened was their CEO, Calista Redmond, gave the opening address. Not only was her address great, I was also happy to meet Redmond several times, and she was both welcoming and fun to talk with. I think she was an excellent choice for a CEO.

Krste Asanovic, one of the professors at UC Berkeley who started the RISC-V project, gave the "State of the Union" address after Redmond, and was followed by an old friend of mine, Martin Fink of Western Digital.

I met Martin years ago when he was working for Hewlett Packard. Martin, like myself, saw in GNU/Linux more than a "hobbyist toy," more than a "geek thing," but a fundamental building block of software development that would change the computing industry forever.

Martin wrote the first book on Open Source business plans, *The Business and Economics of Linux and Open Source*, which was originally published in 2002 and is still considered a classic in the field.

Today Martin is the advisor to the CEO of Western Digital, a very large data storage company, and is focused on "data driven computing." According to Martin, we put too much focus on "processing" and CPUs and not enough focus on the data. Because of this, Western Digital has created a new way of giving access to many different types of computational power that they call OmniXtend™, a cache-coherent fabric [1]. Martin almost shook with glee as he unveiled a system based on this technology.

With that as a start, the conference only got better, as more and more people stood up to talk about how RISC-V technologies and collaboration were going to help solve some of the biggest problems of computing today:

- Secure systems
- Verifiable code
- Efficient power systems to help with energy and cooling savings

I was also happy to see some old friends, such as Keith Packard [2], who I first met when he was working on the X Window System, and, who later worked near me as we wove our way through various companies and GNU/Linux. Keith has been a long-time contributor to Debian, and, he now works for SiFive [3] on their contributions to the RISC-V community.

RISC-V is an Instruction Set Architecture (ISA) that allows a very small "core" set of instructions, purposely kept simple and small so operating systems like GNU/Linux will always be able to support it, but allowing extensions so special-purpose distributions can take advantage of those extensions if they so desire.

I look forward to the next RISC-V Summit, probably next December. In the mean time, if you would like to get started learning about RISC-V, the slides and videos from the RISC-V 2019 Summit are available online [4].

Carpe Diem! ■■■

Info

- [1] OmniXtend: <http://blog.westerndigital.com/omnixtend-fabric-innovation-with-risc-v/>
- [2] Keith Packard: http://en.wikipedia.org/wiki/Keith_Packard
- [3] SiFive: <http://www.sifive.com/>
- [4] RISC-V 2019 Summit Proceedings: <https://riscv.org/2019/12/risc-v-summit-2019-proceedings/>

GameHub displays all your games in a single interface

Organized Games

If you regularly buy games through Steam, GOG, and Humble Bundle, GameHub can help you keep them organized by bringing them all together into a single library.

BY CHRISTOPH LANGNER

Most computer owners have long since stopped buying games in a cardboard box with floppy disks, CDs, or a DVD including a booklet and other goodies. Usually you log on to an online sales platform for games, such as Steam, acquire a license via the portal, and then download the game off the Internet. These portals are very popular, despite the drawbacks. For example, Steam has for years prevented honestly bought Steam games from being resold as used games, as you could do with a game purchased in a box – although that policy has recently been challenged in the EU [1].

Another disadvantage with purchasing games online is the difficulty of keeping track of all the titles in your collection. In addition to Steam [2], there is GOG (formerly Good Old Games) [3], and the Humble platform [4], which in the past has enjoyed massive success with cheap Humble Bundles advertised on social media channels. Both alternatives offer the advantage that, unlike Steam, they do without DRM measures. However, if you buy your games over the various platforms, you have to check in to the individual portals time and time again to install them.

One Front End for All

GameHub is open source software [5] that integrates the three large gaming portals in a single interface. You can also use it to organize manually installed games, as well as old Windows games that can be run in Wine (see the “Proton and Wine” box) and games designed for a variety of old platforms (see the “Emulators” box).

You only need to install the proprietary client program for Steam; for GOG and Humble, you just need to enter your account data in GameHub. On

Fedora and OpenMandriva, you can install directly from the package sources in the *gamehub* package. For openSUSE users, there is a package in the Open Build Service [6]. Users with Arch Linux can import the application from the AUR with a one-liner, assuming an AUR helper such as Pamac or Yay is available. For Ubuntu, you install via a PPA package source that offers packages for Ubuntu 16.04 to 19.10 (Listing 1).

After the installation, a new *GameHub* entry appears in the application menu of the desktop environment. When launched, the program first shows you a setup wizard, which you can use to set up your accounts for Steam, GOG, and Humble Bundle. For GOG and Humble, you just enter your password and email address; for Steam, the official client has to be installed on the system. For current editions of Ubuntu, for example, the Steam client can be installed directly using the package manager. The Ubuntu Software Center lists Steam as the *Steam installer*.

After installing Steam, GameHub automatically connects to the platform. However, to list the Steam games, you either have to publish your game list or create a Steam API key. To do this, click on *Settings* in GameHub and select *Steam* as the game source in the configuration. Pressing *Generate Key* opens the Steam client with the configuration dialog. Then transfer the key to the *Steam API Key* field and restart the GameHub program.

All Games at a Glance

GameHub now displays all games purchased via the three portals (Figure 1). However, the orange header still indicates that the program requires an API key for the Internet Game Database (IGDB) [7] for detailed information on the individual titles. This can be created quite easily and free of charge. Pressing the *Settings* button opens the *Appearance* tab in the GameHub configuration. When you get there the *Generate key* button in the *IGDB* section takes you to the portal website, where you may have to create an account. You

Listing 1: Installing GameHub for Ubuntu

```
$ sudo add-apt-repository ppa:tkashkin/gamehub
$ sudo apt update
$ sudo apt install com.github.tkashkin.gamehub
```

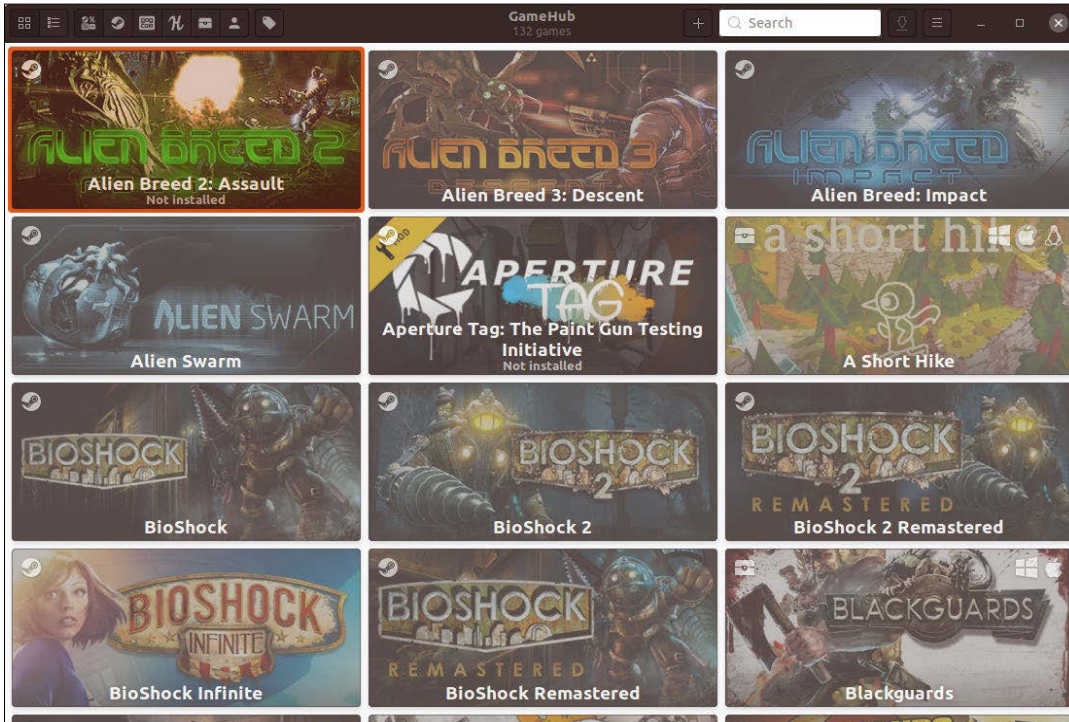


Figure 1: GameHub integrates games from the major games platforms (Steam, GOG, and Humble, as well as manually installed games) in a clear-cut interface.

can then copy and paste the key to GameHub and restart the program again.

Like almost all modern Gnome applications, GameHub moves the icons to the window bar. You can use the buttons to switch between a tile and a list view or filter the displayed games by platform and various tags. To the right of the application name, there is the possibility to manually enter

games in GameHub, launch a search in the games on offer via a search field, and open the settings. You can change the installation paths, for example. By default, GameHub stores the downloaded data in `~/Games/_Collection/` (Figure 2) and the installed games in `~/Games/<Provider>/`.

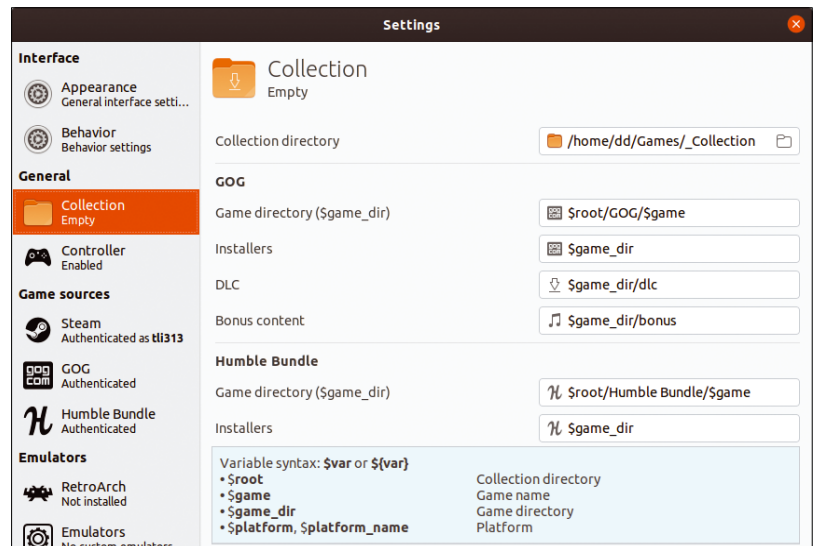
For titles from the Steam and GOG libraries, GameHub automatically loads matching images from the Internet; for Humble Bundle games, you only see an icon. However, a real image can be downloaded off the web. To do this, right-click on the desired title and open the *Details* of the game. A click on the download icon in the top right corner of the preview image opens a dialog that searches for suitable images on the Internet and then offers

Figure 2: In the GameHub settings, you can define the installation paths for games on different platforms.

Proton and Wine

Wine, as well as the Valve-driven Proton fork of Wine, can also be used and installed directly via GameHub – at least on Ubuntu. The options can be found in *Settings | Steam*, where the program lists the available Proton versions. A click on *Install* then installs the desired variant. This option is not available on Arch Linux, but GameHub automatically detects the Wine and Proton libraries installed via the Steam client.

After completing these steps, you can also install and start “Windows-only” games via GameHub. The program changes the context menu from *Execute* to *Execute with compatibility layer* and then shows an additional dialog at startup (e.g., by setting the Proton version or environment variables). How well the desired game works with Wine or Proton does not depend on GameHub, but on the compatibility layer you use and on the Windows game itself.



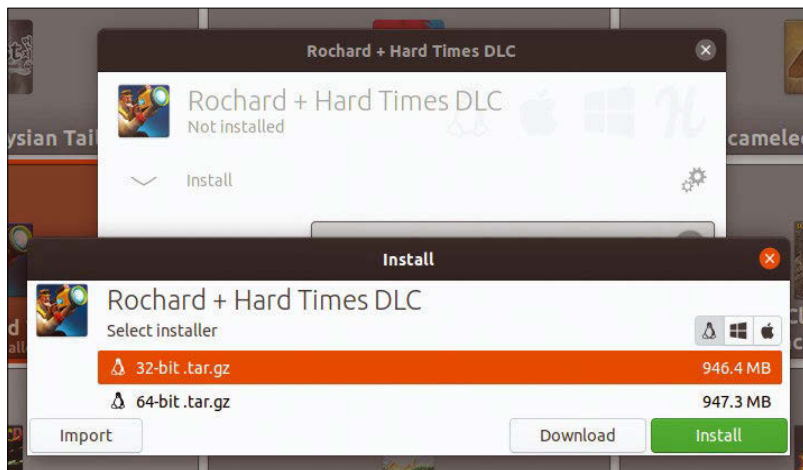


Figure 3: A few mouse clicks are all it takes to install most titles.

them for selection. Pressing the *Install* button lets you install the program on your computer (Figure 3).

Installation Pitfalls

Some games are easier to install than others. The easiest setup is installing Steam titles: All you have to do is click on the game and the Steam client will open and offer to install the selected game. Click on *FINISH* and the dialog closes, while Steam downloads the game off the web in the background and sets it up on your hard disk. If required, the *Manage Downloads* link opens the Steam client’s download manager where you can track the progress. When the installation is complete, the game tile should lose its grey veil and report *Installed*.

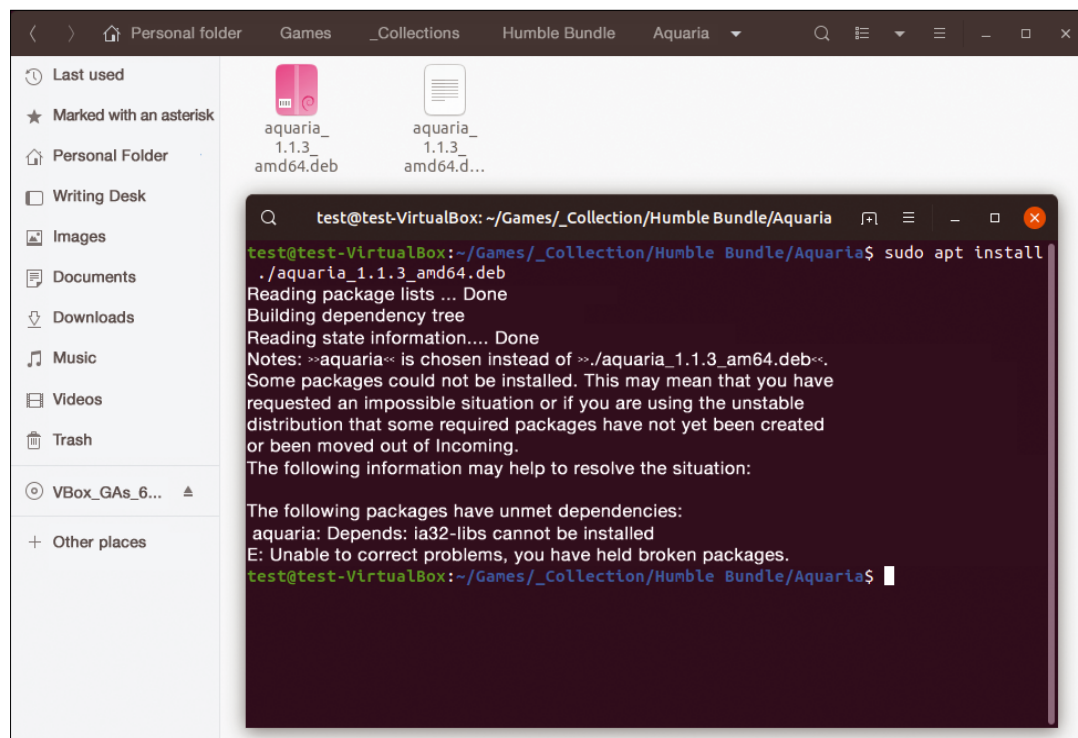
Control by Gamepad

GameHub can be completely controlled by a gamepad if required. The program automatically detects compatible devices and integrates them into the program. The detected devices can be managed in the *Settings* below *Controller*. As soon as a gamepad is active, GameHub displays the key assignment (such as *A* for *Select*, *B* for *Back*, or *X* for the menu) in the window bar. In practice, however, the function was still a little shaky. On an Arch Linux system, the connected Logitech Wingman Rumblepad actually crashed the program. On a system with Ubuntu 19.04, GameHub was stable even with a gamepad, but GameHub only registered the signals from the analog sticks. All the keys remained inoperative.

