

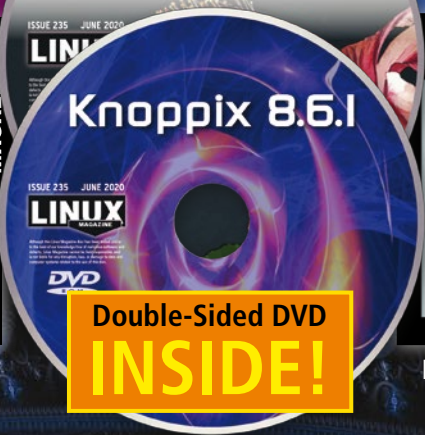
FREE
DVD

OPENMANDRIVA
LX PLASMA 4.1
64-BIT

ROCK THE SCIENCE FAIR
Read and visualize weather
data with MetPy

**WHAT'S NEW
IN SYSTEMD**

LINUX
MAGAZINE



Double-Sided DVD
INSIDE!

LINUX **PRO**

MAGAZINE

ISSUE 235 – JUNE 2020

WHAT'S NEW IN SYSTEMD

Reinventing home directories
with the powerful systemd-homed

Vagrant

Create a custom UI for
managing virtual machines

Glances

Keep up-to-date on system health

PlantUML

Create diagrams using
human-readable text



*"The world around us may be
going through strange times,
but at least so far kernel
development looks normal."*

– Linus Torvalds (see Kernel News)

Tangram

Manage your social
media accounts from
a single interface

BerryLAN

Set up a headless
Rasp Pi over
Bluetooth

LINUXVOICE

Tutorials

- CalDAV Calendar Server
- LÖVE Gravity

- **sncli**: Sync and manage your notes
- **maddog**: It's Time to Innovate



FOSSPicks

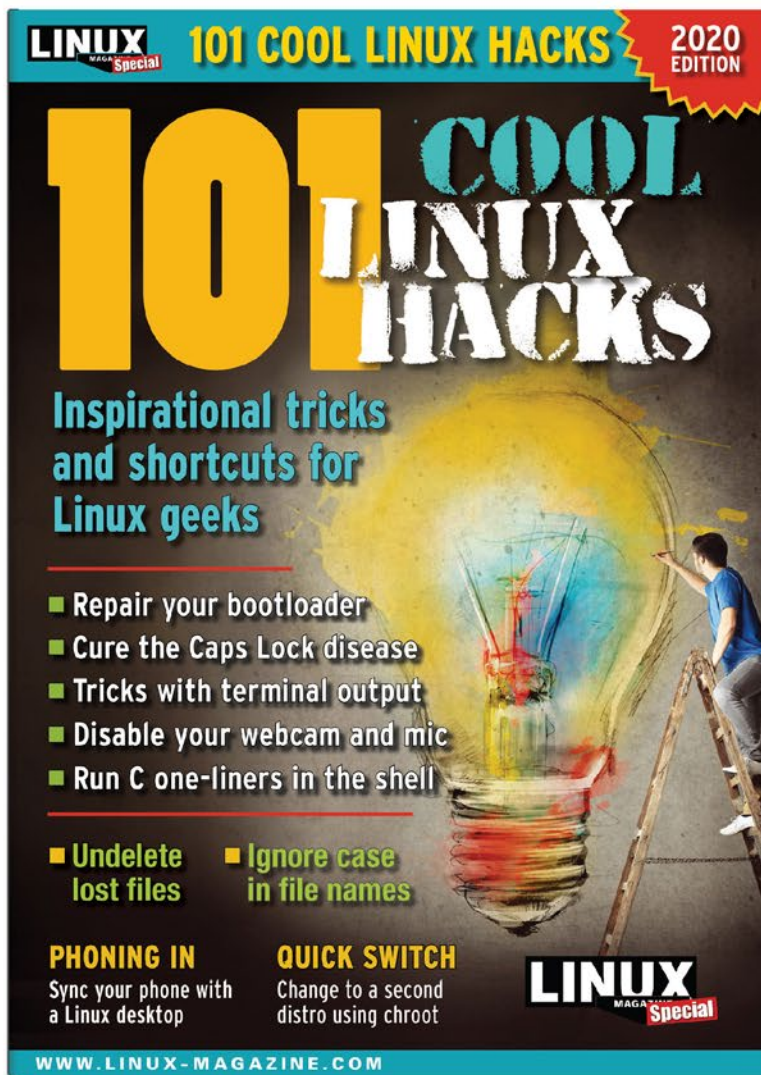
- Radio Scanning
- Session Instant Messenger
- Photoflare



WWW.LINUXPROMAGAZINE.COM

SHOP THE SHOP
shop.linuxnewmedia.com

GET PRODUCTIVE WITH 101 LINUX HACKS



Improve your Linux skills with this cool collection of inspirational tricks and shortcuts for Linux geeks.

- Undelete lost files
- Cure the caps lock disease
- Run C one-liners in the shell
- Disable your webcam and mic
- And more!

ORDER ONLINE:
shop.linuxnewmedia.com/specials

DOMAIN DRAMA

Dear Reader,

Some dramatic stuff is going down right now for a few of the powerful organizations that run the Internet. The .org top-level domain, as you probably know, is home to many charities and nonprofit organizations. Traditionally, the price of a .org domain has been kept low to make it affordable for small-time community organizations and charities. But sometime last year, the Internet Society (ISOC) announced a plan to remove the price caps. They put the question up for public comment and received more than 3,000 comments opposing this idea and 6 in favor of it. In spite of the resounding level of disapproval, they went ahead with the plan. A new contract that removed the price caps was announced in July 2019.

Before going on I should expand a few of these nested acronyms: The Internet Corporation for Assigned Names and Numbers (ICANN) is a nonprofit corporation that contracts with the nonprofit Internet Society (ISOC) for maintaining the .org namespace. ISOC created a nonprofit corporation called the Public Interest Registry (PIR) to manage .org. So PIR was the entity that actually removed the price caps.

No sooner had the dust settled from the price-cap announcement (relatively speaking – some would say the dust never settled), but anyway, sometime in November, the ISOC had another surprise: they announced they were going to sell PIR, along with the responsibility for the .org namespace, to a private equity firm. Why would a private equity company buy a nonprofit corporation? It soon emerged that the company, Ethos Capital, intended to transform PIR from a nonprofit to a for-profit corporation, and they were willing to pay US\$1.1Billion to get control of PIR and the .org domain. Some of that money would be put up in cash, and the rest of it, approximately US\$300Million, would be a loan leveraged against PIR.

It turns out that lifting the price caps for .org suddenly expanded PIR's potential for revenue, because they could raise the rates charged to the organizations that register .org domains. The .org registry was suddenly much more valuable, and most of the world hadn't realized it yet, which is *exactly* the kind of pool that private equity firms like to swim in. But it all happened so quickly.

Observers began to ask, "Who is Ethos Capital," and what are they doing here? According to the reports [1], former ICANN CEO Fadi Chehade personally registered the domain used by Ethos Capital exactly one day after ICANN announced that it was approving the plan to lift the .org price caps. Another Ethos Capital employee, Chief Purpose Officer Nora Abusitta-Ouri, once served as Senior Vice President of ICANN and is a long-term associate of Chehade. In other words, the value of

the .org domain radically increased, and a private company associated with a pair of former ICANN executives bought it through a closed-door contract.

It was enough to attract attention, and it did attract the attention of California attorney general Xavier Becerra. (The ICANN headquarters is located in California, so ICANN is subject to California law.)

Becerra intervened in January to ask for more information, and, as this issue went to press in April, he sent a scathing letter to the ICANN president and CEO urging them to reject the deal.

The attorney general pointed out the ICANN and ISOC incorporation documents call for them to act for the benefit of the Internet community as a whole, but their actions appear contrary to this mission. He also noted that the financial picture following the sale is still unclear. PIR is financially stable now – will it still be after it loses its nonprofit tax-exempt status and takes on \$300Million in debt? Other concerns arise in a flurry of questions. "If ISOC was concerned about diversifying its revenue streams, what did ISOC do, if anything, before deciding to sell the .ORG registry agreement? Why did ISOC not conduct a competitive bid process for a new registry operator if it wanted a change in the registry operator? Did ISOC explore options other than a sale to a private equity firm, given that its nonprofit status was key to PIR becoming the .ORG registrar? What consultation, if any, did ISOC conduct with its stakeholders prior to proceeding with the proposed sale?" [2]

I must admit, there is something almost surreal about a leveraged buyout of a nonprofit that maintains the domain name registry for other nonprofits, but the whole controversy could be over by the time you read this message. ICANN has said it will delay the purchase until May 4 in order to consider the arguments raised in the attorney general's letter. [3]

I think it is fair to add (and most of the commentaries are not so generous) that it is possible the people involved with this deal could have been well intentioned. It appears that ISOC had the goal of establishing an endowment using the \$1.1Billion in proceeds from the sale, so by giving up .org, they would be shoring up other parts of their operation and creating some kind of permanent funding source (the details of which are yet undisclosed). ISOC has also argued that it has put safeguards in place to protect the public interest.

But even if this whole deal is not as bad as it looks, still, it is safe to say that the process lacked transparency. Any deal this big should have been negotiated openly with competitive bids, and the insider relationship of former ICANN employees should have caused the participants to err on the side of *more* information and not *less*.

Whatever the outcome, I hope the Internet community learns something from this dysfunctional divestiture. Nonprofits need open processes to maintain public trust, and any nonprofit, Internet or otherwise, should practice some healthy skepticism whenever a private equity firm offers a helping hand.



Joe Casad,
Editor in Chief

Info

- [1] Internet World Despairs as Non-Profit .Org Sold for \$\$\$ to Private Equity Firm, Price Caps Axed: https://www.theregister.com.uk/2019/11/20/org_registry_sale_shambles/
- [2] Becerra's Letter to ICANN: <https://www.icann.org/en/system/files/correspondence/becerra-to-botterman-marby-15apr20-en.pdf>
- [3] ICANN Delays .Org Sale Again After Scathing Letter from California AG: <https://arstechnica.com/tech-policy/2020/04/icann-delays-org-sale-again-after-scathing-letter-from-california-ag/>

LINUX MAGAZINE

WHAT'S INSIDE

Systemd is a mystery that keeps on giving. Now a new feature of the leading Linux init system will change the way you think about user home directories. This month we take a closer look at systemd-homed.

Also inside:

- **Vagrant** – cool tool for managing and provisioning virtual machines (page 22).
- **Tangram** – integrate social media portals in a single convenient interface (page 52).

Then turn to MakerSpace for a look at the MetPy Python science library, and check out LinuxVoice for a tutorial on building a 2D computer game with the Lua-based LOVE framework.