Click on the tile again to start the game. (If you like, you can use a gamepad for this and other processes within GameHub. See the “Control by Gamepad” box.)

Installing GOG games involves more work. A dialog appears here supporting installation in different languages depending on the title. A click on *Install* downloads the game from the portal archive and installs it on the hard disk using the installation routine integrated in the game. In our lab, this worked without any problems with all the titles

Figure 4: When attempting to install the game Aquaria’s DEB package for 64-bit systems, the Ubuntu Software Center failed without comment. Setting up a terminal window revealed the reason: Current Ubuntu systems no longer support *ia32-libs*.



tested. Like with Steam, you can call up the newly added game by clicking on the tile.

In contrast, complications regularly occurred with games from Humble Bundle. Even choosing the version of a game to install often caused trouble. If you use a distribution like Arch Linux, which is not based on RPM or DEB packages, the package formats are not available. A statically built variant as a `tar.gz` archive is not always available for 32- and 64-bit systems. So not every game can be installed easily via the GameHub interface.

Complications with Humble

Even a suitable version does not mean that the installation will proceed smoothly. Some games try to load `ia32-libs` from the package manager during installation. This compatibility layer for the operation of 32-bit programs on a 64-bit system has not existed for many generations of Ubuntu.

However, Ubuntu's Software Center does not send the associated error message to the interface during installation. You only see that the installation has failed, but not why. The problem can only be determined at the command line (Figure 4) when manually installing the package via:

```
sudo apt install ./File.deb
```

With other titles – especially those you install from a tarball – the installation works, but GameHub afterwards reports that it cannot find the executable file. In this case, you have to right-click on the game's tile to open its *Properties* (Figure 5).

In the dialog box, click on the folder icon to the right of the *Executable file* field, which will open a file manager with the folder of the current game as its content. Select the executable file and close the dialog again with *Apply*. Then, as long as there are no further complications, the game can be started without any problems.

Info

- [1] French court case on Steam and used games: <https://www.polygon.com/2019/9/19/20874384/french-court-steam-valve-used-games-eu-law>
- [2] Steam: <https://store.steampowered.com/>
- [3] GOG: <https://www.gog.com>
- [4] Humble: <https://www.humblebundle.com>
- [5] GameHub: <https://github.com/tkashkin/GameHub>
- [6] Open Build Service for openSUSE: <https://build.opensuse.org/package/show/home:lewellyn:gamehub/gamehub>
- [7] IGDB: <https://www.igdb.com>
- [8] RetroArch: <https://www.retroarch.com>

Emulators

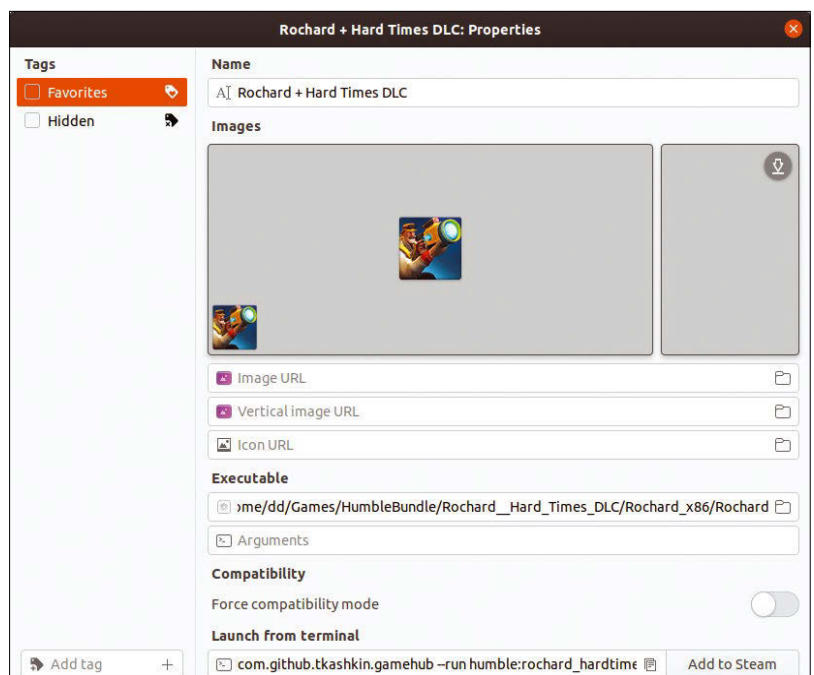
In addition to classic PC games, GameHub also integrates emulator games on request. The system uses RetroArch [8], a front end for numerous emulators from Amstrad CPC to classics like the C64 and the Nintendo 64 game console through to the ZX Spectrum. Alternatively, you can also set up your own emulators. The configurations are listed in *Settings* under *RetroArch* and *Emulators*.

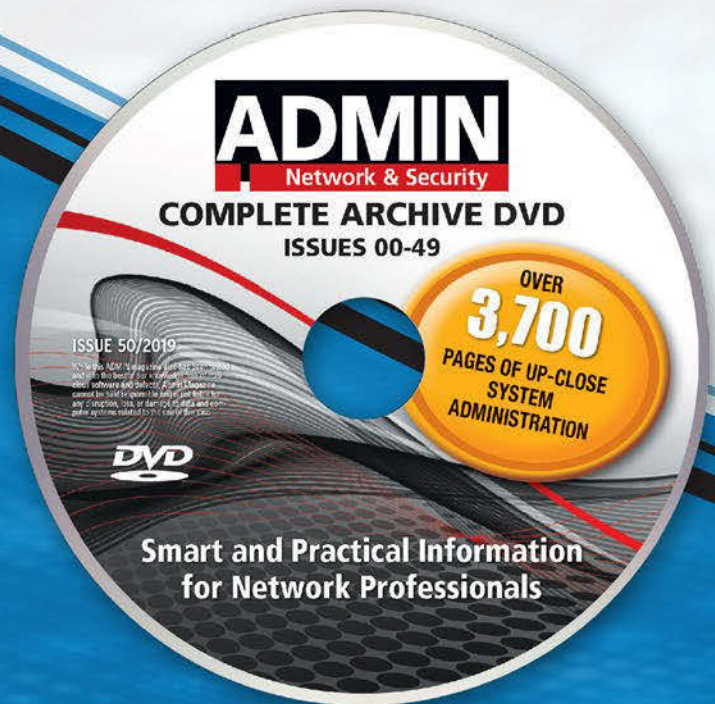
Conclusions

With its strategy of combining all gaming platforms and the games you installed yourself in a single interface, GameHub definitely grants the wish of many gamers for a better overview of their game collections. No matter whether Steam, GOG, or Humble, thanks to GameHub you can download the titles you purchased from the platforms off the web with just a few clicks and proceed to install them on your computer. With a little manual work GameHub becomes a pleasant and functional game library.

In practical terms, the feature for installing games proves to be particularly immature, although this cannot necessarily be attributed to GameHub. If a game is delivered as a DEB package, and the system's package manager fails without an error message because of the installation, then the blame lies firmly with package management. A similar problem arose in our lab directly in GameHub, where the 32-bit libraries required for starting a title imported via `tar.gz` archive were missing. Nevertheless, GameHub proves to be a useful addition to the gamer's desktop. ■■■

Figure 5: Especially when installing games via a `tar.gz` archive, GameHub does not automatically find the executable file. This setting often has to be configured manually, although this is not usually difficult.





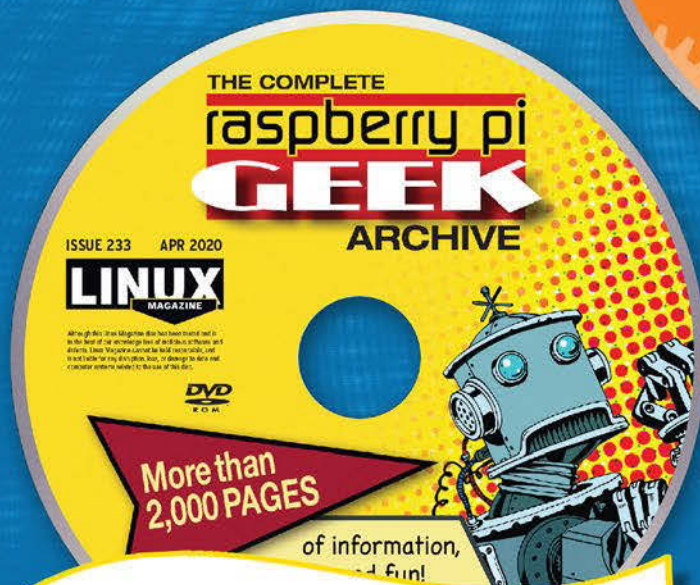
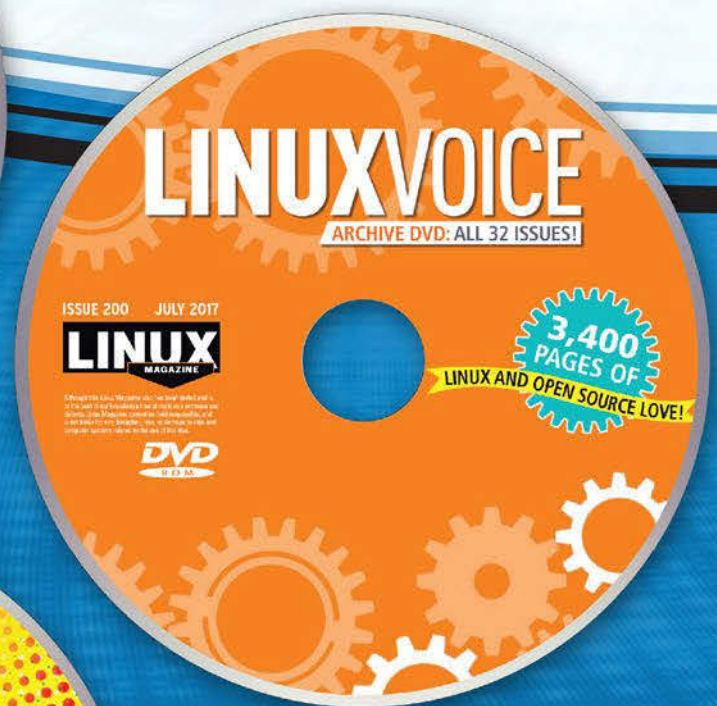
Complete Your Open Source Library with Archive DVDs!

Each fully-searchable archive DVD includes past articles so you can find the content you need quickly.

Save hundreds off the print and digital copy rate with a convenient archive DVD!

Order Your DVD Now!

shop.linuxnewmedia.com



Coming soon!

Building a secure, simple VPN connection

Keep it Simple

With a single command, Dead Simple VPN builds a secure VPN connection.

BY CHRISTOPH LANGNER

If you are on a business trip and need to transfer data over an open hotel WLAN or use an unencrypted WiFi connection at a conference, you need to establish a secure tunnel to the Internet or a secure connection to your home network to protect your data.

The classic approach for this protection is a virtual private network (VPN). A VPN drills a virtual tunnel from the network you are currently using on your computer through the Internet to a trusted server on a trusted network, keeping third parties from accessing the transferred data.

OpenVPN and the more modern WireGuard [1] are classic tools for setting up a VPN. In practice, however, these tools are not exactly easy to use. Dead Simple VPN (DSVPN) [2] offers a simple solution: After installation, a single command is all it takes to establish the VPN connection.

Installation

With the exception of Arch Linux, DSVPN currently is not available in the package sources of popular distributions. Arch Linux users can install DSVPN with the AUR helper Yay by typing:

Listing 1: Installing DSVPN

```
$ sudo apt install git
$ git clone https://github.com/jedisctl/dsvpn.git
$ cd dsvpn
$ make
$ sudo make install
$ dsvpn --help
DSVPN 0.1.3 usage:

dsvpn "server"
    <key file>
    <vpn server ip or name>|"auto"
    <vpn server port>|"auto"
    <tun interface>|"auto"
    <local tun ip>|"auto"
    <remote tun ip>|"auto"
    <external ip>|"auto"

dsvpn "client"
    <key file>
    <vpn server ip or name>
    <vpn server port>|"auto"
    <tun interface>|"auto"
    <local tun ip>|"auto"
    <remote tun ip>|"auto"
    <gateway ip>"auto"

[...]
```

```
yay -S dsvpn
```

For other distributions, you must compile the application from source code. This sounds more difficult than it is in practice. Listing 1 demonstrates the procedure on a freshly installed Ubuntu 19.04. Listing 1 executes the VPN program with `dsvpn --help` and displays the individual parameters (Figure 1).

The `make install` command installs DSVPN, working around the package manager so that it does not appear in the typical package management tools. To cleanly remove the program from the system, either run `sudo make uninstall` in the source directory or delete the `/usr/local/sbin/dsvpn` file with root privileges. No other program files or directories are created during the installation. (See the “Raspberry Pi” box for installation on Raspberry Pi.)

Configuration

To establish a connection, you first need a key. DSVPN does not distinguish between private and public keys. To create the `vpn.key` key file in the

```
test@test-VirtualBox: ~/dsvpn
test@test-VirtualBox:~/dsvpn$ dsvpn --help
DSVPN 0.1.3 usage:

dsvpn "server"
    <key file>
    <vpn server ip or name>|"auto"
    <vpn server port>|"auto"
    <tun interface>|"auto"
    <local tun ip>|"auto"
    <remote tun ip>|"auto"
    <external ip>|"auto"

dsvpn "client"
    <key file>
    <vpn server ip or name>
    <vpn server port>|"auto"
    <tun interface>|"auto"
    <local tun ip>|"auto"
    <remote tun ip>|"auto"
    <gateway ip>"auto"

test@test-VirtualBox:~/dsvpn$
```

Figure 1: DSVPN can be built in a few seconds directly with Ubuntu's on-board tools. After installation, use `help` for tips on using DSVPN.

Raspberry Pi

The Raspberry Pi's ARM CPU works with the Neon multimedia and signal processing extension. DSVPN supports this function, but you have to enable the Neon optimizations during the build with:

```
env OPTFLAGS=-mfpu=neon make
```

current directory using the random number generator `/dev/urandom` available on the system, use the following command:

```
$ dd if=/dev/urandom of=vpn.key \
count=1 bs=32
```

Later on, you will also need the key on the client computers that you want to connect to the server's DSVPN. Copy the key file to a USB stick.

You have now laid the foundation and completed the installation on the server. Repeat all of these steps on the client computer except for creating a key file. Instead, take the prepared USB stick and copy the `vpn.key` file to the client system. A `VPN/` subfolder in the home directory or the home folder itself is recommended as the storage location.

Next, call the VPN service on the server as shown in Listing 2. In the basic configuration, the

Forwarded

On a typical home network, a WiFi router ensures that the connected computers find their way to the Internet. However, the reverse route, from the Internet to a specific computer on the LAN, is not guaranteed: Unsolicited requests from the Internet are usually simply dropped by the WiFi router.

Therefore, for the VPN network, you need to set up port forwarding from the WiFi router to the computer equipped with DSVPN. The procedure is virtually the same for any vendor; in the following example, I use a Fritz!Box with the current firmware version 7.10. Open the administration back end in a web browser with the URL `http://fritz.box`. Then click on the *Port Sharing* tab via *Internet | Permit Access*.

Clicking on *Add Device for Sharing* launches a wizard that helps you with the subsequent configuration. First select the appropriate device, and then click the *New Sharing* button. In the dialog, change the selection to *Port Sharing* and enter the data as shown in Figure 2. Select the *Service Name*; for the port, you must use the port specified when calling `dsvpn`.

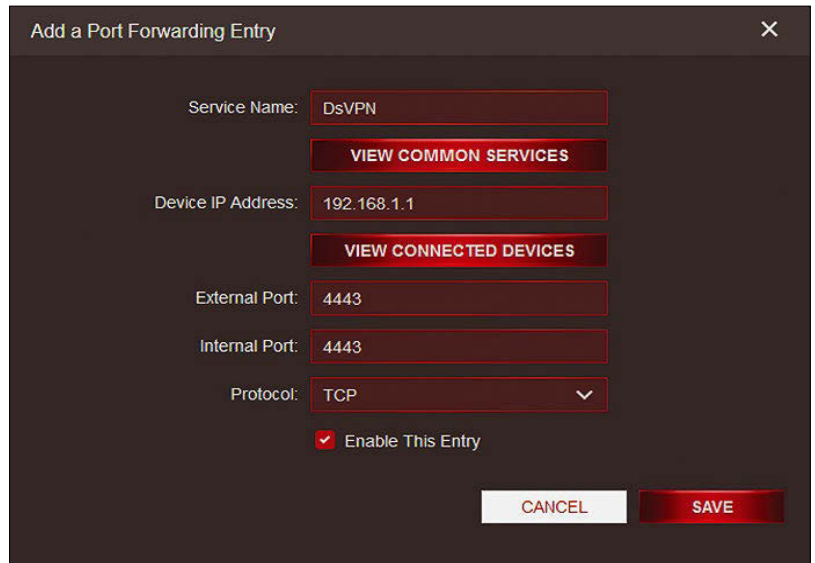


Figure 2: On a home or office network, VPN does not work without port forwarding from the router to the VPN server. Port 4443 is typical for a VPN, but if you want to break out of restrictive networks, port 443 is the best choice.

system listens on port 443. This port, which is actually used for HTTPS connections, usually also works with public hotspots (such as those at hotels or events).

However, since you need port forwarding from the router to the DSVPN server, network-attached storage (NAS) may cause you problems since it usually requires ports 80 and 443 for encrypted connections (see the "Forwarded" box). If necessary, extend the call, adding the desired port number (line 5).

Listing 2: Calling the VPN Service

```
01 $ sudo dsvpn server vpn.key auto
02 Interface: [tun0]
03 net.ipv4.ip_forward = 1
04 Listening to *:443
05 $ sudo dsvpn server vpn.key auto 4443
06 Interface: [tun0]
07 net.ipv4.ip_forward = 1
08 Listening to *:4443
```

Listing 3: Checking the Connection

```
$ ip addr show tun0
3: tun0: <POINTOPOINT,MULTICAST,NOARP,UP,LOWER_UP> mtu 9000 qdisc fq_
    codel state UNKNOWN group default qlen 500
    link/none
    inet 192.168.192.254 peer 192.168.192.1/32 scope global tun0
        valid_lft forever preferred_lft forever
    inet6 64:ff9b::c0a8:c0fe peer 64:ff9b::c0a8:c001/96 scope global
        valid_lft forever preferred_lft forever
    inet6 fe80::f7eb:4642:e501:e5f6/64 scope link stable-privacy
        valid_lft forever preferred_lft forever
```

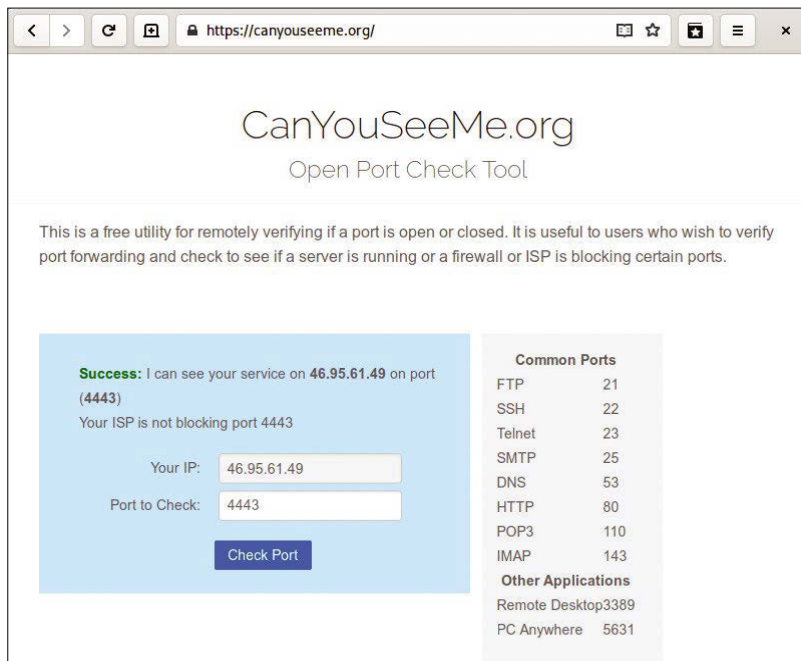


Figure 3: Use CanYouSeeMe.org to see if port forwarding works. Note that the DSVPN server must also be active.

Listing 4: Enabling a VPN Connection

```
$ sudo dsvpn client vpn.key <external IP> <4443>
```

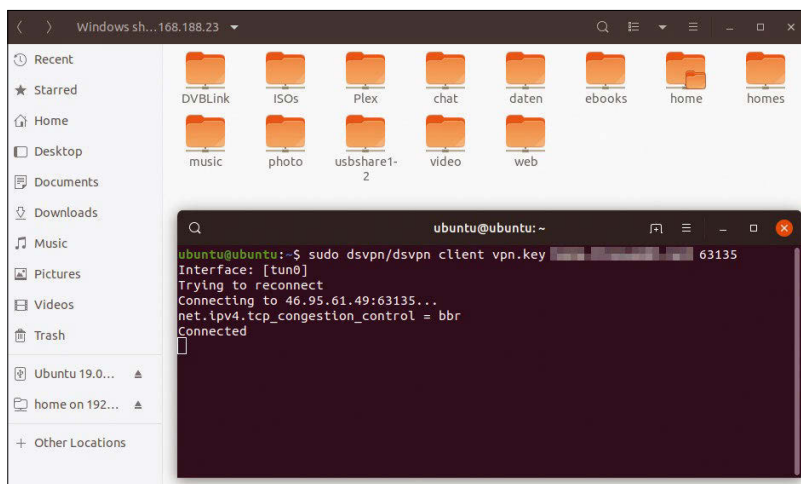
Opening a Connection

To check the connection, call

```
ip addr show tun0
```

The output from this call should show that there is now a new network device named tun0, as well as its IP address (Listing 3). You can check if port forwarding works with CanYouSeeMe.org (Figure 3). The page should automatically determine your external IP address, so you only need to correct the port number (4443 in this example). If the check returns Success, you can proceed to the next step.

Figure 4: An example of the benefits of a VPN connection: You can access the network shares on your home network via the hotspot provided by a smartphone on a mobile network.



For an initial test, use the command in Listing 4 to enable the VPN connection from a remote network to your home computer (e.g., via a hotspot served up by your mobile phone). You then should be able to use all of your home or office network’s resources immediately, even while you are on the move. This includes, for example, file sharing or network drives (Figure 4). The connection is maintained until you close DSVPN on the server or client with Ctrl+C.

If you want to use DSVPN regularly, it makes sense to get a DynDNS address and set up the service on the WiFi router. This means that instead of reaching your network via the Internet IP, which is constantly changing, you use an intuitive URL. Most router manufacturers integrate a corresponding function into the configuration interface; for a Fritz!Box, you will find the settings in *Internet | Permit Access | DynDNS*. Many DynDNS providers charge a monthly fee, but there are also dedicated services such as FreeDNS, which offers up to five hosts for free [3].

Automation

DSVPN remains true to its motto in many areas: There is neither a tool for creating the key file nor a user administration tool. The developer is also reluctant to integrate the program into the system as a service. However, his aversion to the init system does not mean that DSVPN cannot be managed with systemd: You just have to create the corresponding service files manually.

Based on templates by Greek developer Evaggelos Balaskas, this can be done quickly [4]. Under `/etc/systemd/system/` create the `dsvpn_server.service` and `dsvpn_client.service` files, and fill them with the content from Listing 5 (Server) and Listing 6 (Client). Don’t forget to adjust the path to the key file, the server IP address, and the WiFi router port number to the ports forwarded to the DSVPN server.

Next, you have to update the system configuration on both the client and the server with:

```
sudo systemctl daemon-reload
```

Listing 5: Server Service

```
[Unit]
Description=Dead Simple VPN - Server

[Service]
ExecStart=/usr/local/sbin/dsvpn server
/<root>/<vpn.key> auto <4443>
Restart=always
RestartSec=15

[Install]
WantedBy=network.target
```


Listing 6: Client Service

```
[Unit]
Description=Dead Simple VPN - Client

[Service]
ExecStart=/usr/local/sbin/dsvpn client
/<root>/<vpn.key> <IP-Address> <4443>

[Install]
WantedBy=network.target
```

Afterwards the connection can be enabled manually or automatically during the boot process (Listing 7). To disable the automatic start, replace `enable` in the last two commands with `disable`.

Firewall

For the greatest possible security, you must prevent data leaks from the VPN to the Internet. This requires a firewall that intercepts all data packets flowing past the VPN. See Balaskas' blog for a description of implementing such a firewall with `iptables` [5].

Conclusions

DSVPN fills a gap left open by other VPN solutions. For example, it works in isolated environments that only allow TCP/80 and TCP/443 ports to roam the Internet. WireGuard and other lean VPN solutions, such as Glorytun [6], also require UDP, since one TCP port alone is not enough. With DSVPN, the connection between the client and the server is established with a single command, while OpenVPN requires a lengthy configuration.

Listing 7: Automatically Enabling the Connection

```
### Open VPN connection manually:
$ sudo systemctl restart dsvpn_server.service
$ sudo systemctl restart dsvpn_client.service

### Enable VPN automatically on each boot:
$ sudo systemctl enable dsvpn_server.service
$ sudo systemctl enable dsvpn_client.service
```

In practical use, DSVPN shows its strengths if you rarely need a VPN and OpenVPN's long configuration is not worth the effort. One drawback is that DSVPN is only available for Linux and macOS. In many situations, however, you might want to tunnel your Android or iOS smartphone through the VPN to a secure network. The developer states that DSVPN is not intended as a replacement for classic tools like WireGuard. DSVPN solved an existing problem for the developer; perhaps it can solve one for you. ■■■

Info

- [1] WireGuard: <https://www.wireguard.com>
- [2] DSVPN: <https://github.com/jedisct1/dsvpn>
- [3] FreeDNS: <https://freedns.afraid.org>
- [4] Scripts for DSVPN: <https://github.com/ebal/scripts/tree/master/dsvpn>
- [5] "A Dead Simple VPN": <https://balaskas.gr/blog/2019/07/20/a-dead-simple-vpn/>
- [6] Glorytun: <https://github.com/angt/glorytun>



FOSSPicks

Sparkling gems and new releases from the world of Free and Open Source Software



Following Graham's recent coverage of the PC classic *Blade Runner*, running on the latest ScummVM, he was overwhelmingly jubilant to find that GOG (the games site) has finally been able to make it legally purchasable, decades after release. **BY GRAHAM MORRISON**

Information organizer

TreeSheets

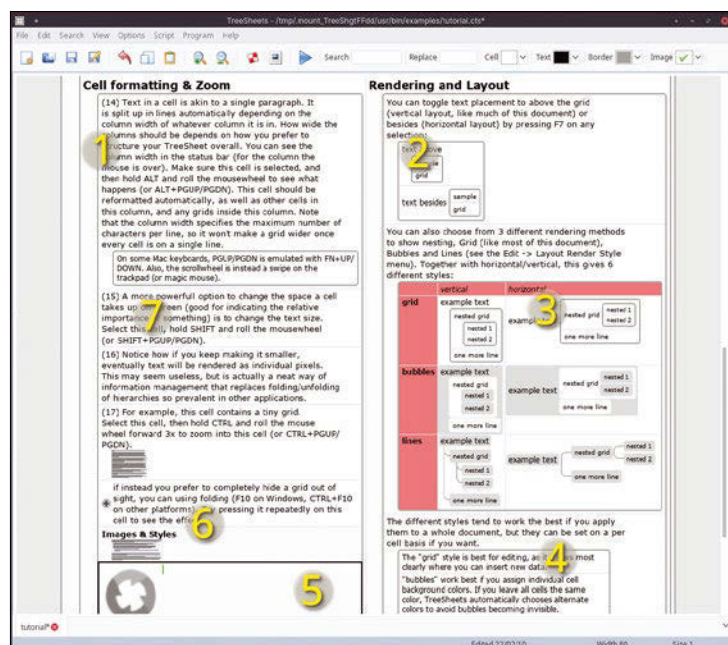
Data organizers, note takers, and memory mapper applications aren't initially the most exciting tools to install and use. The results typically look utilitarian, maybe even a little prosaic, and seldom seem to spark the imagination in the same way 3D tigers do for Blender. But this is all a little unfair, because these organizers

are only really limited by the person using the tool, and there are many times when a project is so big, an idea so convoluted, or a set of notes so unwieldy that you have to love any tool that can help straighten it all out.

TreeSheets is one of those tools. It's a note taking application that looks like it was designed in the late 1990s for

whatever version of Windows worked best at the time (or KDE 2.x). This is mainly due to it using the latest wx-Widgets. And while it has been under development for many years, 2019 finally saw the release of 1.0 and a move to a continuous integration build system, effectively making a new release available whenever new features are pushed to the code repository. But to prove you really can't judge a note taking app by its screenshot, TreeSheets is a more creative tool than a simple glance reveals. This is because while you can use TreeSheets to make notes and archive important pieces of data, you can also use it to organize, annotate, and grow your jottings in a logical and consistent way.