SERVICE

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

NEWS

08 News

- Nextcloud Partners with IONOS
- Linux to Get High Resolution Wheel Scrolling
- KDE Developers Are Working on a TV Interface
- System76 is Developing a New Keyboard
- Embedded Linux Joins the Fight Against COVID-19
- AWS Launches a New Linux Distribution

11 Kernel News

- Kernel Gatherings Canceled Due to COVID-19
- Kernel Meta-Organizations Changing Due to COVID-19
- Minimum Version Numbers for Build Tools

14 Interview – The Xen Project

Lars Kurth of the Xen Project talks about trends, markets, and the project's various threads of development.

COVER STORIES

18 systemd-homed

Systemd has already changed almost everything about the Linux startup process. Now an experimental new feature takes on the challenge of user home directories.



IN-DEPTH

22 Vagrant

With Vagrant, you can automate the creation and management of consistent virtual machines that work across platforms.



28 Glances

Admins and power users like to watch the load on their computers. Glances lets you see immediately if something is wrong.



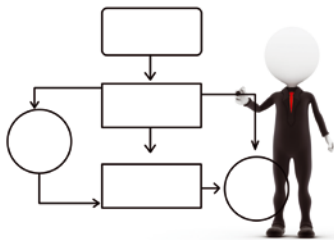
IN-DEPTH

32 Command Line – rkhunter

The Rootkit Hunter script efficiently checks for malware, with the potential to detect over 240 rootkits.

36 PlantUML Diagrams

Create diagrams using human-readable text and reuse them anywhere.



44 Programming Snapshot – Performance Analysis

Experienced sys admins always use the same commands to analyze problematic system loads on Linux. Mike Schilli bundles them into a handy Go tool that shows all the results at a glance.

49 Charly's Column – traceroute

Like every admin, Charly regularly uses the classic traceroute tool. If unfriendly digital natives interfere with an ICMP filter, he simply switches to a clever alternative like LFT.

50 Universal Package Systems

Billed as the future of package management, universal package systems like Snappy and Flatpak have failed to live up to their promise.

52 Tangram

Track social media portals like Facebook and Twitter, as well as web-based messengers like Whatsapp and Telegram, in a single application window.



56 ShellHub

This innovative app offers remote access to your network with minimal reconfiguration of a firewall.



SEE PAGE 6 FOR DETAILS

MAKERSPACE

62 MetPy

Read, visualize, and perform calculations with weather data.

66 BerryLan

Manage a headless Rasp Pi over a Bluetooth connection.



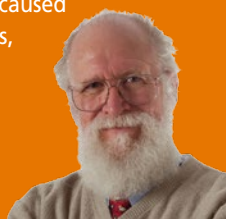
LINUXVOICE

69 Welcome

This month in Linux Voice.

70 Doghouse – United We Stand

It's time to innovate – in the midst of the disruptions caused by the coronavirus, let's find ways to keep everyone communicating, working, and learning.



72 Simplenote and sncli

If you're using Simplenote, check out sncli, a Python-based tool for syncing and managing your notes.

75 Tutorial – CalDAV/CardDAV

You can manage your calendars and address books with the CalDAV/ CardDAV standards, Nextcloud, and a few open source tools.

84 FOSSPicks

This month Graham looks at navi, SDRangel, Photoflare, Ikona, emulator-sun-2, Quantum Tetris, Selfless Heroes, and more!



90 Tutorial – LÖVE Physics

Video game animation is not simply a matter of making your characters move – you also have to consider the physics.

On the DVD

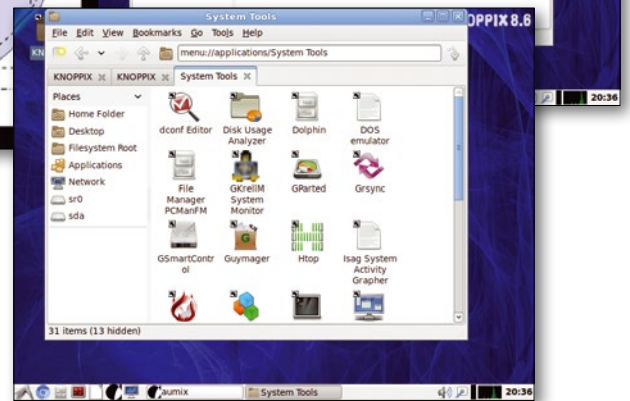


Knoppix 8.6.1

Knoppix is the quintessential Live Linux. Place this disc in your DVD drive, and boot to a full graphical interface with a vast collection of user applications and admin tools. Seasoned sysadmins use Knoppix to troubleshoot downed Linux and Windows systems. The latest version is based on Debian 10 "Buster" and comes with support for both 32-bit and 64-bit architectures.

OpenMandriva Lx Plasma 4.1

OpenMandriva is a free community Linux that lives on as a descendant of the once-popular Mandriva commercial edition. The version included on this disc comes with KDE Plasma 5.17.5, Linux kernel 5.5.0, and the Calamars 3.2.17 distro-independent installer.



Additional Resources

- [1] Knoppix:
<http://knopper.net/knoppix/index-en.html>
- [2] Knoppix 8.6.1 Release Notes:
<http://knopper.net/knoppix/knoppix861-en.html>
- [3] OpenMandriva:
<https://www.openmandriva.org/>
- [4] OpenMandriva Documentation:
<https://wiki.openmandriva.org/en/home>
- [5] OpenMandriva Forum:
<https://forum.openmandriva.org/>

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

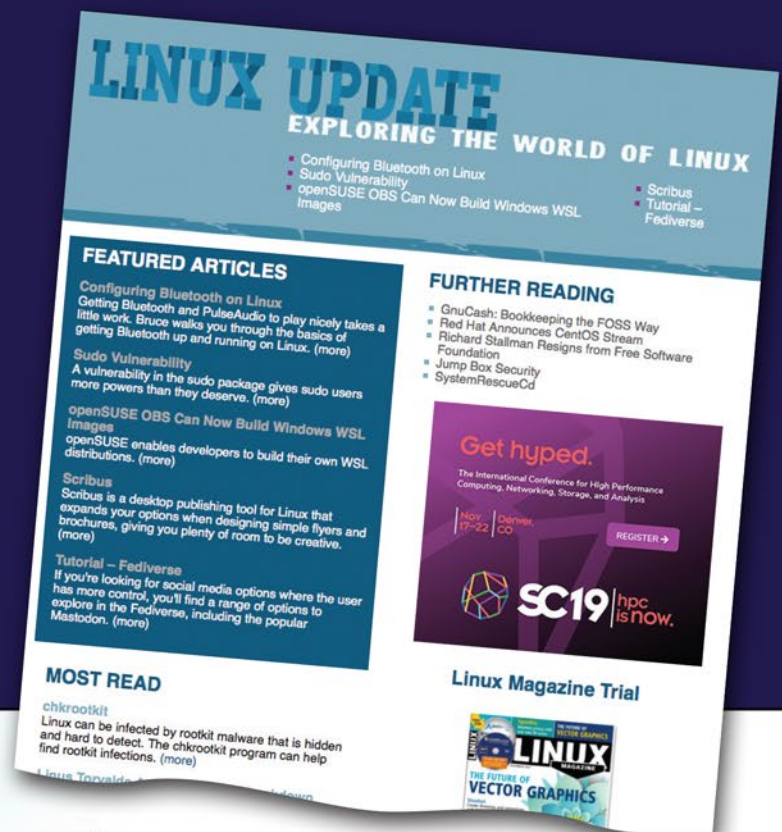
Although this Linux Magazine disc has been tested and is to the best of our knowledge free of malicious software and defects, Linux Magazine cannot be held responsible, and is not liable for any disruption, loss, or damage to data and computer systems related to the use of this disc.

LINUX UPDATE

Need more Linux?

Our free Linux Update newsletter delivers insightful articles and tech tips to your mailbox twice a month. You'll discover:

- Original articles on real-world Linux
- Linux news
- Tips on Bash scripting and other advanced techniques
- Discounts and special offers available only to newsletter subscribers



bit.ly/Linux-Update

NEWS

Updates on technologies, trends, and tools

THIS MONTH'S NEWS

- 08 • Nextcloud Partners with IONOS
- Linux to Get High Resolution Wheel Scrolling
- 09 • KDE Developers Are Working on a TV Interface
- System76 is Developing a New Keyboard
- More Online
- 10 • Embedded Linux Joins the Fight Against COVID-19
- AWS Launches a New Linux Distribution

Nextcloud Partners with IONOS

In times of crisis, open source projects step up big. That's exactly what Nextcloud has done. In the current climate, companies have had to quickly migrate to cloud solutions, only to find themselves bumping up against serious privacy and security issues. Because of the 2018 CLOUD act, authorities could obtain data without prior judicial review for this request.

With so many companies and employees having to migrate from their in-house cloud platforms and turn to various third-party cloud services, Nextcloud and IONOS have come together to ensure sovereignty over customer

data. Because both companies are housed in Germany, anyone using the Nextcloud platform on IONOS is guaranteed maximum protection against the US CLOUD Act.

Achim Weiß, CEO of IONOS, had this to say about the partnership:

"Our cooperation, therefore, gives Nextcloud customers the legal security they need. The consistent use of open standards ensures transparency. Anyone can view the code at any time, check it for security gaps and change it if necessary. Moreover, only on an open-source basis is it easy to link data and applications with other systems."

This solution isn't just a response to the US CLOUD Act. In fact, Nextcloud has pulled this off to help lower the barriers to entry and offer a reliable, compliant and safe place to work online. With a fully managed Nextcloud Hub, users can enjoy document editing, file sharing, groupware, audio/video chat, and much more.

Even better, this particular Nextcloud solution is hosted on IONOS, so users don't have to concern themselves with installing the cloud platform software. To find out how to sign up for an account, check out the official Nextcloud IONOS sign up page (<https://www.ionos.com/cloud/cloud-apps/nextcloud>), where you can get an account for as little as \$0.0069/hour with a max of \$5.00 USD per month (for an XS account).



© Brian Jackson; 123RF.com

Linux to Get High Resolution Wheel Scrolling

The Linux desktop has come a very long way in a short time. But there are a couple of features that lag behind the likes of macOS – such as multi touch gestures and smooth wheel scrolling. That all began to change about a year ago, when high resolution wheel scrolling was merged into the mainline Linux kernel, by adding new axes REL_WHEEL_HI_RES and REL_HWHEEL_HI_RES. However, since that kernel addition, work on the feature fell to the wayside.

Around the same time as support was added to the kernel, Peter Hutterer began working on integrating high resolution mouse wheel scrolling support into Wayland. However, that work also ground to a halt and nothing came of Hutterer's efforts.