Each new document starts with an empty grid, much like an empty spreadsheet. Select a cell and start typing – the cell will expand dynamically to accommodate your text, as will the columns and rows. If you need to insert another row or cell between two ideas, you can do this by simply selecting the border between the two and starting to type. This is easily accomplished with the cursor keys, where you can move around the document without touching the mouse, but you can also click around the cells with your pointer. You can add images, too. While the text in a cell is designed to work like a single paragraph, there are no rules, and you can change the size and formatting of the text within each cell, even shrinking it down to microdot size if you find that a useful organizational idea. Cleverly, like the Inception of note-taking, you can also create grids within cells within cells. The whole document can be switched between the grid view, nested bubbles, and lines showing hierarchy – all using the same data within the document. It's impressive how quickly you can organize your ideas and see them as a nested set of lines and bubbles. All of this can be exported as HTML, which is great if you need an ad-hoc static site generator for your own ideas, and even XML if you need the ultimate format for keeping every chunk of information together in the most flexible and manipulable format.



1. Grid editing: Even though you add your own notes and jottings to a cell, they can be any size you choose.
2. Layout: The text in each cell can be precisely formatted, much like in a word processor.
3. Rendering methods: While the grid is the default layout for your ideas, you can easily switch between grid, bubble, and line charts.
4. XML output: Export as an XML file to maintain the relationship and context between sections.
5. Image import: Images can be imported and scaled alongside text.
6. Zoom levels: Text can be any size, and even so tiny it becomes a thumbnail for an entire grid within a grid.
7. Integrated tutorial: Load the included tutorial at any point to browse TreeSheets' key features.

Project Website

<http://strlen.com/treesheets/>

Virtual chanting

Xmonk

We often look at weird audio software here, but that weirdness is often accompanied by complexity. The amazing Orca audio programming environment, for instance, is the perfect example, because it extrapolate music composition to a terminal window and the placement of ASCII characters, which could only be mastered with careful study of the manual. It was capable of great results, but only after some serious time invested in learning how to get the output you wanted.

Xmonk isn't like this. It makes sounds, but it's altogether simpler and more immediate, and a little like the audio equivalent to Douglas Adam's electric monk, only instead of believing things on your behalf, Xmonk creates a chanting drone for you, so you don't have to.

First, to get sound out of Xmonk, you need to virtually plug it into something. This is because it's an LV2 plugin that needs to be hosted in a piece of audio software like Qtractor or Ardour. With that done, you open the single Xmonk window to see a 1960s-style tie-dyed image of someone meditating between two virtual sliders – one slider for gain and the other for sustain. Beneath this is a small keyboard icon, which, when clicked, opens a virtual piano keyboard. This saves you the trouble of plugging in a physical MIDI keyboard or generating notes another way, because this is how you generate a sound. Press a key, and you'll hear a chanting-like sound at a single pitch. Move the sustain slider up, and this sound remains without any further input. What isn't immediately obvious is that you can drag the cursor across the meditating figure to change the timbre of the sound. In fact, it's a vocal formant filter, with a horizontal drag taking



Despite being described as a toy, Xmonk sounds quite professional, perhaps half-way between a vocoder effect and a formant filter in advanced synthesis.

the sound through the vowel-like sounds, and the vertical axis changing the pitch. But that's much more technical information than you need to appreciate this plugin that generates great output and is easy to use.

Project Website

<https://github.com/brummer10/Xmonk.lv2>

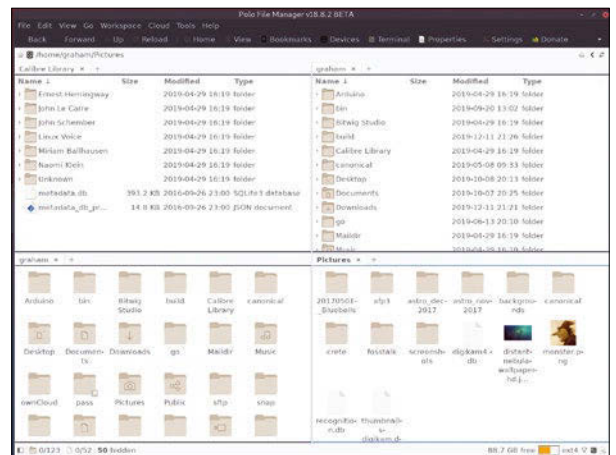
Advanced file manager

Polo

New file managers are always worth trying, because, first, they don't come along all that often, and second, because it's easy to forget that there might be a better way to do things. Polo is one of those file managers that is trying to do things differently, even before you get started. This is because it's developed with funds from Patreon and donations. The entire project is open source, but people who do contribute get easier access to a special set of plugins as a thank you. Another new thing is that before you start you get to choose the layout style you want from the file manager. You can choose between several panel views, for example, and the four panel split view is particularly effective if you're a

power user. These views can be changed manually by pressing F4, and you can split these views into lists, icons, tiles, and media after the startup wizard has gone, but it's great to see these features before you start.

Polo is built on GTK+, but fits just as cleanly into a KDE Plasma environment as it does a Gnome one. In fact, it feels very similar to Plasma's Dolphin. There's drag-and-drop file management, and you can see connected devices, including those encrypted with LUKS, create and view archives, boot an ISO image into a virtual machine and even download videos from YouTube, all without leaving a single application. Its cloud credentials also support adding Google Drive, Amazon Drive/S3, Dropbox, and a variety



An advanced option page lets you limit the number of CPUs and the amount of RAM that Polo takes up while managing your files.

of other file hosting services, all of which can then be accessed just like any other location. There's also extensive theming, and with the right GTK+ icon set installed, you can change almost any aspect of the application's appearance outside of any settings your normal desktop might impose.

Project Website

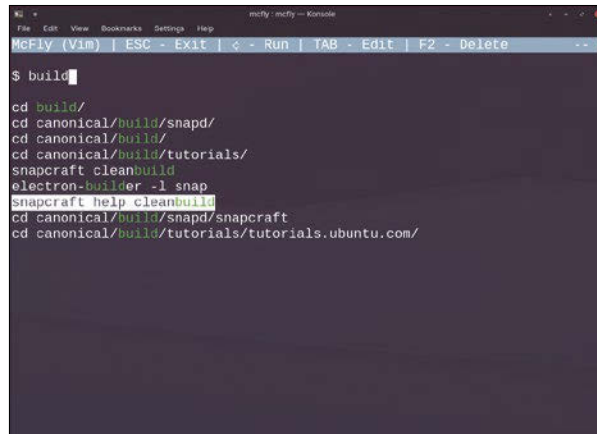
<https://teejee2008.github.io/polo/>

History search

McFly

One of the best features of the Bash command line is its history, handily stored in the `.bash_history` file, so it can be easily deleted. Apart from searching through the file manually, or with the `history` command, and using the up and down cursor keys to manually step through the history, its best trick is accomplished by pressing `Ctrl+R` to initiate a search. This lets you search for commands you know you typed but don't know exactly where to find. As soon as you've mastered `Ctrl+R`, it becomes a part of almost every command incantation on the command line. But `Ctrl+R` for history is also somewhat limited in its search capabilities, and that's where `McFly` comes in.

`McFly` is a `Ctrl+R` history replacement for the command line. It's easily installed via Rust's Cargo package manager and requires a single line added to your shell config. The first time you run your Bash terminal after installing `McFly`, it creates a database from your history, and this takes a few seconds. The new database includes added details, such as exit status and time stamp, but your old history is also maintained in parallel. The indexing is only done on the first launch, so it isn't too much of an inconvenience. You're then left with your normal prompt and the tantalizing possibilities that lie behind pressing `Ctrl+R`. Start typing a command, and pressing `Ctrl+R` takes you into a full-screen terminal application, complete with fuzzy search re-



With `McFly`, your history becomes more accessible. Sadly, without the option to take us back to the future.

sults listed below your prompt. This is the clever part, because behind the scenes a neural network is sorting these results, taking into account your current directory, the commands before the current one, when you last ran the command, and other variables. You can navigate through the results, but they're often so good you won't need to. It works like magic and can transform the way you use the command line.

Project Website
<https://github.com/cantino/mcfly>

Regex parser

rare

Regular expressions are powerful and terrifying in equal measure, and they're not something most of us want to decipher while we're happily navigating about on the command line. `rare`, however, might make you change your mind. It describes itself as a file scanner and regular expression extractor with real-time summaries, and it builds into a single binary you can place in your path. You use it like you might use `grep` or `find`, only rather than returning single results, `rare` will search through files for matches to your regular expression and return the results in different ways. If you use the `histo` argument, for example, the results are summarized into a histogram that's drawn directly into the command output. Using

the `analyze` argument will extract numbers that tell you details such as mean, median, min, and max about the nature of your results. Similarly, `tabulate` turns the same results into a table. There's even a mode, using the `filter` argument, that will return results just like `grep` without the analysis. The elephant in the room with all this power is that you will need to construct your own regular expressions for the searches you want to perform. There are plenty of resources that can help you if you've not done this before, with interactive online builders being a good recommendation. But with that out of the way, `rare` becomes as simple as `grep`, albeit with many options to help you target your data, even without the regular expression.



`rare` is a very quick and efficient way of analyzing data from the command line.

There is an interactive mode, for example, where you can follow changes in the data and update your selected output automatically. You can decompress a file while reading, sort rows, and set your own delimiters, and each output has its own set of arguments. Just like regular expressions, `rare` is difficult to use but can produce unrivaled results quickly.

Project Website
<https://github.com/zix99/rare>

Audio ripper

Flacon

Even in this age of music streaming, there are very few services that can offer audio at the same quality as you find on a humble compact disc. This is because CD audio is uncompressed and mastered specifically to get the best out of the sample rate and bit depth of its medium. It doesn't suffer anti-aliasing noise on high-frequency voices or high hats, and doesn't break into noise on long fade outs. Converting the audio into an AAC, MP3, or Vorbis stream not only compromises the quality, it can break continuity between album tracks, alter the dynamic range when tracks are normalized, and even make the original album order difficult to discern. So it's still worth buying CDs, or the WAV source files,

when you know the music is worth it. But when you do, it's not always easy turning that audio into something smaller and more transportable, especially from a single track rip off a CD. This is where Flacon can help.

Flacon is a GUI application that can convert a single audio file that's a concatenation of multiple tracks into separate files containing a single track each, as used by most media players. The input format can be WAV, but it can also be FLAC, APE, WavePack, and True Audio. The output format can be FLAC, WAV, AAC, OGG, or MP3. In-between taking the input and producing the output, Flacon will take the track information from a CUE file, the text-based description that most rippers generate from a CD, or provided with a download, and turn it into separate tracks, complete with filenames that match the track names. It can also use CDDb as



Even if you no longer have a CD player or drive, Flacon is still useful when converting purchased WAV files into individual FLAC files.

a source of this data, though the original site is down, and scan a collection of audio automatically, which is handy if you have not ripped the file yourself. And that's all there is to it. Flacon performs a simple job that's difficult to achieve without it.

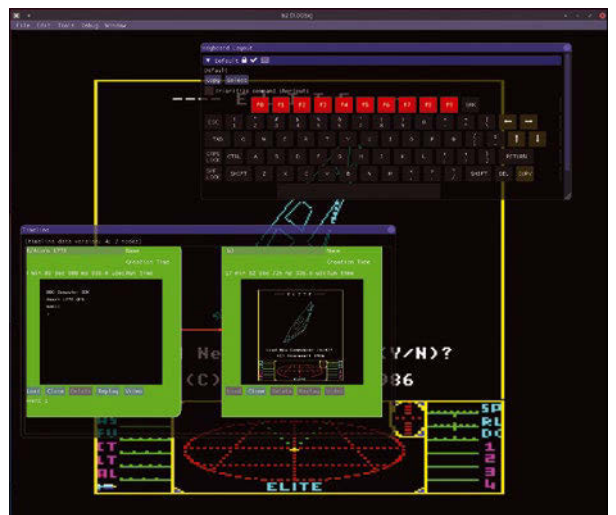
Project Website
<https://flacon.github.io/>

BBC Micro Emulator

b2

In the UK, in the early 1980s, there was a company called Acorn that made relatively expensive home computers that were mostly found in schools. However, many teachers also got a discount, and this meant you could also find them in the homes of teachers' children. These machines were the BBC Micro, and while they weren't as capable as a Commodore 64 or Atari 800, or as cheap and functional as a Sinclair ZX Spectrum, they were easy to program, easy to expand, and most of all, ran the original version of Elite. And back in 1984, Elite was the game to play. All of which means that while there's plenty of nostalgia locked up in these old systems, there's not quite the plethora of emulators you usually get for other famed old '80s computers.

Which is why it's great to see b2, a cross-platform BBC Micro emulator. From the moment you start it up and hear that simple square wave beep, you're transported back to your friend's house (the one with the teacher as a parent). b2 emulates the more expensive models: the Model B, the Model B+, and the even more expensive Master 128 editions of the hardware, the latter of which just happens to host the definitive version of Elite. If you have a disk image handy, you simply load it into the emulator and try to remember the load incantation for the BASIC interpreter (hint: run `*CAT` to get a file listing, and then load a file with `CHAIN "filename"`). There's a lovely on-screen display too, using an unusual but pleasing graphical toolkit. It's used to show when drives are being accessed and to reconfigure the keyboard layout. The output always looks fantastic, thanks to



Acorn, the makers of the original BBC Micro, later became ARM of CPU fame (an acronym that originally stood for Acorn RISC Machine).

its use of OpenGL, and the thoroughly modern audio output even includes disk noise. But the best feature is the timeline. This lets you record an emulated sequence, save states, and generate video from your recording. All of which makes revisiting the BBC Micro a complete joy.

Project Website
<https://github.com/tom-seddon/b2>

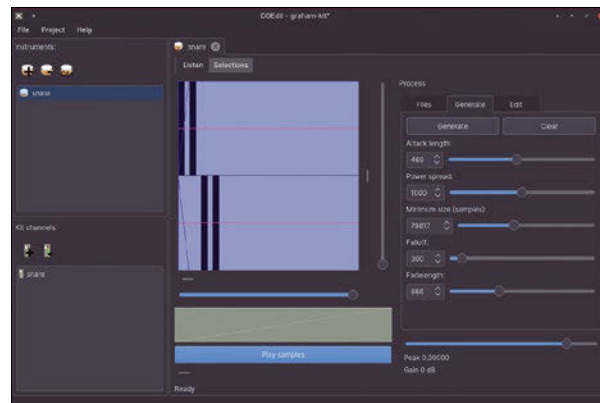
Drum generator

DrumGizmo

Even if you're not particularly into making your own music, you have to be into making your own drum beats! It requires none of the patience you need with pitched notes and chords, where you place one pleasing note after the other. Instead, you can often simply click around in a beat-maker drum machine for instant gratification. Which is probably why this type of software has been around for decades, with even the 8-bit sample-driven Micro-Rhythm, running on a Commodore 64 in 1986 being capable of professionally-passable output. Audio capabilities have obviously improved a lot since then, but the basic premise behind drum making software is the same. They generate sounds from pre-assigned MIDI note values, so that pressing a specific key will generate the same type of sound across drum kits and across different drum applications. A kick drum can be found on MIDI note 36, for example (C1), while the snare is on 36 and the high hats on 42. But often, you don't need to worry about the MIDI or note values, because those notes are represented by rows on a grid,

with timing on each column. In this way, a grid represents when a sound should be triggered as a cursor moves from left to right. It's how the brilliant Hydrogen works, for example, but it's not how this great piece of software, DrumGizmo, works.

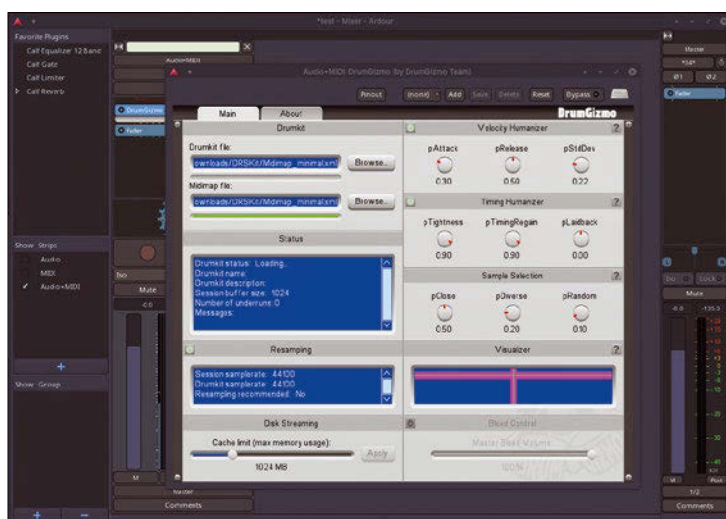
DrumGizmo is simply the drum engine used to process and generate percussion sounds. It does this either as a standalone application or as a plugin within your favorite music making environment. MusE or Qtractor are good examples, but there are many others, and these will often include the drum matrix that DrumGizmo is missing. DrumGizmo's focus is on the sounds, rather than the patterns, and the sounds themselves need to be downloaded separately. The sounds linked to from the main site have been meticulously recorded, often from multiple microphones across many different tracks, and they're huge. Even the smallest is a 2GB download. The DSRKit, for example, consists of eight different drums, recorded from a real kit into 13 different microphones. These recordings are



Each set of drum sounds has been captured with meticulous detail, and there's even a standalone editor to help you make your own.

each from close mics on the drums themselves and two overhead mics to capture the ambience, before being mixed into 13 separate channels, all of which appear in Ardour, a wonderful multi-channel panning tool. The quality is amazing and easily rivals commercial sample packs, which is hugely unusual for Creative Commons (CC BY 4.0) licensed media.

With the sounds loaded into DrumGizmo, there are further creative possibilities from within the user interface. You can use sliders to humanize, or add variation, to the attack and release times of the samples and change the timing of when a sound is triggered and the amount of variation in which particular sample is chosen for a hit velocity to make the output sound more natural. The developers have even written a white paper on how the sample selection algorithm works by estimating the power of a sample so that it can sort and trigger those that most accurately reflect the input strike velocity and dynamics. It's difficult to understand, but the results speak for themselves. Whether you're banking away on a QWERTY keyboard, using a cheap drum pad toy, or even a professional MIDI kit, the output sounds realistic, playable, and remarkably effective for a piece of open source software.



DrumGizmo's minimal user interface hides the complexity of the sample playback and audio data that helps to make it generate such realistic output.

Project Website
<https://www.drumgizmo.org>

Urban planning

A/B Street

Optimizing a traffic control system as the traffic – cars, vans, and trucks – travels around an urban road network is a classic challenge often used in computer science classes to teach problem solving and programming. If the current state of our roads is anything to go by, it's a challenge that has yet to be solved. Maybe this lack of progress is because the problem has never been made entertaining enough, which means this game, A/B Street, could be the beginning of a transportation revolution. It's a game based on optimizing a traffic control system while exploring how small changes to a city affect the movement of drivers, cyclists, transit users, and pedestrians. It does this through a top-down 2D map of a city, complete with traffic updates

and data on how the whole vehicular dance is being orchestrated. A/B Street does a brilliant job of making urban planning a fun and engaging activity.

What's most impressive is that this 2D map isn't a computer generated simulation, but a real map of Seattle taken from OpenStreetMap. Alongside this is a complete set of King County GIS data that includes sidewalks, parking, bike lanes, turning lanes, and even bus stops. All this data has been folded into a software engine that simulates pedestrians, cars, buses, and bikes, with unseen avatars even moving from one mode of transport to another. This is where the gameplay comes in. The game is a sandbox built within this functional environment where you do things like changing the timing on



Play with real traffic data on a real roadmap of Seattle and see if you can improve bus travel times or traffic efficiency for journeys across the city.

traffic signals, converting on-street parking to bus lanes, or changing turning directions. You can do this side-by-side with the original simulation to see if you're improving things or making things worse. When you've finished messing about, the game itself includes challenges for you to solve.

Project Website

<https://github.com/dabreegster/abstreet>

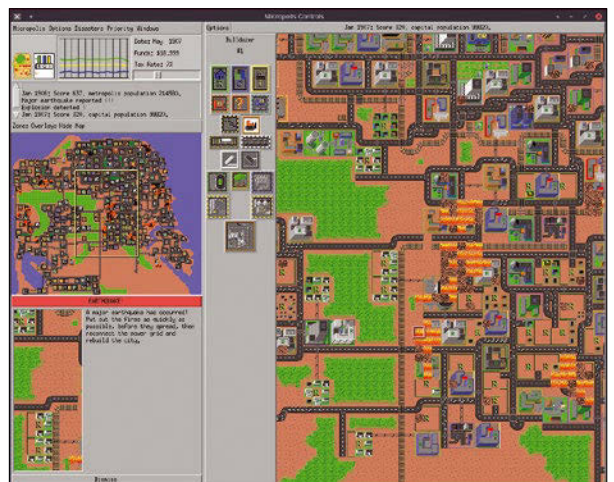
Simulated cities

Micropolis

Micropolis is a game that links the very latest kind of urban planning, as seen in A/B Street above, to an old 8-bit classic that's been around since 1989. That's because Micropolis is built from the source code of the original urban planning game, SimCity, whose code was released as GPLv3 in 2008, partly thanks to the One Laptop Per Child project. SimCity has obviously gone through many iterations since then, but there's a great charm and immediacy in playing the original version that's sometimes lost in the later more ambitious releases. Micropolis also requires very little CPU power and will run on anything – there's a great version for old Palm devices, including the WristPDA, if you still have any

of them around, for example, or try running it on a Raspberry Pi handheld.

The Linux version still runs well, and it's amazing to load it up in the 21st century and see the various simulated scenarios running so quickly. If you've not played before, the main aim of the game is to create a functional and expanding city, building the road network, zoning industrial and residential areas, and investing in police, health, and emergency services, all while giving more scrutiny to the plethora of incoming data than the average politician. If this sandbox mode seems a little serene, then there are various pre-set scenarios that challenge you to solve an imminent crisis. These include the 1906 San Francisco earthquake, the



The retro feel in Micropolis even extends to the original low-fi audio samples, which immediately transport you back to early '90s sound capabilities.

bombing of Hamburg in 1944, and a fictitious nuclear meltdown in Boston (set in the then distant future of 2010). These disaster scenarios have become a game cliché and have been copied in many other games, but it's also easy to forget just how immediate and fun they can be – 30 years after they were created.

Project Website

<http://wiki.laptop.org/go/Micropolis>

LINUX MAGAZINE
MAY 2019

MANAGING PDFs in LibreOffice
SNAPSHOTS PROTECT YOUR DATA

SNAPSHOTS
A better way to protect your data?
Plasma 5.14 Exploring KDE's new age desktop
Optimize Gimp Images with Python Plugins
tmate Emergency repairs from

LINUX MAGAZINE
JUNE 2019

FreeBSD 12.0
kubuntu
Double-Sided DVD INSIDE!

GNOME Remote Desktop Remote sharing for Wayland-based systems
GIT TRICKS VERSION CONTROL SECRETS FROM THE EXPERTS

GIT TRICKS
Version control secrets from the experts
Choose a Git GUI
Bash 5.0 Long-awaited update for the classic hacker shell
FreeBSD 12.0 Will this excellent server OS work as a desktop system?
VolksPC Run a Debian desktop over Android
Google Firebase Cloud storage for your IoT project
Sudo Voodoo Fine-tune system privileges

LINUX MAGAZINE
JULY 2019

le/PROJECT FOSS phone for the masses
ENERGY CONSUMPTION WHY ARE SOME APPLICATIONS MORE EFFICIENT?

ENERGY CONSUMPTION
Why are some applications more efficient?
Network Analysis Look for intruders with Wireshark and Netflow
File Encryption with EncFS
Picture Perfect We compare 5 image viewers
Maker Tricks Analyze malware on a hacked Rasp Pi
mitproxy Troubleshooting HTTPS connections

FOSSPicks
• KDElive
• Spookworm
• Plasma Pass

LINUXVOICE
Taking Notes with Joplin

LINUXVOICE
FOSSPicks
• Surge Switch
• Klok keyboard anonymiser

LINUX MAGAZINE
AUGUST 2019

Plant Monitoring with a Raspberry Pi
IoT on the Cheap Hack your smart plugs and make them talk to Linux

IoT on the Cheap
Hack your smart plugs and make them talk to Linux
Find Performance bottlenecks with eBPF
Karoshi The easy way to set up a complex server system
Timer Tools Schedule commands and scripts with at, cron, and anacron
chrony Sync up with the other time service daemon
Go Tricks Search for photos by GPS coordinates

FOSSPicks
• KDE Partition Manager
• Olvina Cloud Music Player
• DOSBox-X
Tutorials
• Scripting: Add User Input
• Preparing an Object for 3D Printing

LINUX MAGAZINE
SEPTEMBER 2019

IoT TRICKS Indoor climate control with MQTT
SPOTLIGHT ON SMALL DISTROS

SPOTLIGHT ON SMALL DISTROS
The best small-footprint distros for your environment
The new Linux firewall tool
your favorite PDFs
Network Sleuth Searching for problems with ARP
Timeshift Create and manage system snapshots

LINUX MAGAZINE
OCTOBER 2019

GPX VIEWER Map your tracks and view GPS data in a web browser
BACKUP Practical tools for protecting your data browser

BACKUP
Practical tools for protecting your data
Local server for more secure DNS
Generators way to better security
LibreELEC 9.0 Turn your Rasp Pi into a media center
Doomsday Rule A Go program for finding days of the week
Jade Desktop Will this innovative desktop environment change the game?

LINUXVOICE
Count Your Money with GnuCash
Famurio Time Management Tool
maddog on Huawei and 5G

LINUXVOICE
FOSSPicks
• Pick colors with Pick
• Name Generator
• Heaptrack

LINUXVOICE
Photo Tools
Tips for licensing and copyright

LINUX MAGAZINE
NOVEMBER 2019

ANONYMOUS FILE SHARING
Share files secretly with OnionShare
Glib: Analyze disk usage
anara Music Player Organize your music collection with podcasts and Internet radio
Login Use a Rasp Pi to resurrect your old CD player
HifiBerry Debian restructures the all-important /usr directory
Love For Lo Wireless net at a distance

LINUX MAGAZINE
MARCH 2019

VirtualBox Hacks
Save time and extend the power of your virtual machines
Scapy Automate packet analysis with Python
Put a Recording Studio on a Raspberry Pi
Elementary OS Elegant Linux with an ambitious vision
Create a Cartoon With open source tools
Tools for Writers Find words and organize your thoughts
Clear an

LINUX MAGAZINE
FEBRUARY 2019

FEATURING UBUNTU 18.10 "Cosmic Cuttlefish"

CUSTOMIZE THE BOOT MENU
Clean up your startup for clarity and your mistakes
PCIe SSDs Make the storage fit your hardware
IoT Tricks Store data in memory with Redis
Exploring Ubuntu 18.10
Pulse Sensor Measure your heartbeat with a Raspberry Pi
EncryptPad Text editor with easy encryption

LINUX MAGAZINE
JANUARY 2019

NAVIGATION
Free services that extend OpenStreetMap
Kernel Disposal Are old kernels sucking up valuable space on your hard drive?
Build a Robot Car
Zorin OS Will this well-appointed Linux succeed in attracting Windows users?
Put It On! Build your own wearable with Android and MIT's App Inventor
WebAuth Authenticate without a password
OOM Killer What happens when the kernel runs out of memory?

LINUXVOICE
FOSSPicks
• Gaia Sky
• Firefly apps sandbox
• Linux-72 retro-gaming engine
Tutorial
• Microblogging with Mastodon

LINUXVOICE
FOSSPicks
• eBEX-UI
• Auryo
• CalcuLat!

Tutorials
• Bash Arrays
• Natron Video

LINUXVOICE
FOSSPicks
• Sweet Home 3D 6
• Tintool
• Flare Empteen Campaign

Tutorials
• Shell Variables
• Natron Video Effects

LINUXVOICE
FOSSPicks
• Retroshare - Secure communication over the Internet
• maddog - IoT in Brazil
• Military-Malware Complexes - Security, cyberwar, and the search for zero days
Tutorial
Getting started with KDE's Plasma desktop

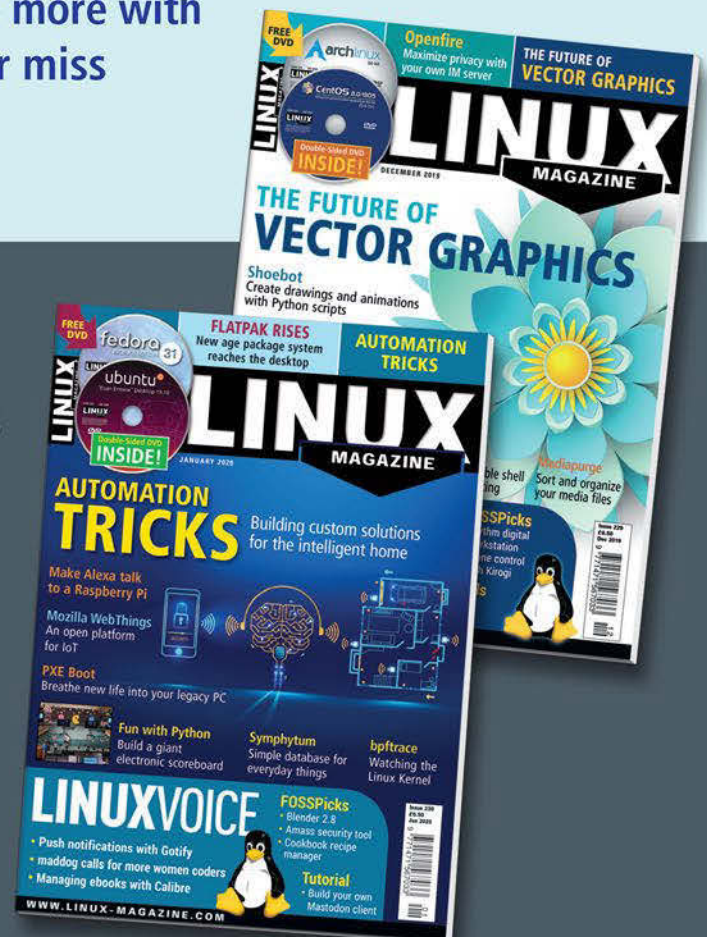
Linux Magazine is your guide to the world of Linux. Look inside for advanced technical information you won't find anywhere else!

Expand your Linux skills with:

- In-depth articles on trending topics, including Bitcoin, ransomware, cloud computing, and more!
- How-tos and tutorials on useful tools that will save you time and protect your data
- Troubleshooting and optimization tips
- Insightful news on crucial developments in the world of open source
- Cool projects for Raspberry Pi, Arduino, and other maker-board systems

If you want to go farther and do more with Linux, subscribe today and never miss another issue!