Good news, Linux desktop users, Mr. Hutterer has resumed work on getting “buttery smooth” implemented into Wayland. The work is happening by way of lib-input, Wayland, Mutter, GTK, and XWayland. Although this new feature won’t make it into Ubuntu 20.04 or Fedora 32, it is possible to add this feature into the latest iteration of Fedora, using the repository found on the Fedora COPR site for the project (<https://copr.fedorainfracloud.org/coprs/whot/high-resolution-wheel-scrolling/>).

Once this feature is finally rolled out, the Linux desktop experience will be much improved.

To find out more about this project, follow Peter’s blog (<https://who-t.blogspot.com/>).

KDE Developers Are Working on a TV Interface

Never one to remain stagnant, the developers of the KDE desktop are hard at work creating what they have dubbed “Plasma Bigscreen.” This new project has one goal – to develop a user interface aimed at television screens.

This new interface will also integrate with the open source Mycroft AI voice assistant to create a smart TV platform that will include full voice control and can be expanded with Mycroft “skills.” The platform will be free, open source, innovative, and community supported. Out of the box, Big Plasma will include some simple skills, such as the YouTube Voice Application, which allows users to interact with YouTube via voice command.

Plasma Bigscreen will also include the Aura Browser, based on the QtWebEngine. This browser has been designed to work completely with arrow key navigation, so you won’t need a mouse to control the app (just your remote). In fact, the entire

Plasma Bigscreen interface is intended to be easily used via remote control, and includes experimental support for HDMI-CEC (HDMI Consumer Electronics Control).

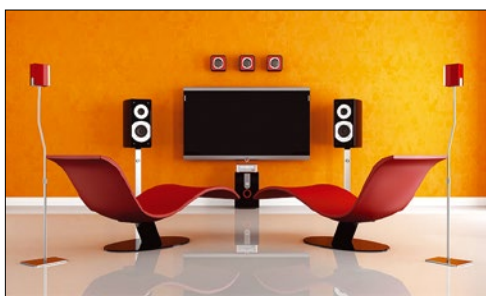
Plasma Bigscreen is intended to be used with a TV, but will also work on a regular monitor. The developers already have the platform working on a Raspberry Pi 4. Although the project is intended

for small computing platforms, it will be able to run on just about any computer.

Anyone with a Raspberry Pi 4 can download the beta image.

Original announcement: <https://dot.kde.org/2020/03/26/plasma-tv-presenting-plasma-bigscreen>

© Paolo De Santis, 123RF.com



System76 is Developing a New Keyboard

System76 (<http://www.system76.com/>) is famous for designing and developing some of the most powerful Linux-based computers on the planet. Never one to rest on reputation, System76 is constantly innovating. This time around, they are focusing their efforts on improving a device we all take for granted – the keyboard.

This new keyboard is being designed with Linux and Linux users in mind. In fact, according to Carl Richell, CEO of System76, “Auto tiling in our upcoming Pop!_OS 20.04 release is designed to work extremely well with this keyboard, and I think that people are going to really respond to it...” This new keyboard will make typing much more comfortable. For instance, the keypad has been completely removed, so the mouse can be moved closer to where your hands rest.

Another change to the standard keyboard design is that the spacebar has been significantly reduced. Of this change, Richell says “we’ve found that spacebars typically, for example, are way too long, which means your strongest digit, your thumb, isn’t very useful.” To that end, the layout will drastically change.

This keyboard will also be highly configurable. The entire keyboard consists of keys that are only three different sizes. That means you will be able to swap them out to create a custom layout perfectly suited for your needs.

MORE ONLINE

Linux Magazine

www.linux-magazine.com

ADMIN HPC

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

MPI Apps with Singularity and Docker

• Jeff Layton

The Message Passing Interface (MPI) is one of the frameworks used in high-performance computing (HPC) for a wide variety of parallel computing architectures.

ZFS Tuning for HPC

• Petros Koutoupis

If you manage storage servers, chances are you are already aware of ZFS and some of the features and functions it boasts.

ADMIN Online

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

Monitor and Optimize Fibre Channel SAN Performance

• Roland Döllinger

We discuss the possible bottlenecks in Fibre Channel storage area networks and how to resolve them.

Build Operating System Images on Demand

• Martin Loschwitz

If you are looking for a way to build images quickly and easily, FAI.me is the place to go.

Linux nftables Packet Filter

• Thorsten Scherf

The latest nftables packet filter implementation, now available in the Linux kernel, promises better performance and simpler syntax and operation.

System76 hopes to bring this new ground-breaking keyboard to market late summer, 2020.

Original source: <https://blog.system76.com/post/612874398967513088/making-a-keyboard-the-system76-approach>

Embedded Linux Joins the Fight Against COVID-19

There are new devices on the market, working with embedded Linux, to help combat the growing pandemic. One such device is the Kogniz Health Cam (<https://www.kogniz.com/health>), a camera that can scan groups of people walking through an entrance and check for temperatures from 16 feet away. Another similar device is the Raspberry Pi-based FluSense (<https://www.raspberrypi.org/blog/flusense-takes-on-covid-19-with-raspberry-pi/>), which uses machine learning to detect coughing and crowd size and analyze the data to monitor flu-like symptoms. Both devices are capable of leveraging Linux to aid in the battle against COVID-19.

In fact, there are a number of open source projects that are working for this cause. For instance, Chai's Linux-driven BeagleBone-based Open qPCR (<https://www.chaibio.com/openqpcr>) is a Coronavirus Environmental Testing Kit which can test surfaces for COVID-19 from swab samples.

Another project is Opentrons (<https://opentrons.com/>) lab automation platform which is currently being adapted for COVID-19 testing on humans. Because Opentrons have open sourced their devices, the specs and code for the apps, protocols, and hardware are publicly available (<https://github.com/Opentrons>).

There are also two open source projects dedicated to solving the ventilator problem. The first is OpenLung BVM Ventilator (<https://gitlab.com/open-source-ventilator/ventilator/OpenLung>). This is an open collaboration between OpenLung and OpenSourceVentilator to produce a low resource, quick deployment ventilator design that utilizes a bag valve mask as its core component. The second is the Open Source Ventilator Project (<https://simulation.health.ufl.edu/technology-development/open-source-ventilator-project/>), created to address the predicated ventilator shortage. For this project, you can download the specs for all the modules and grab the source from Github (https://github.com/CSSALTLab/Open_Source_Ventilator).

AWS Launches a New Linux Distribution

Amazon's cloud platform (AWS) has created and released a new distribution of Linux aimed at container deployments for bare metal and virtual machines. The new operating system, Bottlerocket (<https://aws.amazon.com/bottlerocket/>), is still in the developer review phase, but can be tested as an Amazon Image Machine for EC2.

This purpose-built Linux distribution supports all images that follow the Open Container Initiative (<https://www.opencontainers.org/>) image format (such as Docker images) and uses a read-only file system for security and integrity. To further bolster the security of the platform, SSH access is discouraged and only available through the Bottlerocket admin container tool (<https://github.com/bottlerocket-os/bottlerocket-admin-container>).

Bottlerocket shrugs off the standard update process in favor of automatic image-based updates by way of an orchestration service, such as Amazon EKS. The single step update process reduces management overhead and improves uptime for container applications by minimizing update failures and enabling easy rollbacks.

The Bottlerocket OS offers much-improved resource usages because it contains only the essential applications and services to run containers. This means Bottlerocket is a purpose-built platform and not intended for general usage.

Once Bottlerocket has been released for general usage, it will be supported for three years. Already, Bottlerocket has a number of interested partners, such as Alcide, Armory, CrowdStrike, Datadog, New Relic Sysdig, Tigera, Trend Micro, and Waveworks.

Original announcement: <https://aws.amazon.com/blogs/aws/bottlerocket-open-source-os-for-container-hosting/>



**Get the latest news
in your inbox every
two weeks**

**Subscribe FREE
to Linux Update
bit.ly/Linux-Update**

Zack's Kernel News

Kernel Gatherings Canceled Due to COVID-19

There are normally many Linux and open source conferences throughout the year. These have started to see some disruption due to the COVID-19 pandemic. In mid-March, Josef Bacik announced that one of the storage, memory, and networking summits would not be happening in April as planned. He said:

"We currently do not have concrete plans about how we will reschedule; the Linux Foundation is working very hard at getting us alternative dates as we speak. Once the new plans are concretely made, we will notify everyone again with the new plans."

"The tentative plan is to keep the attendees as they are if we reschedule within 2020. This includes anybody that declined for travel-related concerns. We will re-send all invitations again to the original invitees so it's clear that you have been invited."

"If we have to reschedule into 2021, then we will redo the CFP once we are closer to the actual date again and redo all of the invites and topics so we're as up to date as possible with the current state of the community."

"We will keep the current program committee, and I will continue to chair until we have the next LSF/MM/BPF."

Kernel Meta-Organizations Changing Due to COVID-19

In mid-March, Laura Abbott announced some changes to the Linux Foundation's Technical Advisory Board (TAB) charter that affected the way the 10 members were elected. In particular, there would be provisions for absentee voting, in case community members were not able to attend the events where elections were traditionally held. There would also be an expansion of voter eligibility, to include more than just the invitees to the Kernel Summit events.

The TAB is a very interesting construct. Technically part of the Linux Foundation, it typically consists of high-powered members of the kernel development community itself, acting as a liaison between

the developer community and the Linux Foundation Board of Directors.

A number of folks had issues with Laura's changes, having to do with ballot stuffing and other potential cases of voter fraud. As Theodore Y. Ts'o pointed out, this issue is a lot more important now than it was five years ago. Nowadays, given the ongoing assaults on elections everywhere, it's not inconceivable that something like TAB elections could come under attack.

Ted said, "we need to be smart about how we pick the criteria [for identifying eligible voters]. Using a kernel.org account might be a good approach, since it would be a lot harder for a huge number of sock puppet accounts to meet that criteria. We don't have a final proposal for something which can be objectively measured, but can't be easily gamed by someone who is trying to subvert the system. It is pretty clear, though, that we need to have that clearly articulated, in writing, *before* we start the nomination for the next round of TAB candidates."

Ted also pointed out that there was probably no choice but to allow absentee voting, given the COVID-19 pandemic. He said, "it's only responsible to consider what we should do if in fact health and safety restrictions are such that we might not be able to hold *any* Linux systems conferences in 2020."

He proposed that "in that case, we might be forced to either keep TAB members in place beyond their original elected term, or we might have to go to a pure electronic voting for the upcoming TAB election. I very much hope that won't be the case, but we need to be prepared for that eventuality."