Subscribe now!
shop.linuxnewmedia.com/subs



An open source, federated video sharing platform

Freer Video Hosting and Sharing

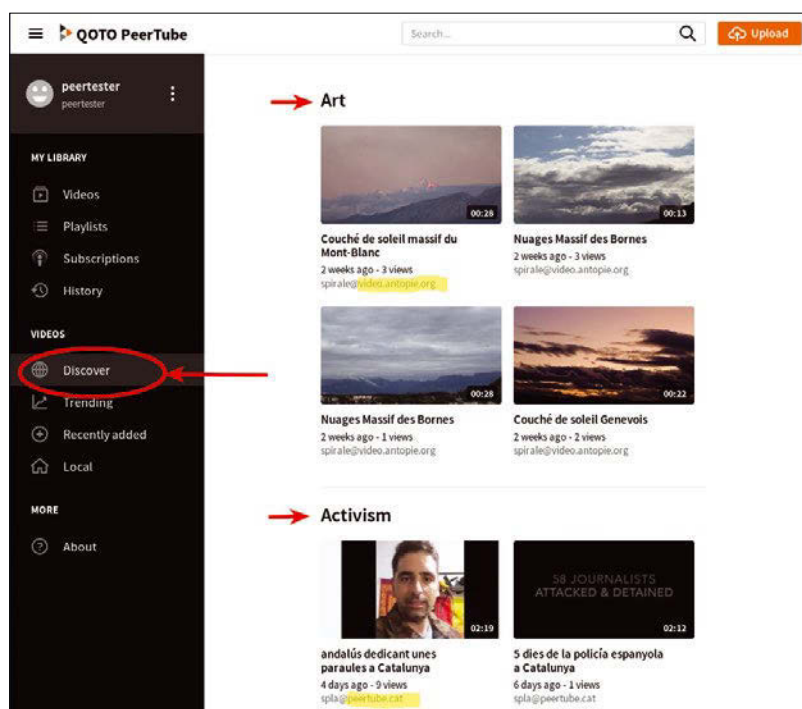
With PeerTube you can self-host your videos without the limitations embedded in YouTube and similar platforms.

BY MARCO FIORETTI

PeerTube [1] is a free and open source video hosting and sharing platform with great ambitions. With PeerTube, everybody can set up a website where all registered users can upload, organize, and share as they like all their videos, without the constraints and risks that plague platforms like YouTube.

In this tutorial, I will describe the main components of a full PeerTube installation, how they work, and above all the pros and cons of running your own instance versus just registering as a user on somebody else's instance. This is necessary, because ignoring PeerTube would be a serious error, but it would also be a mistake to jump into it without clear ideas of what's involved.

Figure 1: The highlighted links show how one PeerTube account gives access to videos on different servers.



I will also outline how to set up and use a PeerTube user account on existing instances, and how to prepare yourself for installing your own.

PeerTube Architecture

The core of PeerTube is a Node.js application, distributed under the AGPL v3 license. In practice, in order to work, a PeerTube-based video hosting website requires a whole bundle of free software. For example, PeerTube needs the Yarn package manager and a PostgreSQL database server to store user parameters and video metadata. Under the hood, the FFmpeg transcoder converts video to the open formats most suitable for efficient streamings in multiple resolutions.

Equally important are the NGINX web server, and the Certbot/Let's Encrypt system for handling the SSL certificates that allow encrypted connections to a PeerTube instance. The WebTorrent protocol allows multiple simultaneous viewers of the same video to download and share different parts of it among themselves, in order to not overload the server.

Now, if this were the whole story, PeerTube would be just a do-it-yourself, standalone, web-based video manager: a nice and useful project for sure, but nothing really special.

But the real value of PeerTube is not its looks, ease of use, or how well it transcodes video clips: It is federation. This, as shown in Figure 1, is the ability for any single instance to show videos uploaded to other instances, side by side with the ones hosted locally. Technically, this is done through the ActivityPub protocol, which is used in the same way by Mastodon, the most popular open source alternative to Twitter.

Pros

By automatically exchanging notifications and metadata with other instances, even a small installation of PeerTube on a cheap, low-end server may show its users one single catalog of thousands of videos. Some day, ActivityPub should allow you to browse even MediaGoblin instances in the same way! In general, any application that supports ActivityPub may syndicate content from PeerTube instances.

You can get a very good feeling for how all this works in practice by visiting a PeerTube demonstration server [2], and then any of the public instances registered in the official tracker [3].

The same tour will also show how its architecture gives every administrator or (to a much smaller extent) simple user of PeerTube much more freedom and other advantages than are available today on platforms like YouTube.

A PeerTube instance frees all its own users, as well as everybody else browsing its videos, from automated tracking and profiling (i.e., from advertising or surveillance). Besides, no corporate policy (within the limits of law, of course) can forbid a PeerTube administrator to host any video or take it offline without explanation or excuses. Inside PeerTube, no flawed copyright-enforcement algorithm may automatically erase your content and maybe make you lose money, too.

At the same time, as a PeerTube administrator, only you decide what is acceptable content and user behavior on your instance, or with which other servers to federate. As an administrator, you can also enable or disable user registration, moderate uploaded videos, and set user quotas.

If you don't want to or cannot be a PeerTube administrator, but just want to watch videos without banners and tracking, don't worry! Even simple users of Peertube instances can enjoy many of those freedoms. There already are lots of independent instances, and many more will surely appear in the near future, each with its own, possibly unique terms of service, but never as addictive as YouTube. This means that each individual, whatever her needs are, can sooner or later find an instance that suits her well.

Cons

Unlike YouTube, PeerTube instances are not designed to lure you into watching as many videos as possible, in order to profile you and maximize advertising profits. Whether local or federated, PeerTube just displays videos as they appear (Figure 2). Above all, federation is voluntary: If you subscribe to one PeerTube instance, but most of the videos you would find interesting are published on another instance, they will automatically show up in your original PeerTube account

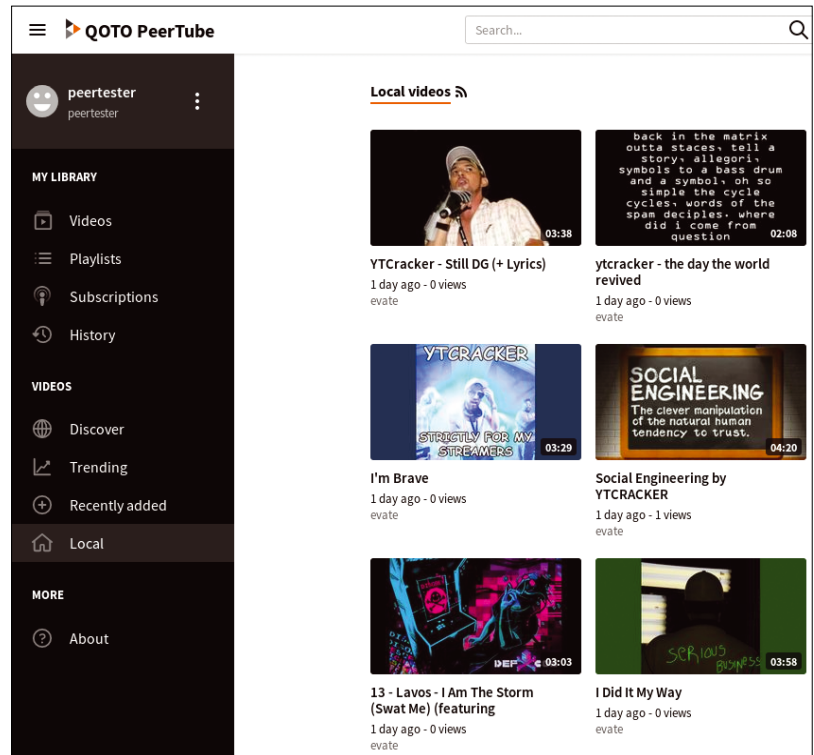


Figure 2: The local content of a PeerTube instance has its own separate tab.

only if both the administrators involved agreed to federate their instances.

On one hand, this can make discovery of new, interesting content sensibly slower than it is inside YouTube. Surely, we could and would still use search engines to discover video content. Still, in a PeerTube world, we may likely end up watching *less* video than today – maybe much less. If you think about how much time we waste on YouTube as a society, this comes out as a feature, not a bug.

At the same time, as long as federation happens at instance level, the only way for you to get seamless access to any PeerTube instances you may be interested in is to set up your own, which, as I will argue shortly, still looks pretty far from being an easy task. I propose a solution to this problem at the end of the tutorial.

Next on the list of PeerTube cons is performance: A major reason why YouTube became so popular is that it can spend billions on bandwidth and state of the art, constantly optimized content delivery networks. The video streaming from a PeerTube instance, unless its owner spends significant money on connectivity, will statistically be slower and more subject to glitches than that from YouTube, even if that instance only hosts a few videos. Please note that WebTorrent won't change that: Torrents work as advertised only when many people want to download the same file, in the same moment.

Last but not least, with great power comes great responsibility. Whoever runs a PeerTube instance becomes not only the first target of any

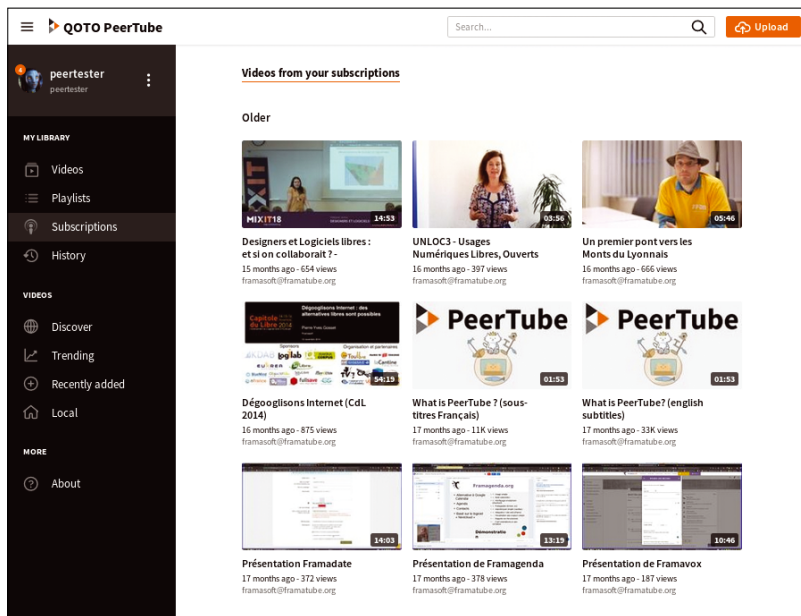
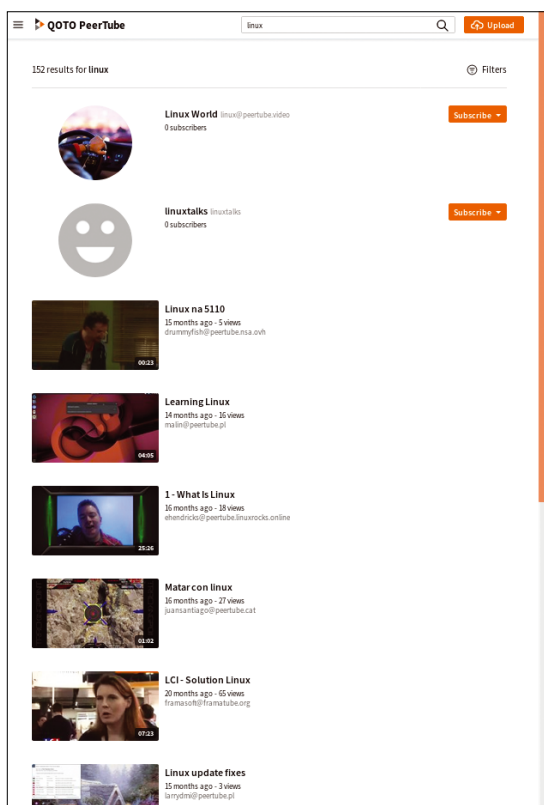


Figure 3: Customize PeerTube by finding and subscribing to interesting channels.

Figure 4: Type any keyword in the search box, and discover all relevant results in all PeerTube instances federated with yours.



copyright complaint, but also the sole arbiter of any fight among users. For some users, this will be a nonissue, for others a showstopper. Now that you have been warned, let's see what you may do with PeerTube, first as a normal user, then with your own instance.

Using PeerTube

To start using PeerTube, first find the instance that best suits you, from the official tracker [3]. In this phase, never forget that PeerTube was also created to let any video hosting organization deny

(or allow) whatever they wanted on their own server. Therefore, even before browsing the videos of each instance, check out its About page very carefully, to find out what its policies are. Once you have found a

proper home for your own content, or just the place with the most interesting videos, choose a user name, create an account, and wait for the email confirmation. When that comes, log in and configure your account, starting with notifications: You can decide whether to be informed right away, or not, about new videos, new followers, new comments to your own content, or your mentions in somebody else's comments.

Next, you will need to declare whether you want to see videos only in certain languages, see or hide sensitive material, and enable or disable autoplay. Another important decision is about using WebTorrent in your browser. If you do it, you will exchange parts of each video you download from PeerTube with other, simultaneous viewers of the same video. This will improve playbacks, both yours and theirs, but also expose your IP address to unknown third parties. Of course, it's up to you to decide whether you like this or not.

Once you are done with the basics, set up your own channels. PeerTube channels work like independent playlists, each with a different topic. The "independent" part means that a PeerTube user cannot subscribe to "whole" accounts of other users. The only way to see every video from a user who set up several channels would be to subscribe to every one of them separately.

This may seem like a bit of a hassle, and sometimes it is. But it also is one more support to avoid distraction. Subscription, in fact, makes you automatically see everything the users post in the channels to which you subscribed (Figure 3).

You can find videos about specific topics, inside your instance and all the others federated with it, by typing relevant keywords into the search box of your main PeerTube window (Figure 4). You can then subscribe to the channels that posted those videos by simply visiting them and then clicking on the corresponding button (Figure 5). (For the record, you can also perform most of the main functions of the PeerTube web interface in the third party Android app called Thorium [4]).

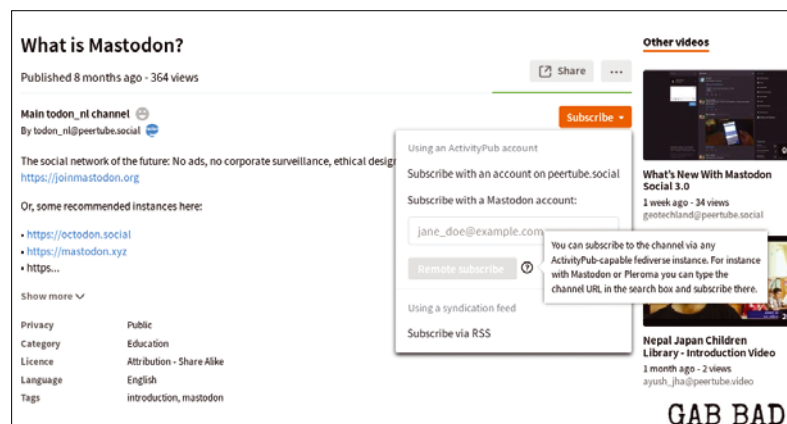


Figure 5: You can subscribe to PeerTube channels in several ways: from PeerTube, from Mastodon, or with good old RSS!

If you already know the URL of a channel you want to subscribe to, you may just paste that URL either in your browser address bar, or in the PeerTube search box. Of course, this procedure works only when you had already discovered, in some other way, where to find interesting videos. As I already said, in a loose, continuously changing federation of independent instances, nobody goes out of their way to take note of what you like and make suggestions.

A dirty but simple trick suggested by the PeerTube website solves this problem, at least partially: Periodically check out the PeerTube’s instance tracker [3], sorting the results by platform version, as this may make it easier to spot newer instances.

A completely different way to make PeerTube your one-stop interface to all the videos you care about is the Firefox and Chrome extension called PeerTubeify [5]. As shown in Figure 6, whenever you watch a video on YouTube, if there is a video with the same title on the PeerTube instance you told it to check, PeerTubeify will display a clickable thumbnail that will lead you straight to that video.

You can upload videos to your PeerTube account from different sources: your own computer, PeerTube torrents, or just normal URLs of video files. In the latter case, the files must be raw MP4 video, or any other format supported by the *youtube-dl* utility. Besides language, tags, and privacy settings, Figures 7 and 8 show two parameters that deserve particular attention: licenses and categories. The reason is that, in both cases, you can only choose among the options predefined by your PeerTube administrator. On the other hand, all PeerTube instances support independent video creators: You get a dedicated box to describe where and how your followers may send money or support you in any other way.

When you are done with publishing a video,

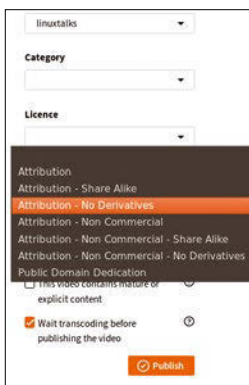


Figure 7: PeerTube administrators decide which licenses are acceptable on their servers.

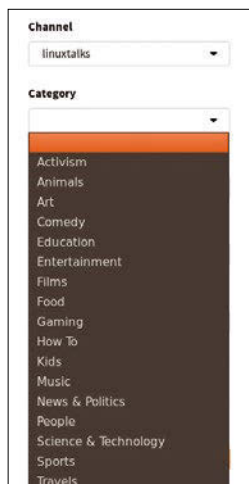


Figure 8: Even categories are decided by PeerTube administrators.

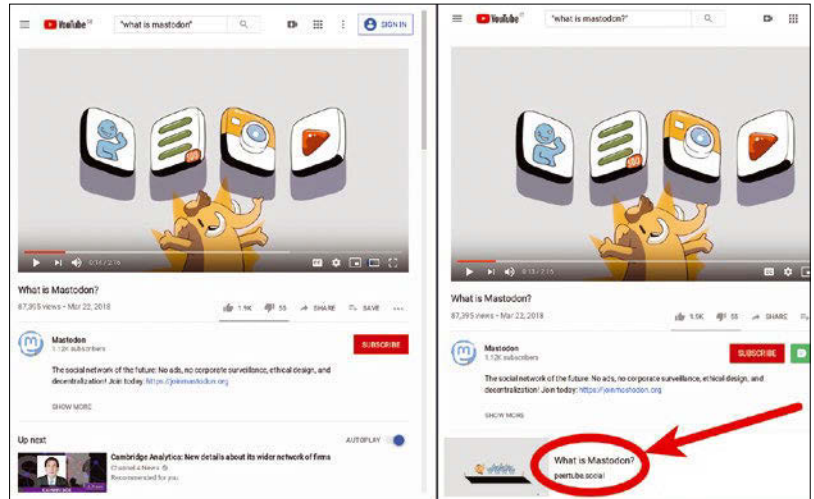


Figure 6: Thanks to PeerTubeify, only the copy of Firefox on the right can lead you to PeerTube copies of videos hosted on YouTube.

you can finally preview it inside PeerTube (Figure 9), choosing among different resolutions and playback speeds. Depending on the policy of the instance you subscribed to, you may experience the same problem I encountered with the test account I created for this tutorial: All your videos may be auto-blacklisted for several days, until a moderator has time to check and approve them. That’s the price to pay for not being wrongly blacklisted by algorithms, I assume.

Running Your PeerTube Instance

In theory, the need to avoid blacklisting and the other limits for PeerTube end users mentioned above are excellent reasons to set up your very own instance. After all, that’s the whole point of free and open source software, isn’t it? In practice – and I write this

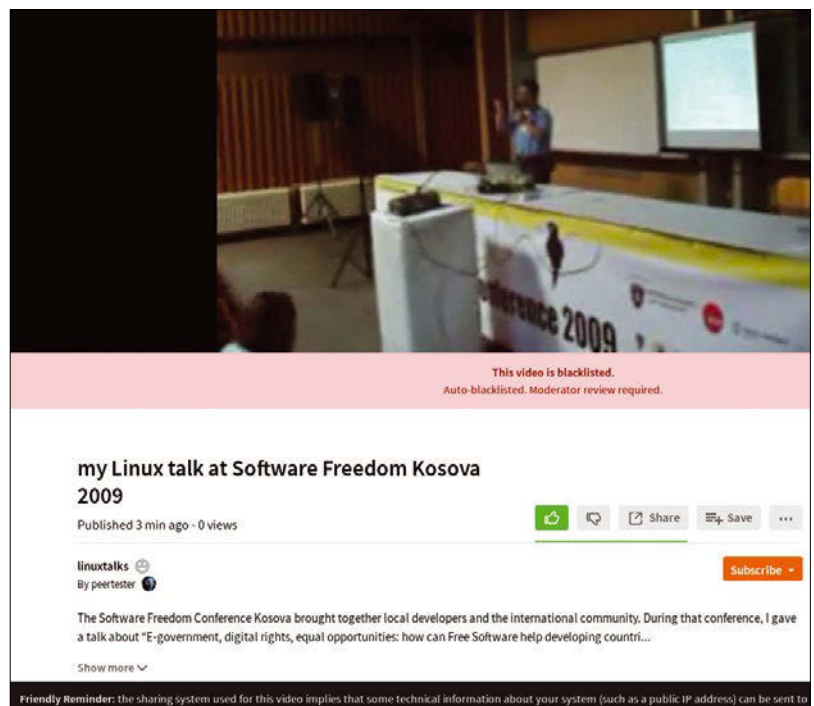


Figure 9: Watching videos in PeerTube: Note the automatic moderation warning!

with full consciousness of how weird it is to give such advice in a free software tutorial – think twice before making such a move.

The reason is that, as good and necessary as it is, PeerTube has one objective constraint that is not its fault at all, plus (at time of writing) a more critical sore spot, that will surely heal as the project matures but is not negligible right now.

The objective constraint is hardware-related. According to the PeerTube developers, the main demo server [2], which has 2 vCores (the minimum recommended to enable transcoding, which is crucial to optimize bandwidth) and 2GB of RAM, consumes on average less than 35 percent of CPU cycles and less than 150MB of memory.

Those are good figures, but only tell half the story. Video needs much, much more disk space than other content and much more bandwidth, too. If you want to offer a reliably decent service, forget the cheap hosting packages that would be more than enough to run almost any other type of website.

The sore spot is in the software. PeerTube's end-user interface is very simple and pleasant to use. Installing and configuring PeerTube and its environment is much more complicated and unpredictable than any other package I have installed in the last five years.

PeerTube has a lot of critical dependencies [6], many of which are not exactly simple to install and configure in the first place: PostgreSQL, NGINX, Node.js, npm, Yarn and an up-to-date version of g++ to run it, FFmpeg, the Redis in-memory data store, the Traefik reverse proxy, and more.

Besides, it seems that each of those packages must be exactly of a certain version, and configured exactly in one, and one mode only, needed for PeerTube. The final result is also likely to depend on the previous, specific configuration of the Linux distribution that hosts PeerTube.

Even Docker containers, the default solution for this kind of problem, may not be enough for PeerTube. Some documentation is available [7, 8], but it is not sufficient, in my opinion. When testing Docker containers for PeerTube, I have encountered problems I never had with containers for other packages, from not having a compatible version of `docker-compose` to what seemed not fully documented dependencies on packages (e.g., a PostgreSQL server) external to the container.

Now, don't get me wrong here: PeerTube is great, its developers are doing a great job, and, in spite of what I just wrote, it *is* possible to get PeerTube up and running on Linux, without being a hard-core developer. I am only saying that it is a longer, much more complex process than installing stuff like WordPress or Drupal.

With PeerTube, there are many more parameters and steps to get exactly right. For all these reasons, plus lack of space, I don't feel comfortable describing a complete procedure that may work only on a server identical to mine, and which, even in that case, may be half obsolete anyway by the time you read this.

Instead of doing that, I will conclude this tutorial with advice less specific than a complete installation procedure that should still save you a lot of time if you decide to install PeerTube.

First of all, check if there is one single Docker image that contains everything needed to run PeerTube (including the scripts mentioned below), already preconfigured and fully customizable with only one, fully documented environment file. If such a container exists, try that first, to get familiar with PeerTube and its administration procedures.

When you feel ready, go for a real installation, but in this way: Ask on the PeerTube forum [9] who is already running the current version of PeerTube without problems and then just clone that full Linux installation, from the kernel up, on a virgin virtual or physical server, reserved exclusively to PeerTube. A more expensive alternative, but worth the money if you can afford it, would be to ask on the same forum for Web hosting providers that officially support PeerTube installations.

Post-Install Tips

First, be prepared for migration, before you even start using PeerTube. There is no real "data ownership" if you cannot move to any other server, in any moment, with zero or minimal loss of service, and doing that with PeerTube may be as complex as installing it. Detailed migration instructions are available on the website [10], but, just as with installation, fully cloning your whole PeerTube/Linux installation to a new machine may be the best solution.

Second, decide very carefully with which servers to federate, and regularly check whether their policies change. You can add or remove instances to follow in the *Manage Follows* | *Follow* tab of the Administration panel. There also is a script, called PeerTube autoDiscover [11] that can facilitate the discovery of new instances.

Third, regularly check which instances follow yours, as you may want to block them if your respective policies are too different. You need to do this because, at least at time of writing, PeerTube automatically accepts requests to be followed.

Fourth, study and take advantage of the PeerTube maintenance scripts [12], which you can launch on the server or even remotely, from your laptop.

For example, once you are sure that your instance is configured properly, you can automatically load

both your own and third-party videos inside it. In the latter case, of course, you must comply with their licenses. To upload videos from your computer straight into a PeerTube account, use the `peer-tube-upload.js` tool [12].

Videos already hosted online, instead, are the target of another script, called `peer-tube-import-videos.js` [12]. This utility can upload to PeerTube all the videos of any account on YouTube, Vimeo, Dailymotion, or any other website supported by the `youtube-dl` tool. The script also knows how to avoid downloading the same video twice, so you can even run it as a cron job, to keep your instance always up to date.

Fifth, keep video transcoding under control. Efficient video streaming heavily relies on transcoding (i.e., converting videos in formats or resolutions more suitable for streaming than their native ones). The PeerTube “server tools” [12] for this task are called `create-transcoding-job.js` and `prune-storage.js`. The first forces transcoding of already existing video files. The second finds and deletes all the temporary files created by some transcoding procedures, recovering space on your server.

Conclusions

Federation is the way to go to build a better web than what we have today, and PeerTube may play a critical role in this space. The (current) complexity of its software [13], coupled with the intrinsically heavy requirements of video streaming, probably make PeerTube not yet suitable for many individual users.

At the same time, I consider PeerTube already suitable, and perfectly within reach, for many organizations of any size and nature, from schools and universities to NGOs and web-hosting cooperatives. Considering this, let me finish with two proposals that may make PeerTube much easier to use, and thus greatly facilitate its adoption.

One is to create an official ISO image and container of some bare-bones Linux distribution, with all (really all) and only the software needed to run PeerTube.

Another is a single-user PeerTube aggregator, that is a web front end to multiple, independent PeerTube instances. Such an interface (“myPeerTube?”), may be installed even on free or very cheap hosting accounts, because it would not host video. It may even become a

Nextcloud plugin! In any case, it would allow any individual to seamlessly use multiple, independent instances that, for whatever reason, do not want to federate with each other. Anybody could subscribe from a single window to those instances and see all the corresponding videos and notifications in that same, single window. I believe that with a client like this, and a really turn-key “PeerTube Linux” distribution or container, PeerTube adoption would greatly increase. What do you think? ■■■

Info

- [1] PeerTube: <https://joinpeertube.org/>
- [2] PeerTube demonstration server: <https://peertube.cpy.re>
- [3] PeerTube’s instance tracker: <https://instances.joinpeertube.org/>
- [4] Thorium: <https://f-droid.org/en/packages/net.schueller.peertube/>
- [5] PeerTubeify: <https://addons.mozilla.org/en-US/firefox/addon/peertubeify/>
- [6] PeerTube dependencies: <https://github.com/ChocoboZZ/PeerTube/blob/develop/support/doc/dependencies.md>
- [7] PeerTube in a Docker container: <https://github.com/ChocoboZZ/PeerTube/issues/1227>
- [8] Migrating videos to PeerTube in a Docker container: <https://www.artificialworlds.net/blog/2018/08/15/migrating-videos-from-youtube-to-peertube-inside-a-docker-container/>
- [9] PeerTube forum: <https://framacolibri.org/c/peertube>
- [10] Migration guide: <https://docs.joinpeertube.org/#/maintain-migration>
- [11] PeerTube autoDiscover: <https://github.com/Jorropo/PeerTube-autoDiscover>
- [12] PeerTube command-line tools: <https://framagit.org/framasoft/peertube/PeerTube/blob/develop/support/doc/tools.md#cli-wrapper>
- [13] Questions about the current state of PeerTube: <https://github.com/ChocoboZZ/PeerTube/issues/1457>



Using Readline functions in Bash Between the Lines

Readline provides you with a rich set of tools for working with text and moving around quickly and efficiently on the command line.

BY PAUL BROWN

Try this: Open a terminal window and type the following at the prompt:

```
ls non-existent_dir
```

Then move your cursor back to the space between `ls` and `non-existent_dir` and press `Ctrl+K`. Next hit `Ctrl+Y` and see what happens.

Congratulations: You are now using Bash as it was used 30 years ago. Or, to be precise, you are using Bash's Readline [1] interface.

Originally written by Brian Fox, the creator of Bash, Readline is a library for advanced key-jockeying for text-based interfaces like Emacs and, yes, Bash. It provides a set of utilities for cutting and pasting (called "killing" and "yanking" in Readline parlance) before such things were made common in graphical interfaces.

In the examples above, the *K* in `Ctrl+K` stands for "kill" and the keyboard shortcut `Ctrl+K` kills everything to the right of the cursor until the end of the line. The *Y* stands for "yank" and it pastes whatever was last killed into what is called the kill ring.

One Ring

The kill ring is where killed strings go. Remember that killing is like cutting, so the strings you kill don't really go away. The kill ring is like the desktop's clipboard, but completely independent from it.

The nice thing about the kill ring is that it has a number of slots, usually about 60, so you can store quite a few strings in it, and you can rotate through them when yanking (pasting) them back.

To understand better how it works, try this – type

```
one two three
```

onto an empty line in your terminal.