Jani Nikula pointed out some problems with requiring a kernel.org email address. He said, "it's at the kernel.org admins discretion to decide whether one is ultimately eligible or not. Do we want the kernel.org admin to have the final say on who gets to vote? Do we want to encourage people to have kernel.org accounts for no other reason than to vote? Furthermore, having a kernel.org account imposes the additional requirement that



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

"The world around us may be going through strange times, but at least so far kernel development looks normal."

– Linus Torvalds, March 23, 2020

Author

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

you're part of the kernel developers web of trust, i.e. that you've met other kernel developers in person. Which is a kind of awkward requirement."

But Greg Kroah-Hartman said that in fact, the whole point of requiring a kernel.org account was to make sure voters were part of the developer web of trust. He said, "That is the goal here; if you know of some other way to determine this, please let us know. We went through many iterations of this and at the moment, it is the best we can come up with."

He added, "Also, note that the 'kernel.org admin' is really a team of people who have been doing this for 9 years, it's not a single person responsible for giving out new accounts to people that do not meet the obvious requirement levels as published on kernel.org."

Greg also did acknowledge, "These are 'baby steps' we are taking here, to try to allow for remote voting. We are not saying this is the end-all-be-all solution, but you have to give Laura credit for coming up with this as 'better than nothing' which is what has been the case for the past decade or so."

Several folks pointed out that the charter changes were always an expansion of the voting pool, not a constriction of it. As Steven Rostedt said, "We are trying to extend who can vote beyond those that the charter allows. We are not preventing those that can vote under the current rules from voting. IUC, we are trying to create absentee voting which we never had before. Thus, you can either vote the current way by getting [to] travel to wherever Kernel Summit is and attending the conference, or we can extend the charter so that if you can not come for whatever reason, you have an option to vote remotely, if you satisfy the new requirements. Remember, not attending means you do not satisfy the current requirements."

Steven added, "The TAB has bikeshed this a bit internally, and came up with the conclusion that kernel.org accounts is a very good 'first step'. If this proves to be a problem, we can look at something else. This is why we are being a bit vague in the changes so that if something better comes along we can switch to that. After some experience in various methods (if we try various methods), we

could always make whatever method works best as an official method at a later time."

At a certain point in the conversation, James Bottomley offered some historical context. He said, "When the TAB charter was written (in 2006), the original reason was to prevent manipulation (real or imagined) by the committee who would then become the arbiters of nominations and thus able to influence who might run for the TAB. There are a couple of reasons for the electorate clause: When the TAB was formed, it was done by the kernel developers unhappy at the way OSDL (precursor organization to the LF) was behaving with regard to the kernel, who forced their way onto its board and formed the TAB to gain input and control on behalf of kernel developers, so the TAB was formed by kernel developer[s] for kernel developers and, since most other non-kernel open source groups had their own foundation like entities, keeping it kernel only wasn't seen as a problem. The other reason was that OSDL was a bit unhappy to be reformed in this way, and we foresaw that one way to dilute the reforming influence of the TAB would be to dilute kernel developer representation since they were the main community interested in that reform. When the OSDL became the LF, some of the initial antagonism and need for reform went away, and the elections were opened to the co-located conferences as a sign of improved trust."

Minimum Version Numbers for Build Tools

One of Linus Torvalds' development values is that the kernel should be buildable using tools that are very old. This way very old systems can continue to build fresh kernels without having to do major software upgrades. And this is good because not all hardware can tolerate major software upgrades.

So whenever Linux changes the minimum version number required for a given tool, it's a significant event. One of the big tools is the GNU C Compiler (GCC), but lesser tools are also important. Recently Borislav Petkov posted a patch updating the minimum version number for binutils, from version 2.21 to version 2.23.

Binutils is a set of GNU tools that are used specifically for writing software.

They include an assembler, a linker, a profiler, and a lot of other truly amazing developer tools that are absolutely essential for Linux kernel development.

The problem, as patched by Alan Modra in 2012 (!), was in the linker – that part of binutils that combines a bunch of compiled binaries into a single executable program. To do that, the various files need to make reference to key memory locations within one another. And under some circumstances, the older version of the linker was pointing to the wrong locations.

So, as Borislav said, changing the required binutils version number was long overdue. He pointed out that the various Linux distributions already shipped with the fixed version of binutils, so no one using a standard distribution would even notice the new version requirement.

Kees Cook and Jason A. Donenfeld approved Borislav's patch, and Jason pointed out an immediate way to further clean up the kernel source tree, when he remarked, "then we'll be able to use ADX instructions without `ifdefs`."

In fact, Jason advocated bumping up the minimum binutils version number as high as 2.25, saying, "we could get rid of `*all*` of the `CONFIG_AS_*` `ifdefs` throughout."

However, that suggestion got lost in the wind. Bumping up to version 2.23 was already controversial enough. In particular, the timing of the patch was important. Borislav wanted to get the patch in the 5.7-rc development cycle after 5.7-rc1, to ensure it would be too late for the official 5.7 release, but would still get the longest possible period of testing, before coming out in the official 5.8 kernel release.

The goal was to get as much testing done by developers as possible, so that any users who still relied on the earlier version of binutils could be discovered, and any valid arguments not to bump up the version number could be revealed.

The controversy, however, was not about whether to bump up the version number – everyone agreed it was essential. No, the most controversial thing about Borislav's patch was in waiting until Linux 5.8. Both Kees and Jason had hoped to see this change go into the kernel in time for the 5.7 release. Jason said, "I don't think that filing these away in some subsystem

tree will actually result in any more bugs being shaken out, than if we just do this at the very, very beginning of the 5.7 cycle.”

But Borislav remarked wryly, “Are you or Kees going to deal with any fallout from upping the binutils version, rushed in in the last week before the merge window? Because I sure as hell am not. *Especially* if there’s no big difference when it goes in.”

At this point, Linus came into the discussion, saying:

“I think it’s ok. It’s not going to cause any _subtle_ failures; it’s going to cause very clear ‘oh, now it doesn’t build’ errors. No?”

And binutils 2.23 is what, 7+ years old by now and apparently had known failure cases too.

But if there are silent and subtle failures, that might be a reason to be careful. Are there?”

Borislav said he didn’t know of any subtle failures, but better safe than sorry. He remarked, “a lot of hectic testing of last minute fixes in the past have taught me to take my time.”

He acknowledged that binutils 2.23 had been tested for a very long time already, and that probably all would be well. And if Linus wanted the patch to go into 5.7, then who was he to argue. But he did say, “since it doesn’t really matter when the patch goes in – there’s always the next merge window – I would prefer to take our time and have it simpler in -next for max period.”

Meanwhile Kees said he didn’t really mind waiting. He remarked, “I have plenty of other hills to die on, so I have no urgency on this change. I actually thought it had already happened, since it was brought up a while ago. :) I am just excited to see it happen since it unblocks other work I’ve been touching. As long as it’s ‘eventually’, I don’t care when. :)”

Meanwhile Arvind Sankar asked, isn’t this just a documentation patch? No actual code will change, so why is everyone being so careful about it?


Linus remarked that while this patch was indeed just a documentation change, “there’s a second patch that knows that if we have binutils > = 2.22, then we don’t need to check for

AVX2 or ADX support, because we know it’s there.”

But Arvind also pointed out that the Linux kernel already hadn’t supported pre-2.23 binutils since Linux 5.3 – the kernel build would actually break with anything less than binutils 2.23. So, as Arvind put it, “we’ve already gone 3 kernel versions without complaints.”

That really put an end to all further discussion.

But it’s nice to watch the kernel developers try to deal with this sort of change. They all really wanted to make the best choice that would maximize the possibility of identifying absolutely any user whose system might break as a result. So OK, Arvind gave us a smile at the end, realizing the entire discussion was moot. But it shows the sort of things that really matter in kernel development. Reliability.

Contrast that with so many devices and systems in the commercial world, where products go into end-of-life solely in order to force users to buy the next product. Linux has a very different philosophy, and it shows in the raw ubiquity of Linux. It is truly everywhere. 

DevOpsDays

28 - 29 OCT 2020

CALL FOR PAPERS

runs until July 5

devopsdays.org





The Xen Project

The Big Picture

Lars Kurth of the Xen Project talks about trends, markets, and the project's various threads of development. *By Mayank Sharma*

There's nothing like someone's passing to get one to reflect on the fickle nature of life. The Xen Project's [1] very lively chairperson, Lars Kurth, sadly passed away not long after our interview at the Open Source Summit in November 2019. Lars had been with the project for almost a decade and was instrumental in several pivotal moments in Xen's history, including its move to the Linux Foundation. He conceptualized and executed several key decisions and supervised the significant architectural changes that helped the project go beyond the realm of server virtualization and cloud computing. In his last interview, Lars talked about the project's various threads of development and how Xen is all set to disrupt the auto industry.

Linux Magazine: What's happening with the Xen Project?

Lars Kurth: The big picture, ultimately, right now, is [that] there's a number of different trends happening. And I'm trying to kind of condense this into kind of [a] coherent strategy where different stakeholders in the community benefit from it. First, we have the server virtualization and the cloud market segment. Second, we have a whole segment of users, such as Qubes OS [2], and then there's similar products, which are used by the US military called SecureView. So that [SecureView] is based on open embedded, Xen, and extra bits and pieces of OpenXT [3], and they also [are] kind of

rethinking the way how the platform is working and also rethinking the approach to Xen. So basically the idea there would be to take Xen as it is right now and just reduce it to the absolutely core minimum, basically as a separation hypervisor, but use the same code base, [and] Kconfig it down to something really small.

In the embedded and automotive space, we're seeing quite a lot of traction as well. So, ARM has just announced their safety software reference architecture at ARM TechCon, and that is basically going to be a reference stack of open source components. But they're also working with proprietary vendors at various levels of the stack and at the bottom is going to be an RTOS and a hypervisor, and they basically chose Xen for that. There's going to be significant investment in this area. This means we have to deal with things like safety certification, and how to do this in an open source project. That's the last juicy community problem in open source.

LM: Are you talking about security?

LK: Well, security is easy. Safety certification is different. Let me take a step back. You see companies like Tesla and stuff today; they have self driving car functionality, autopilot, and stuff. Google does the same. They don't have safety certification for this. When you certify your software to certain safety standards, you suddenly have the capability to basically delegate

risk to insurances and stuff like that. If you don't have that – like Tesla – if something goes wrong, they're going to have to pay for everything. So if you don't have a lot of users, and it's still quite experimental, or you want to disrupt the market and have deep pockets, you kind of can go down that route; a whole industry cannot go down that route.

So if you look at the traditional open source software development, well, you get about your code, you test your code, and there's basically a strong collaborative model around it. But it's totally different from how you would approach developing system software [with] safety in mind, because there you start with requirements, and then you break it down into designs, and so on and so forth. Then you reassemble; then you write all your tests, which prove all your claims; then you have a third party, which looks at all your paperwork and all the code and says "well, those guys are saying that they're doing this and are doing this;" and then you get the stamp of approval. This is not easy to implement in open source. But you can do it for a sufficiently small piece of code.