Move the cursor to the beginning of *three* and press `Alt+D`. This will erase *three* from the line. The difference with `Ctrl+K` is that it deletes until

the end of the current word, not the current line. So, if you move your cursor to the beginning of *one* and press `Alt+D`, *one* will disappear into the kill ring, but *two* will stay.

Finally, move to the beginning of *two* and press `Alt+D` to pull it into the kill ring. Now the ring contains *two*, *one*, and *three* in that order: *two*, the last element you killed, is at the top of the kill ring, then *one* is underneath *two*, and *three*, the first element you killed, is at the bottom. (I know: the metaphor of a ring with a top and a bottom doesn't seem to make much sense right now, but bear with me.)

Hold down `Ctrl` and press *Y* as before and *two*, the top element, pops up on your line. Do the same thing again and *two* will pop up again. But this time, hold down your `Alt` key and press *Y*. Suddenly *two* becomes *one*.

You have just taken *two* off the top of the kill ring, pushed it to the bottom, and the next element has risen to the top instead. In fact, you can press `Alt+Y` several times, and you will cycle through all the elements in the kill ring. When you reach the last element, you will cycle back to what used to be the first item again. The ring metaphor makes more sense now, right?

Apart from killing and yanking, Readline also supplies functions that allow you to move around on a line. You can of course use the Home key to move to the beginning of a line instead of having to press `Ctrl+A`, or End instead of using `Ctrl+E` to move to the end of a line. But in any case, the everyday actions of moving to the beginning or the end of a line, even deleting a character, are all Readline functions mapped to special keys on your keyboard.

Talking of deleting, try this: Say you wanted to kill the FileZilla application but typed *firefox* by mistake:

```
killall -s KILL firefox
```

You could press backspace five times or you could hold down the `Alt` key and press *5*. Your

until the end of the line. Note that you can have several of these combos one after another to carry out different tasks, like `\C-x\C-r`, which reloads the list of macros into Bash after you have modified them.

- `\C-` followed by the name of more than one key means hold down the Ctrl key while you hit the first key listed. Then, let go of Ctrl and hit the second key listed. For example, `\C-xx` means “hold down Ctrl and hit X, let go of Ctrl, and hit X again.”
- `\M-` key means use the Alt key instead of Ctrl. It works the same as Ctrl, and an example would be `M-3` to enter 3 as an argument for a Readline function.

To create a new key combination for yourself, you don't have to edit `/etc/inputrc`, since you can create your own file in your home directory. Let's do that so we have a keyboard shortcut to copy and cut regions to the kill ring.

Fire up your favorite text editor and create a file called `.inputrc` in your home directory (see Listing 2).

Line 1 pulls in `/etc/inputrc` so that you get all the benefits from the system-wide defaults. Line 3 defines a new keyboard shortcut. In this case, you are associating the `Ctrl+X` and then `C` combination with Readline's `copy-region-as-kill` function. Line 4 is the same thing, except you are associating the `Ctrl+X` and then `X` combination with the `kill-region` function. For the record, there is a complete list of Bash's Readline functions in the online documentation [2].

Let's try it out. Save `.inputrc`, exit your editor and press `Ctrl+X` and `Ctrl+R` so the changes are applied for the current shell. If there is anything wrong with your script, Readline will show an error.

When everything is okay, type a long line into your terminal:

```
wget "https://somedomain.com/
somefile" && echo
"Downloaded successfully"
```

Move your cursor to the `s` of `somefile` and press `Ctrl+@`. This sets the mark on the `s`. Now move your cursor to the double quotes after `somefile` and press `Ctrl+X` and then `C`, the keyboard combination you set up in `.inputrc` to copy the region to the kill ring.

It may seem that nothing has happened, but now move the cursor to after `Downloaded` and press `Ctrl+Y` to yank the latest item on the kill ring into the line. Hey presto!

```
wget "https://somedomain.com/
somefile" && echo
"Downloaded somefile successfully"
```

If instead of using the keyboard shortcut for copying, you had held down `Ctrl` and pressed `X` and then let go of `Ctrl` and pressed `X` again, you would've cut the region between the mark and the point.

Apart from associating shortcuts with Readline's predefined functions, you can also create your own functions and keyboard shortcuts for them.

Open `.inputrc` in your text editor again and add the following line:

```
"\e1": "ls\015"
```

This creates a keyboard combination (press `Esc`, let go, and then press `L`) that lists the content of the current directory. The `\015` part of the command is the code for *Carriage Return* (Enter) in ASCII.

You can find more key codes using the shell's `showkey -a` command (Figure 3).

For example, suppose you wanted to list the contents of the current directory by pressing `Alt+Right Arrow`. Run `showkey -a` and hold down `Alt` and press the right arrow to discover the key code (Listing 3).

What you're interested in is the first line `showkey` spits out after pressing `Alt+ Arrow Right`.

Figure 3: Use `showkey -a` to find out the codes for non-character keys on your keyboard.

```
[paul@Rachel ~]$ sudo showkey -a
[sudo] password for paul:
Press any keys - Ctrl-D will terminate this program
^[[A      27 0033 0x1b
          91 0133 0x5b
          65 0101 0x41
^[[C      27 0033 0x1b
          91 0133 0x5b
          67 0103 0x43
^[[1;3C  27 0033 0x1b
          91 0133 0x5b
          49 0061 0x31
          59 0073 0x3b
          51 0063 0x33
```

```
Listing 2: .inputrc
01 $include /etc/inputrc
02
03 "\C-xc": copy-region-as-kill
04 "\C-xx": kill-region
```

```
Listing 3: Getting Key Codes
01 $ sudo showkey -a
02 Press any keys - Ctrl-D will terminate this
   program
03 ^[[1;3C 27 0033 0x1b
04          91 0133 0x5b
05 ...
```


In Listing 3, that would be line 3. Focus on the bit that says `^[1;3C`.

Press `Ctrl+D` to exit `shoukey` and open `.inputrc` in your text editor.

Change the line that says

```
"\e1": "ls\015"
```

to

```
"\e[1;3C": "ls\015"
```

Save the file and reload the Readline macros (`Ctrl+X, Ctrl+R`).

Now hold down `Alt`, press the `Right Arrow` key on your keyboard, and the current directory will list.

History

The Readline set of functions also covers Bash's history. Every time you press the `Up Arrow` to navigate to a command you have used earlier, you are invoking a Readline function. But there is more to finding entries than just pressing the `Up Arrow` over and over.

Try this: Start on an empty line and press `Ctrl+R`. Bash will enter incremental search mode. This

```
[paul@Rachel ~]$
(reverse-i-search)`help': filelight --help
```

Figure 4: The Readline search functions show entries in the Bash history as you type.

means you can start typing in some of the letters of the command you are looking for and Bash will search upwards through the history and show you matches as you type (Figure 4).

Conclusion

There are so many more useful Readline functions. Check out the documentation and you'll see how the Unix pioneers made do with text-only terminals very well indeed, thank you very much.

Nowadays, many of these shortcuts may not be as helpful as before, but others give you very useful shortcuts that can make life on the command line much more productive and interesting. ■■■

Info

- [1] Readline: <https://tiswww.case.edu/php/chet/readline/readline.html>
- [2] Bash Readline documentation: https://www.gnu.org/software/bash/manual/html_node/Function-Index.html

What?!
I can get my issues SOONER?

Available anywhere, anytime!

Sign up for a digital subscription and enjoy the latest articles on trending topics, reviews, cool projects and more...

shop.linuxnewmedia.com/digisub

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.



SCaLE 18x

Date: March 3–5, 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.

CloudFest 2020

Date: March 14–19, 2020

Location: Europa Park, Germany

Website: <https://www.cloudfest.com/>

CloudFest 2020 will explore how AI helps you maximize the potential that hypervisor partnership offers. This year, we will focus on how the cloud industry is preparing for the AI evolution in terms of technology, oversight, economics, and morality.

SUSECON 2020

Date: March 23–27, 2020

Location: Dublin, Ireland

Website: <https://www.susecon.com/>

At SUSECON 2020, you will learn the latest developments in enterprise-class Linux, Ceph storage, Kubernetes, Cloud Foundry, and other open source projects from technical experts, ecosystem partners, and your peers from around the world.

Events

AI Hardware Summit	March 10-11	Munich, Germany	https://www.aihardwaresummit.eu.com/events/ai-hardware-summit-europe
Cloud Expo Europe	March 11-12	London, United Kingdom	https://www.cloudexpo-europe.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
SUSECON 2020	March 23-27	Dublin, Ireland	https://www.susecon.com/
KubeCon + CloudNativeCon Europe 2020	March 30-April 2	Amsterdam, Netherlands	https://events.linuxfoundation.org/kubecon-cloudnativecon-europe/
Open Networking & Edge Summit North America	April 20-21	Los Angeles, California	https://events.linuxfoundation.org/open-networking-edge-summit-north-america/
Strata Data Conference	April 20-23	London, United Kingdom	https://conferences.oreilly.com/strata-data-ai/stai-eu
DevOpsCon	April 21-24	London, United Kingdom	https://devops.jaxlondon.com/
LinuxFest Northwest	April 24-26	Bellingham, Washington	https://linuxfestnorthwest.org/conferences/2020
Linux Storage, Filesystem and Memory Management Summit	April 27-29	Palm Springs, California	https://events.linuxfoundation.org/lsmfmm/
IcingaConf 2020	May 12-14	Amsterdam, Netherlands	https://icingaconf.com/
Linux Presentation Day 2020	May 16	Cities across Europe	https://l-p-d.org/en/start
DrupalCon Minneapolis 2020	May 18-22	Minneapolis, Minnesota	https://events.drupal.org/minneapolis2020
OSCON (Open Source Software Conference)	June 13-16	Portland, Oregon	https://conferences.oreilly.com/oscon/oscon-or
Software Architecture Conference	June 15-18	Santa Clara, California	https://conferences.oreilly.com/software-architecture/sa-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

Paul Brown	92
Zack Brown	12
Bruce Byfield	42, 62
Joe Casad	3
Mark Crutch	65
Marco Fioretti	46, 86
Jon "maddog" Hall	67
Charly Kühnast	40
Christoph Langner	20, 68, 74
Rubén Llorente	24
Vincent Mealing	65
Pete Metcalfe	52
Graham Morrison	78
Dr. Roland Pleger	56
Mike Schilli	34
Mayank Sharma	16
Alexander Tolstoy	30
Jack Wallen	8

Contact Info

Editor in Chief

Joe Casad, jcasad@linux-magazine.com

Copy Editors

Amy Pettie, Megan Phelps

News Editor

Jack Wallen

Editor Emerita Nomadica

Rita L Sooby

Managing Editor

Lori White

Localization & Translation

Ian Travis

Layout

Dena Friesen, Lori White

Cover Design

Dena Friesen

Cover Image

© tashatuvango, 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

While every care has been taken in the content of the magazine, the publishers cannot be held responsible for the accuracy of the information contained within it or any consequences arising from the use of it. The use of the disc provided with the magazine or any material provided on it is at your own risk.

Copyright and Trademarks © 2020 Linux New Media USA, LLC.

No material may be reproduced in any form whatsoever in whole or in part without the written permission of the publishers. It is assumed that all correspondence sent, for example, letters, email, faxes, photographs, articles, drawings, are supplied for publication or license to third parties on a non-exclusive worldwide basis by Linux New Media USA, LLC, unless otherwise stated in writing. Linux is a trademark of Linus Torvalds.

All brand or product names are trademarks of their respective owners. Contact us if we haven't credited your copyright; we will always correct any oversight.

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, Ziebländstr. 1, 80799 Munich, Germany.

Approximate	
UK / Europe	Mar 07
USA / Canada	Apr 03
Australia	May 04
On Sale Date	

Multimedia Tricks

The versatile Linux supports a delightful and diverse array of tools for digital media. Next month we explore the Gnome Cast to TV extension and study how to dial up Spotify from the command line.

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>

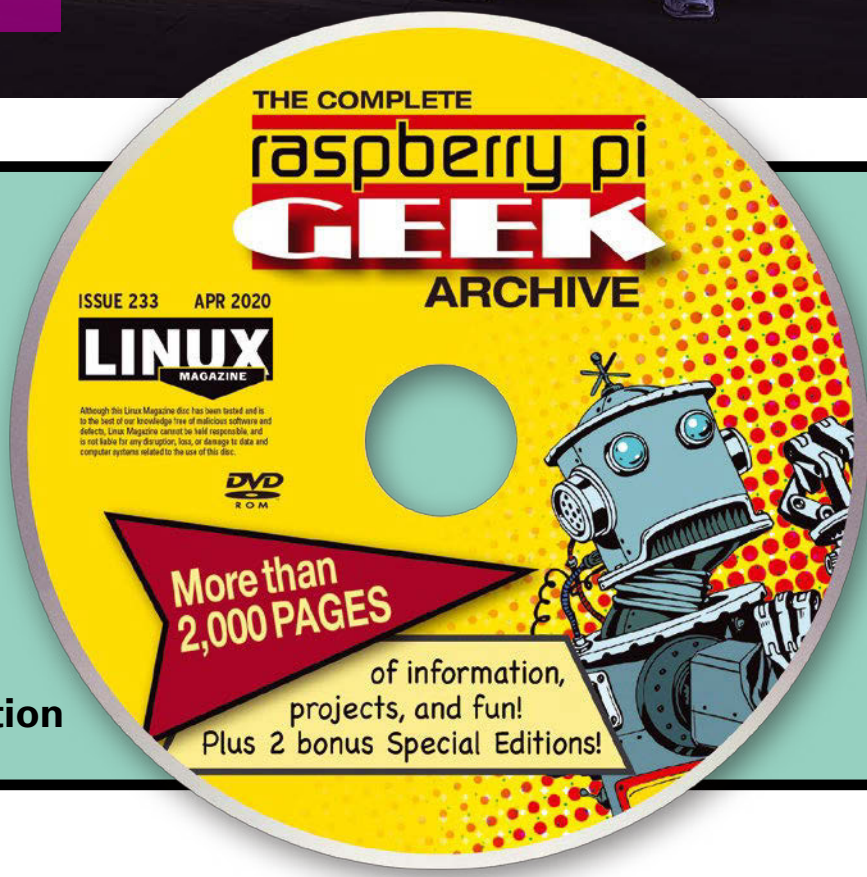


Lead Image © Sergey Nivens, 123RF.com

SPECIAL DVD RELEASE!

Along with our usual mix of technical articles, issue 233 includes the **Complete Raspberry Pi Geek Archive DVD**. This exclusive DVD includes every issue of *Raspberry Pi Geek* and 2 bonus Special Editions – *Raspberry Pi Adventures* and *Raspberry Pi Handbook*.

Secure your copy by subscribing now to the Linux Magazine DVD edition at: bit.ly/Linux-Magazine-subscription



CLOUDFEST

See The Future

in 20/20

March 14 - 19, 2020

EUROPA-PARK | RUST | GERMANY

www.cloudfest.com



REGISTER FOR FREE!
CODE: **fvV5HGrf**

CloudFest Themes

The Intelligent Cloud
An examination of AI in the
Cloud ecosystem

Web-Pros in the Cloud
Smart Creativity at Scale

Hone your skills with special editions!

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

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

Available in print or digital format

Check out the full library!

shop.linuxnewmedia.com