LM: Does this open you up to larger markets?

LK: If we manage to do this (and I'm more than 50 percent confident now that we can do this) and ... do this using Xen to enable downstreams or vendors to then do this themselves in a cost effi-

cient manner, then basically we have a huge new market open, where today open source isn't relevant.

Jim [Zemlin] kind of keeps on bringing this slide in his keynote about which market segments Linux and open source [have] penetrated. You notice, embedded is sort of 45 percent, and everyone else is like above 80 or near to 100, because the 55 percent are safety software. That can only be cracked from a community perspective, if we can adapt our development model to make it compatible with that world. So that's kind of the wide context; these are big themes, and it will take multiple years to follow.

LM: Moving on, what's happening on the cloud server virtualization side?

LK: On the cloud server virtualization side, I think we and everyone else [are] basically chasing hardware vulnerabilities in side-channel attacks. It is forcing us to rethink and rewrite some of the core components of the hypervisor, because we can't trust the hardware anymore. So either we could just go into this mode where we're just going to plug one hole after the next, which is not very efficient, or we can just take a step back and think if we can re-engineer this in some ways that should be generally more resilient towards these kinds of attacks. This is what is happening now.

LM: Any concrete steps you can point out?

LK: One of the things we as a project just did is SUSE rewrote the scheduling framework. So we now have core scheduling in the next Xen release. It's not on its own sufficient to switch hyperthreading back on again, like you remember [in] MDS and L1TF [4]. We have the core-scheduling part in place, and then we either need what we call the secret free Hypervisor or the more synchronized scheduling.

This is how it becomes interesting. We've seen an increase in participation from Amazon. We had our development event in June; they sent nine people. This was the biggest contingent of people from any vendor. I mean, Citrix had eight there. And then they just started presenting some of the ideas publicly, and they started to pick up things which we have to do but which were too slow. One of the things they picked up and have already sent patches for is a group

of functionality which we call a secret-free Hypervisor [5].

LM: What is a secret-free Hypervisor?

LK: So the basic idea there is ... [that] we restructure the way how data is stored, how the memory is laid out, and so on, such as if there is a side-channel attack, you can't get access to anything useful. The first step is to remove the direct map. So that was something that Citrix started picking up. Then now Amazon is running this, and they're able to do this a lot quicker, because they just hired an awful lot of people to work on this. If you look at the top contributors to the project, the traditional sort of split used to be Citrix around 40 percent, SUSE around 20 percent, and then there would be hardware vendors, which were usually in the sort of 7 percent bracket. From what I've seen, I think we will see Amazon to be in our top three, probably displacing SUSE.

LM: Isn't that worrying? Doesn't Amazon have a tendency to fork?

LK: I think they [Amazon] have finally won the battle internally to say "actually we now can probably work with open source." All the people I've worked with are really professional. Also, the other thing is [that] they have started hiring the leadership from the community to continue doing this. So obviously there is a little bit of a worry there, because they are still figuring out their level of commitment [and] haven't proven to me yet that they are trustworthy. But they are going in the right direction, and that's a good thing. Looking at the KVM Forum schedule, I think they seem to be doing the same there. I don't know what this means in the long run, but it does look as if they are gearing up to follow a multitechnology strategy.

LM: Once the Xen Project gets that security certification, will the ball be in your court?

LK: They're [Amazon] really helping on all the security stuff. The most interesting piece [that] they announced – I mean, this is really, really complicated stuff; we had lots of discussions around this at the Developer Summit – is live updating [6]. So basically, the way they're designing this is [that] they can just update the Xen version without rebooting. I mean, if they can pull this off, wow! But you know, this is still in the planning stage, and this is not implemented yet. We are seeing some discus-

sions around this now. And it's kind of really cool. Knowing them, I don't think they would be suggesting this if they couldn't pull it off.

LM: What are you excited about in the next release?

LK: The core piece we're seeing there in Xen 4.13 [7] is the piece around core scheduling. There is actually a talk at 11:30 [at Open Source Summit] by Dario [Faggioli] [8]. He's covering KVM as well, but the KVM folks are quite a bit behind on this. So you can see ... lots of security-related stuff, not so much focused on features with the exception for there's been quite a lot of focus on AMD hardware and stuff like that. Intel's also been doing interesting stuff in the area. For the side-channel attacks, we typically need to update your microcode, and normally you do this when you reboot, right? We've implemented functionality in Xen, which allows you to do this at run time, so you don't have to reboot. This whole theme around security is around two things. One is just trying to get ahead with some of the side channel – existing and [a] few potentially future side-channel related stuff. I think we also have some mitigations for Spectre-v1 now in this release as well. Then the second piece is around, basically, being able to just keep everything up to date without reboots.

LM: Any noteworthy developments on the hardware-support front?

LK: So there's support for the latest AMD Rome CPUs and stuff. Traditionally what would happen in the past is that when a new vendor would bring out a new CPU ... it would actually work on Xen and KVM out of the box. But then we would add features to kind of improve the performance and start to exploit the new CPU features. What we're now starting to see is that actually new hardware breaks some of the old versions of hypervisors. So, that kind of also shows you some of the jumps of innovation which are happening all over the place. So like these AMD Rome CPUs, they don't work on Xen 4.12 or older, because they've just done really clever stuff with the hardware which requires code which wasn't there before. And it won't even run unless the stuff is there.

The embedded segment is really interesting. So, the history there is, like, Xen ARM [9] was originally written for servers. And then we always had this strong

tie in with the sort of military space – like there are systems that use Xen-based virtualization for things like radar. And they started experimenting with embedded use cases initially on x86. Then when the ARM Quad came along, they started pushing us a little bit for secret little use cases into the ARM space as well. What has happened in the last three years is as virtualization extensions on ARM CPUs became relevant for some of these embedded use cases, we've seen interest from automotive and other market segments, and this came entirely as a surprise. I mean I was supporting this just as a cool marketable kind of project.

LM: When did you notice that change in gears, so to speak?

LK: Around the beginning of last year, there was a kind of step change. We had a number of fairly big companies like EPAM [Systems]; they are like a consultancy that [has] about 30,000 people working for them. They decided to speculatively – which consultancies don't normally do – build a reference platform for automotive that includes Xen at the bottom, and they have a number of different use cases where they can mix infotainment with cockpit control [10] and some other things. But then you have workloads of different criticality, [which] means you have to have a level of safety certification. But they were kind of just ignoring this at the time and just said, “we're just going to prototype that, and we're going to identify where the gaps are.” They've thrown quite a lot of money into this just to get to the stage to ... be able to show the demos. And they also started then taking this further. There's a whole kind of piece of this where you have a VM [virtual machine] where you can download container workloads, like, you know, basically microservices from a cloud provider.

LM: What practical purpose would this serve?

LK: The use case there would be, for example, if you have a taxi fleet, you would download a container, which kind of tells you about all your cars. Or, if you were interacting with a smart city, you could drive to the city [and] download something, and then your car can interact with it. The possibilities are endless. Or things like in the UK, in Europe, a lot of insurances offer to install a little black

box in a car, and it [the car] sends telemetric data about your driving and that would already be pre-integrated into the platform. They could have all the insurance companies, which they can just tie on, and then they can just channel the workload for this specific use case and enable it in a car, and it stays updated.

So it [has] really lots of interesting potential. And they [EPAM] were the leaders in this, and then ARM, at the end of last year, started to build an automotive product group. I think the conclusion is that there will be, like, significant investment.

At the same time, we set up a working group where we have safety expert companies on it – where we have companies like ARM, EPAM, and then we will have other companies on that as well – which are driving this entire safety story forward. It's kind of a really interesting, interesting project. And it's challenging. But we're also working with the Linux Foundation, the ELISA [Enabling Linux in Safety Applications] project, and Zephyr, which is trying to do something similar as well. We're all facing quite similar issues.

LM: They're all planning for safety certification as well?

LK: Yes. It's a really tough problem, because basically just the way how to safety software that was conceptually designed in the 80s and the way how we do software now is very different. But actually, the more I look into this, the more I think this is actually possible, and most of the issues aren't as huge [11]. Do you remember like 20 years ago when everybody said, you can't do open source in enterprise? And now everybody says you can't do open source for safety. But actually, if we can change some of the tooling – the tooling which is generally used in this field is so antiquated and you have all these different tools, and you have to store documentation in one tool, [and] you store another piece of documentation in another one. They are all kinds of server-based things, and the code is somewhere else. And then you have to have a manual change process, which uses another tool where you keep all your artifacts in sync. You just look at this and break it down and see that actually this can't be that hard with a bit of support and a bit of will.

You basically need to build up enough confidence now, which gets us to the same point as we were in the 90s with

Linux, where a couple of big vendors come out and support this, and then we're over the crucial barrier. We're close to this point now. I'm so excited, since it's such an interesting problem. I originally come from the embedded space. I don't know whether you're aware that in the 80s, Daimler had an EU-funded project called PROMETHEUS, and they were developing a self-driving car. The first variant was called Beta One, and I have worked a little bit on something called Beta Two. This was pre-GPS, so it was all image processing based. You could never productize that, because we had like a 500 CPU supercomputer in the boot of the car. But it was an interesting proof of concept, and some of the stuff which is being done now actually comes from some of these early ideas, which is why I'm kind of excited about that. ■■■

Info

- [1] The Xen Project: <https://xenproject.org>
- [2] Qubes OS: www.qubes-os.org
- [3] OpenXT: <https://openxt.org>
- [4] Understanding MDS and L1TF hardware vulnerabilities: <https://www.kernel.org/doc/html/latest/admin-guide/hw-vuln/mds.html>
- [5] The need for the secret-free Hypervisor: https://www.slideshare.net/xen_com_mgr/xpdds19-keynote-secretfree-hypervisor-now-and-future-wei-liu-software-engineer-citrix
- [6] Live updating Xen: https://www.slideshare.net/xen_com_mgr/xpdds19-liveupdating-xen-amit-shah-david-woodhouse-amazon
- [7] Xen 4.13 release notes: https://wiki.xenproject.org/wiki/Xen_Project_4.13_Release_Notes
- [8] Dario Faggioli at Open Source Summit Europe 2019: <https://osseu19.sched.com/event/TPcT/core-scheduling-for-virtualization-where-are-we-if-we-want-it-dario-faggioli-suse>
- [9] The Xen ARM initiative: https://wiki.xenproject.org/wiki/Xen_ARM_with_Virtualization_Extensions
- [10] EPAM's automotive initiative: <https://xenproject.org/2019/03/12/revolutionizing-the-auto-industry-with-open-source-epams-xen-powered-virtual-cockpit/>
- [11] Xen safety certification: https://www.slideshare.net/xen_com_mgr/ossjps19-the-road-to-safety-certification-how-the-xen-project-is-making-progress-within-the-auto-industry-and-beyond-lars-kurth-citrix-systems-uk-ltd

Shop the Shop

shop.linuxnewmedia.com

DISCOVER LibreOffice



Explore the **FREE** office suite used by busy professionals around the world!

Create your own:

- Word processing docs
- Spreadsheets
- Presentations
- Databases

Whether you work on a Windows PC, a Mac, or a Linux system, you have all you need to get started with LibreOffice today. This single-volume special edition will serve as your guide!

Order online:

shop.linuxnewmedia.com/specials

For Windows, macOS, and Linux users!



Reinventing Linux home directories with systemd-homed

New Kid

Systemd has already changed almost everything about the Linux startup process. Now an experimental new feature takes on the challenge of user home directories.

By Jack Wallen and Joe Casad

Systemd [1] was originally released on March 30, 2010 as a replacement for System V (SvsV) and BSD init. SysV init had been part of Linux for many years. The name System V, in fact, invokes memories of an early version of the Unix operating system that predated the Linux era.

The init system is the first service that starts after the system boot, and it is responsible for starting all the other services. The term “init” is short for “initialize,” and the role of the init system is to start everything that needs to be started when the OS springs to life.

SysV init was stable and predictable, but many developers believed it was past its prime. Perhaps the biggest issue with SysV init was that it was designed before the age of multiprocessing systems and could only do one thing at a time. Processes and services started in serial fashion, which significantly slowed down the boot process. Other developers and users wished for a system that was more consistent, with a better API and a more sophisticated means for expressing dependencies.

A number of alternatives to SysV init emerged and were the subject of lively debates on Linux blogs and news lists. SysV also had its supporters. Many Linux developers didn't see sufficient reason to change it, since it had been working well for so long. Others believed the complex frameworks offered as replacements were a violation of the core Unix design preference for simplicity and single-purpose tools.

A massive debate played out in the Linux space, and in the end, all the major Linux distros adopted systemd. A few holdouts remain (Devuan, for instance, is a fork of the Debian project that still used SysV), but now that we have reached 2020, it is clear that systemd is here to stay.

Systemd provides a uniform method for addressing the various system daemons and utilities, as well as device management, user login, network connections, and event logging. Prior to systemd, these tasks were handled by a collection of single-purpose tools. So when you needed to configure various pieces of the initialization process, you'd have to touch each one of those systems individually.

Lennart Poettering and Kay Sievers, both software engineers for Red Hat, developed systemd with the goal of creating a single system to control initialization that would express dependencies, allow more concurrent or parallel processes during boot time, and reduce computational overhead within the shell. But Poettering said from the beginning that systemd would be “never finished, never complete.” The systemd environment already includes such features as:

- Daemon initialization
- Snapshot support
- Process tracking
- Inhibitor locks

But Poettering wasn't content. Another important part of the system startup that was due for an upgrade was user home directories. With the new systemd-homed service that will appear in systemd 245 [2], the developers will address an aspect of the Linux startup process that many experts say should have been changed a long time ago.



within any systemd-homed-enabled Linux distribution.

Imagine being able to copy your home directory (and every associated configuration file) to a USB flash drive and have everything work as if it were still happily connected to your home computer – not just the user files, but the entire environment, including personal settings, preferences, and even authentication information. This new system could lead to completely self-contained users that can easily be migrated from system to system. In fact, you could plug in that flash drive and the user will (upon being granted permission by an admin user) automatically exist on the system.

How It Works

The `~/.identity` subfolder of the user's home directory contains a JSON-formatted user record that is used to confirm the user's identity. The details of the process depend on the storage mechanism. Systemd-homed supports a number of different mechanisms for storing and accessing the user home directory, including:

- Plain directory/Btrfs subvolume
- fscrypt home directories
- CIFS home directories
- LUKS home directories

According to the developers, the LUKS option is "...the most advanced and most secure storage mechanism." In the LUKS scenario, an encrypted LUKS2 volume is stored either on removable media or in a loopback file. A GPT partition table within the storage media or loopback file points to the enclosed LUKS volume. The label for the LUKS2 volume must be the same as the username. The LUKS volume contains an ext4, Btrfs, or XFS file-system with a single directory named for the user. This directory becomes the user home directory. The `~/.identity` subfolder within the home directory has a copy of the user record that matches the copy stored within the LUKS header.

The user's login password is also the password used to unlock the LUKS2 volume. When the user successfully logs in, systemd-homed uses the password provided with the login to unlock the LUKS volume. The user record stored within the header is then compared to the record stored in the `~/.identity` folder to ensure the data has not been tampered with. If the records match and are signed with a recognizable key, the home directory is then mounted and accessible to the user (Figure 1).

Managing Users

Like other systemd services, systemd-homed comes with its own command-line management utility. You will not be able to use the `useradd` utility or other classic command-line tools to manage systemd-homed users. Instead, use the `homectl` command to create, remove, or change a user directory. The `homectl` command supports several subcommands and options for setting up a user or changing user account settings.

For example, to create the user *bennie* for someone whose real name is Bennie Beanbag and to assign 400MB of disk space to the user directory, use:

No Place Like Home

The Linux home directory is a crucial part of the system directory hierarchy. Residing in the `/home` directory are user files, user data, and configuration settings. To date, many Linux users and admins have placed `/home` on a partition that is separate from the operating system so that, should something go wrong with the operating system, the data will still be safe. However, because of the way `/home` is handled within the operating system, system-to-system `/home` migration is never quite as easy as it should be.

In a conventional Linux environment, user information (such as username, ID, full name, home directory, and shell) are stored in `/etc/passwd`. Any user on the system can view that file. The password associated with users is stored in `/etc/shadow`. Unlike `/etc/passwd`, `/etc/shadow` cannot be viewed by just anyone. During login, the system authenticates the password with `/etc/shadow` and, upon successful authentication, reads `/etc/passwd` for the location of the user's home directory. For the simple act of logging in, three pieces are required for success, and two of those pieces reside with the system files – outside of the `/home` directory.

With systemd-homed, the systemd developers eliminate this dependency on external files, thus allowing home directories to become truly portable. The new approach places all information (username, group membership, password hashes) in a cryptographically signed, user-specific record within a single JSON file. This file is flexible, so it can handle just about any type of user information.

Portable Home

Eliminating the dependency on the system-based `/passwd` and `/etc/shadow` files means you will be able to take your home directory and move it to another drive (even a flash drive) and work with your data and configurations from

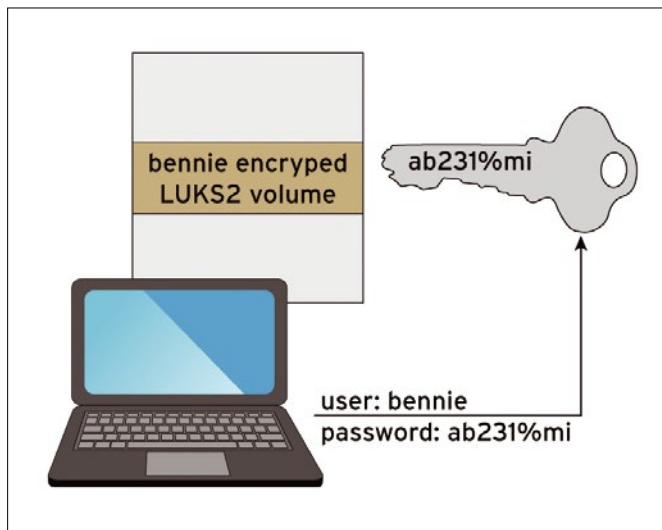


Figure 1: LUKS storage option: When a user logs in, the systemd-homed service looks for a LUKS2 volume with a label that matches the username. The service then attempts to unlock the encrypted volume using the login password. If the process completes successfully, the home directory is mounted and the user granted access.

```
homectl create bennie --real-name="Bennie Beanbag" \
--disk-size=400M
```

See the box entitled “More on homectl” for some of the basic homectl commands, or look for the homectl documentation to study the many options and command variants [3].

More on homectl

The homectl command has the following syntax:

```
homectl [OPTIONS...] {SUBCOMMAND} [NAME...]
```

The numerous options include format settings, user record properties, encryption settings, and more [3]. Some of the important subcommands are:

- `list` – list all home directories currently managed by the service
- `activate` – activate one or more home directories
- `create` – create a new home directory with the specified name
- `passwd` – change the password on the specified home directory and user account
- `resize` – change the setting for the amount of disk space assigned to the specified home directory
- `lock` – temporarily suspend access to the user’s home directory and remove any associated crypto keys from memory

The wide range of options and subcommands within a single command is similar to the format used with other systemd services. Unlike old-school command-line utilities, which tended to have a single, specific purpose, the broad and versatile systemd utilities encapsulate several different tasks within a single command structure, which leads to very tidy and hierarchical documentation. However, like systemd itself, it poses a challenge to those who prefer simple tools with a single purpose.

What’s It All For?

The systemd developers do not think of systemd-homed as a solution for all situations. First of all, this technique is not intended for system users (users with a UID < 1000). In general, systemd-homed is intended for end-user accounts. The ability to move a complete self-contained user home directory – not just user files – but the complete configuration and even login information, could be a major benefit in some environments. But even if you don’t plan on migrating your home directory, having a user directory system that is integrated with the rest of systemd will be a welcome development for many users and admins. Of course, the chorus of users who don’t like systemd in the first place are certainly not going to like it *more* because of systemd-homed.

Several potential users have already expressed concerns with how systemd-homed will handle SSH. If you’ve been paying close attention, you’ve probably already come to realize that, if systemd-homed leaves the user’s home directory encrypted until successful login authentication, how will users be able to log into a remote machine via SSH? Remember, the `.ssh` directory exists within the user’s home directory (in `~/ .ssh/`).

No universal solution to the SSH problem exists at this point, but, as Lennart Poettering recently pointed out on Reddit, systemd-homed is intended for laptops, workstations, and client systems “...machines you SSH from a lot more than SSH to...” [4]. The primary purpose is to support local login on end-user systems.

The biggest fear that comes with systemd-homed is that users and administrators will have to learn a new way of handling authentication and home directories. This is especially true for app developers, who might very well have to make serious changes to how their applications deal with these issues.

And although change is something most feared, others believe this change might well be a change for the better. Linux has been in need of improvement with how it handles user authentication and the home directory. For those who have been hoping for better home encryption, a centralized authentication system, and more portable home directories, systemd-homed is exactly what you’ve been looking for.

You can test systemd-homed in v245-rc1 from GitHub [5]. The installation of this system is not for the faint of heart, so it’s best to wait until the official release is available to kick the tires. The release of systemd 245 should happen sometime this year (2020). ■■■

Info

[1] systemd: <https://systemd.io/>

[2] systemd-homed: https://systemd.io/HOME_DIRECTORY/

[3] homectl: <https://www.freedesktop.org/software/systemd/man/homectl.html>

[4] Poettering regarding SSH: https://www.reddit.com/r/linux/comments/d6w03y/systemdhomed_reinventing_home_directories/

[5] systemd-homed at GitHub: <https://github.com/systemd/systemd/releases/tag/v245-rc1>



CLEAR OFF YOUR BOOKSHELF WITH DIGITAL ARCHIVES

Complete your collection of *Linux Magazine* and *ADMIN Network & Security* with our Digital Archive Bundles.

You get a full year of issues in PDF format to access at any time from any device.

2019
Archives
Available
Now!

bit.ly/2019-Digital-Archives



Managing and provisioning VMs

Touch and Go

With Vagrant, you can automate the creation and management of consistent virtual machines that work across platforms. *By Mayank Sharma*

Automation is the key ingredient for efficiency in any system administration strategy. If your job involves spinning virtual machines (VMs) regularly, you should familiarize yourself with Vagrant [1], which helps you make consistent virtual environments available to your users with a few keystrokes. Vagrant provides a simple and easy to use command-line interface (CLI) that helps automate the creation, editing, running, and deletion of VMs. Moreover, it supports all major virtual platforms, such as VirtualBox and VMware, and plays nicely with all the well-known software configuration tools, such as Chef, Puppet, Ansible, Fabric, and more.

Since VM hypervisors like VirtualBox and VMware have their own CLIs that can be used to automate provisioning, why should you choose Vagrant? Vagrant offers consistency and interoperability. With Vagrant you can define your virtual environment in code and then use it to provision VMs on top of any hypervisor running on any operating system. Additionally, anytime you make changes to the virtual environment, instead of sharing the complete VM that can be worth several gigabytes, you can share a simple

text file that can then be used to provision VMs with the changes.

A Rolling Start

While Vagrant can provision VMs with various virtualization software, by default it uses the free and open source VirtualBox.

To get started, the first thing you need to do is install Vagrant and VirtualBox. The Vagrant project recommends downloading the packages for Vagrant [2] and VirtualBox [3] directly from their respective websites, since package managers will probably have outdated versions.

Guest Additions Plugin

The VirtualBox guest additions is an essential component that is required to take full advantage of Vagrant. However, rolling them into every box and then making sure the boxes are running the latest version of the guest additions is a time consuming task and an unnecessary distraction. A Vagrant plugin can take care of installing and updating the guest additions automatically.

To install the plugin, head to the terminal and run the following Vagrant command:

```
$ vagrant plugin install vagrant-vbguest
```

Once Vagrant has been equipped with the plugin, every time you launch a box, Vagrant will check whether the VM is equipped with the guest additions. If the latest version is installed, it'll continue booting the machine. If an update or installation is required, Vagrant will then automatically download all dependent packages and then install the guest additions from the VirtualBox ISO (Figure 2).

Next, you'll need to create a Vagrant configuration file that defines all the VM's characteristics from a template. You can search for templates based on various operating systems and pre-defined purposes on the project's website [4]. For example, fire up a terminal and enter:

```
$ mkdir ~/demo
$ cd ~/demo
$ vagrant init centos/7
```

to create a VM based on the CentOS 7.6.181 release inside the ~/demo directory.

Lead Image © adchariya champipat, 123RF.com

```

bodhi@galaxy: ~/demo
File Edit View Search Terminal Help
bodhi@galaxy:~/demo$ vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Box 'centos/7' could not be found. Attempting to find and install...
    default: Box Provider: virtualbox
    default: Box Version: >= 0
==> default: Loading metadata for box 'centos/7'
    default: URL: https://vagrantcloud.com/centos/7
==> default: Adding box 'centos/7' (v1905.1) for provider: virtualbox
    default: Downloading: https://vagrantcloud.com/centos/boxes/7/versions/1905.1/providers/virtualbox.box
    default: Download redirected to host: cloud.centos.org
    default: Progress: 0% (Rate: 56508/s, Estimated time remaining: 2:19:56)
    default: Progress: 0% (Rate: 147k/s, Estimated time remaining: 1:17:41)
    default: Progress: 0% (Rate: 109k/s, Estimated time remaining: 1:18:35)
==> default: Successfully added box 'centos/7' (v1905.1) for 'virtualbox'!
==> default: Importing base box 'centos/7'...
==> default: Matching MAC address for NAT networking...
==> default: Checking if box 'centos/7' version '1905.1' is up to date...
==> default: Setting the name of the VM: demo_default_1582727289208_19762
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
==> default: Forwarding ports...
    default: 22 (guest) => 2222 (host) (adapter 1)
==> default: Booting VM...

```

Figure 1: The first time you bring up a VM, it'll take a lot longer, because it downloads the image from the Internet.

Under the ~/demo directory, this command creates a file called Vagrantfile, which is the main configuration file that defines all the VM's attributes. If you want to make changes to a VM, you'll need to edit this file. You'll need to be well-versed with the Vagrantfile's anatomy in order to modify or create one as per your needs.

For now, you can bring up the VM using the default settings, by typing:

```
$ vagrant up
```

This command will create the actual VM reading the configuration specified in the Vagrantfile. This tells Vagrant to create a new VirtualBox machine based on the base image specified in the Vagrantfile. It'll do so by copying the virtual hard disk files from the remote server (Figure 1).

Once it's done, you'll have a fully featured CentOS 7 VM running headless in the background. If you get errors regarding missing guest additions, refer to the "Guest Additions Plugin" box to leverage Vagrant's extensive plugin infrastructure to solve the issue permanently.

Once the VM is up and running, no extra window will pop up on the screen; you'll be returned to the shell prompt, because Vagrant VMs run headless by default. Instead you can access the VM by typing:

```
$ vagrant ssh
[vagrant@localhost ~]$
```

As you can see, the command uses SSH to connect to the VM. The command will automatically authenticate the SSH sessions and drop you at the SSH shell in the machine. You can now interact with the VM like any other CentOS installation (Figure 3). You can also share files between the host and the VM (see the "Shared Filesystem" box). When you're done working inside the VM, type:

```
[vagrant@localhost ~]$ exit
logout
Connection to 127.0.0.1 closed.
$
```

which will drop you back to your host's CLI.

Vagrantfile Anatomy

Before you can make changes to your VMs, you must familiarize yourself with Vagrantfiles. Vagrant is configured separately for each VM, each of which has its own isolated configuration environment. At the crux of each VM is the configuration file, Vagrantfile.

The Vagrantfile is a simple text file that Vagrant reads in order to create the virtual environment. The Vagrantfile describes the various parameters that are necessary to create the VM. Vagrant reads this file to configure and provision the VMs.

Because of this singular Vagrantfile, other users can use Vagrant to automatically and easily create their virtual environment with a single command. Vagrantfiles are portable, which means you can use them to create and provision VMs on every platform that Vagrant supports, including Linux, Windows, and Mac OS X.

As shown previously, the Vagrantfile is created automatically when you

```

bodhi@galaxy: ~/demo
File Edit View Search Terminal Help
bodhi@galaxy:~/demo$ vagrant plugin install vagrant-vbguest
Installing the 'vagrant-vbguest' plugin. This can take a few minutes...
Fetching: micromachine-3.0.0.gem (100%)
Fetching: vagrant-vbguest-0.23.0.gem (100%)
Installed the plugin 'vagrant-vbguest (0.23.0)'!
bodhi@galaxy:~/demo$ vagrant up
Bringing machine 'default' up with 'virtualbox' provider...
==> default: Checking if box 'centos/7' version '1905.1' is up to date...
==> default: Clearing any previously set forwarded ports...
==> default: Clearing any previously set network interfaces...
==> default: Preparing network interfaces based on configuration...
    default: Adapter 1: nat
==> default: Forwarding ports...
    default: 22 (guest) => 2222 (host) (adapter 1)
==> default: Booting VM...
==> default: Waiting for machine to boot. This may take a few minutes...
    default: SSH address: 127.0.0.1:2222
    default: SSH username: vagrant
    default: SSH auth method: private key
==> default: Machine booted and ready!
[default] No Virtualbox Guest Additions installation found.
Loaded plugins: fastestmirror
Loading mirror speeds from cached hostfile
 * base: mirror.vanehost.com
 * extras: mirror.vanehost.com
 * updates: mirror.vanehost.com
Resolving Dependencies
--> Running transaction check

```

Figure 2: The vbguest plugin will automatically download the latest guest additions if it isn't available in the VM.

```
vagrant@localhost:~
File Edit View Search Terminal Help
bodhi@galaxy:~/demo$ cat /proc/version
Linux version 4.18.0-25-generic (buildd@lcy01-amd64-025) (gcc version 8.3.0 (Ubuntu 8.3.0-6ubuntu1~18.10.1)) #26-Ubuntu SMP Mon Jun 24 09:32:08 UTC 2019
bodhi@galaxy:~/demo$ vagrant ssh
Last login: Sat Feb 29 19:36:17 2020 from 10.0.2.2
[vagrant@localhost ~]$ ls -l /vagrant
total 8
-rw-r--r--. 1 vagrant vagrant 184 Feb 29 17:00 provision.sh
-rw-r--r--. 1 vagrant vagrant 3178 Feb 29 16:05 Vagrantfile
[vagrant@localhost ~]$ cat /proc/version
Linux version 3.10.0-957.12.2.el7.x86_64 (mockbuild@kbuilder.bsys.centos.org) (gcc version 4.8.5 20150623 (Red Hat 4.8.5-36) (GCC) ) #1 SMP Tue May 14 21:24:32 UTC 2019
[vagrant@localhost ~]$
```

Figure 3: Vagrant doesn't support SSH using a normal password; instead it uses public key authentication.

issue the `vagrant init` command, for example:

```
vagrant init centos/7
```

By default, the file is heavily commented out and only exposes a handful of parameters. However, these are the most essential ones, and you can use this Vagrantfile to create a VM with any additional modifications.

Also note that a Vagrantfile is written in Ruby. Even if you aren't well versed in Ruby, you'll be able to configure every aspect of Vagrant without ever learning the programming language.

Shared Filesystem

With Vagrant, you can easily share files between the host and the VM. The files placed inside the shared filesystem are kept in sync between the physical and the virtual machines. You can then easily roll the shared filesystem into your host's backup strategy. In addition, these files will always be available to the VM whenever you create a new VM instance.

By default, the current project folder (the root folder that contains the Vagrantfile) is shared between the host and guest machines. You can verify this by first SSHing into the VM and then listing the files in the directory:

```
$ vagrant ssh
[vagrant@localhost ~]$ ls /vagrant
Vagrantfile
```

As you can see, this is the same Vagrantfile that's kept under the `~/demo` directory on the host. Any file or directory you place inside this folder will be available in `/vagrant` inside the guest.

You can easily share another folder from the host to the VM by tweaking the Vagrantfile. Open the file and add the following line:

```
config.vm.synced_folder "
"/home/bodhi/Documents/", "/docroot"
```

This line asks Vagrant to share the contents of the `/home/bodhi/Documents/` folder on the host inside the `/docroot` directory in the VM. After saving the file, simply restart the VM with:

```
$ vagrant reload
```

During boot up, Vagrant will automatically create the `/docroot` directory inside the VM and mirror the contents of `/home/bodhi/Documents/` inside it. You can again verify this by SSHing into the VM and then listing the directory's contents:

```
$ vagrant ssh
[vagrant@localhost ~]$ ls /docroot
```

you issue `vagrant up` for the first time. Vagrant saves this box file for future usage, so it won't have to be downloaded again, even when you remove a Vagrant VM.

The required box needs to be specified for every VM using the `config.vm.box` parameter in the Vagrantfile. In my example, this is set to `centos/7` since this was the box I requested with `vagrant init` earlier. Remember that multiple Vagrant environments can have the same `config.vm.box` value, which gives you the flexibility to create different VMs for different purposes with the same box image.

Port Forwarding

An important setting for my particular use of VMs is port forwarding, which allows me to access services running inside the VM from the host. Vagrant supports this type of networking as well. I can tweak the Vagrantfile to set up a port on the host machine to forward to a port in the VM. In effect, this allows me to access the services running inside the VM.

For instance, if I forward the standard port for HTTP content (port 80) in the VM to port 8081 on the host, then I can access the web server running in the VM by pointing the web browser on the host machine to `http://localhost:8081`. In the background, the traffic sent to port 8081 is actually forwarded to port 80 on the VM.

Fire up the Vagrantfile in a text editor and enter the following parameter:

Listing 1: Provisioning Shell Script

```
01 $ sudo nano provision.sh
02
03 # !/usr/bin/env bash
04
05 echo "Now installing the Apache web server quietly in the background"
06 yum update httpd > /dev/null 2>&1
07 yum install -y httpd > /dev/null 2>&1
08 systemctl start httpd
```

```
config.vm.network "forwarded_port", 2
guest: 80, host: 8081
```

With this parameter, I am instructing Vagrant to forward port 80 from inside the VM to port 8081 on the host. Save the file and restart the VM:

```
$ vagrant reload
```

To check the settings, you can use the `SimpleHTTPServer` module in Python to read the contents of the `/vagrant` directory. Once the VM is up and running, SSH into it and bring the web server online with:

```
$ vagrant ssh
[vagrant@localhost ~]$ cd /vagrant
[vagrant@localhost ~]$ sudo python -m 2
SimpleHTTPServer 80
```

Now open a browser on the host and point it to `http://localhost:8081`, and you'll get a `/vagrant` directory listing served from inside the VM. You can now tweak the example to install any service inside the VM and modify the Vagrantfile to shuttle traffic between the host and the VM.

Provisioning VMs

At the moment, the CentOS VM is pretty basic. To harness Vagrant's power, you'll have to flesh out the installation. You can do this in a couple of ways. You can use the CentOS package manager to add packages manually. However, the better option is to automatically install the software as part of the VM's creation process, which is known as provisioning. Vagrant supports provisioning with shell scripts, Chef, or Puppet, and you can add more provisioners via plugins.

I personally prefer provisioning via shell scripts as opposed to Chef or Puppet, which are overkill for my sim-

ple requirements. Working with shell scripts is fairly straightforward. All you need to do is simply gather all the required operations that need to be handled automatically inside a script (see Listing 1).

Line 3 of Listing 1 specifies which shell to use to execute the rest of the file (bash in this case). The script will then display that it's about to install the Apache web server without any further prompts or outputs, so as to not fill the screen with unnecessary output.

Vagrant will run the script as root, which is why there is no need to actually use `sudo` on lines 6, 7, and 8. The `-y` flag tells `yum` to automatically respond "yes" to any prompts. This is important since I am provisioning the VMs automatically. Because there is no human interaction, if `yum` were to ask for confirmation, the script would simply never finish. As previously mentioned, `yum`'s output is sent to `/dev/null` instead of the terminal.

Once the shell script has been created, the next step is to configure Va-

grant to use it to provision the VMs during boot. For this, you can point to the script from the Vagrantfile with the following parameter:

```
config.vm.provision "shell", path: 2
"provision.sh"
```

This tells Vagrant to provision the machine with the shell provisioner and to use the shell script named `provision.sh` that's available in the project directory (`~/demo` on the host).

Now bring up or reload the VM. Thanks to this parameter, Vagrant will execute the `provision.sh` script after the VM is up and running (Figure 4).

GUI Guest

While Vagrant is primarily used to create and provision headless VMs, you can also create a VM with a full blown graphical desktop. Open the Vagrantfile, scroll down, and uncomment the following lines or add them manually:

```
config.vm.provider "virtualbox" do |vb|
  vb.gui = true
  vb.memory = "1024"
end
```

These lines tell Vagrant to enable the VirtualBox GUI mode and allocate 1024MB of RAM to it. Once you've saved the changes, you can restart the VM. This time, it'll pop out a VirtualBox window and drop you at the shell login

Figure 4: Once a VM has been provisioned, the script will be ignored on subsequent reboots.

prompt. Even though Vagrant uses key-based authentication by default, it is a general convention to set the password for the “vagrant” user to *vagrant*. You can use these login credentials to log into the VM manually.

Since the VM I specified earlier is a CentOS server distribution, you’ll have to pull in packages for a full-blown CentOS graphical desktop, which is fairly well-documented [5], after logging into this VM.

Cleanup

Once you’ve worked with your Vagrant VMs, the last order of business is to clean up the environment. Depending on your needs, you can use Vagrant to take snapshots, suspend, halt, or destroy the VMs.

If you need to verify your VM’s current state before running any commands, you can get some useful output with:

```
$ vagrant status
```

Suspending the guest machine will save the machine’s current running state and stop it:

```
$ vagrant suspend
```

A suspended machine can be resumed with the `vagrant up` command.

Halting the guest machine will shut it down pretty much like a physical computer. To turn off the machine, type:

```
$ vagrant halt
```

When you enter this command, Vagrant will first attempt to gracefully halt the machine by executing the proper commands to initiate a shutdown from within the guest machine. However, if it is unable to communicate with the machine and the shutdown sequence times out, Vagrant will forcefully shut it down.

It’s also wise to take regular snapshots of the VMs, which you can then roll back to if you run into problems. To do this, use:

```
$ vagrant snapshot save <name>
$ vagrant snapshot restore <name>
```

Here `<name>` is the unique string to identify the snapshot. The first com-

mand creates the snapshot, while the second command restores from it (Figure 5).

Finally, when you are done with a VM you can zap it from your host machine and remove all traces of it by deleting hard disks, state files, and so on with:

```
$ vagrant destroy
```

Remember that destroying a VM will cause you to lose all changes, as well as any files or folders created outside of the shared filesystem. When you now issue a `vagrant up` command, Vagrant will create the VM from scratch, which means it’ll provision it again as well.

Conclusions

Now that you have a basic understanding of Vagrant, I hope you can see its potential. If you choose to deploy Vagrant in your production environment, you should first read Vagrant’s extensive documentation section [6]. ■■■

Info

- [1] Vagrant: www.vagrantup.com
- [2] Download Vagrant: www.vagrantup.com/downloads.html
- [3] VirtualBox Linux downloads: www.virtualbox.org/wiki/Linux_Downloads
- [4] List of predefined boxes: <https://app.vagrantup.com/boxes/search>
- [5] Installing desktop environments inside CentOS 7: www.techrepublic.com/article/how-to-install-a-gui-on-top-of-centos-7
- [6] Vagrant documentation: www.vagrantup.com/docs

Author

Mayank Sharma has been writing and reporting on open source software from all over the globe for almost two decades.



```
bodhi@galaxy: ~/demo
File Edit View Search Terminal Help
bodhi@galaxy:~/demo$ vagrant status
Current machine states:

default                running (virtualbox)

The VM is running. To stop this VM, you can run `vagrant halt` to
shut it down forcefully, or you can run `vagrant suspend` to simply
suspend the virtual machine. In either case, to restart it again,
simply run `vagrant up`.
bodhi@galaxy:~/demo$ vagrant snapshot save centos-mariadb-installed
==> default: Snapshotting the machine as 'centos-mariadb-installed'...
==> default: Snapshot saved! You can restore the snapshot at any time by
==> default: using `vagrant snapshot restore`. You can delete it using
==> default: `vagrant snapshot delete`.
bodhi@galaxy:~/demo$ vagrant snapshot list
==> default:
centos-mariadb-installed
centos-provisioned
centos-webserver-installed
bodhi@galaxy:~/demo$ vagrant snapshot delete centos-provisioned
==> default: Deleting the snapshot 'centos-provisioned'...
==> default: Snapshot deleted!
bodhi@galaxy:~/demo$
```

Figure 5: It’s a good practice to take a snapshot of the VM after making any significant change.



Linux Magazine is your guide to the world of Linux. Look inside for advanced technical information you won't find anywhere else!

Expand your Linux skills with:

- In-depth articles on trending topics, including Bitcoin, ransomware, cloud computing, and more!
- How-tos and tutorials on useful tools that will save you time and protect your data
- Troubleshooting and optimization tips
- Insightful news on crucial developments in the world of open source
- Cool projects for Raspberry Pi, Arduino, and other maker-board systems

If you want to go farther and do more with Linux, subscribe today and never miss another issue!

Subscribe now!
shop.linuxnewmedia.com/subs

GET IT NOW!
FAST DELIVERY WITH OUR PDF EDITION

NO ADS

LINUXVOICE

System monitoring with Glances

Watchdog

Admins and power users like to watch the load on their computers. Glances lets you see immediately if something is wrong.

By Erik Bärwaldt



Linux has no lack of monitoring tools. Because Linux is used so frequently in a server context, monitoring applications have always been important.

For the Linux desktop, you will find many smaller applications that focus on specific areas of monitoring. Glances [1] is a program that keeps you up to date with the current health state of your system.

Glances is available from the software repositories of most popular distributions and derivatives. For Debian, Ubuntu, or Linux Mint, use

```
sudo apt install glances
```

to install Glances. The project site [2] lists other ways to obtain Glances, which was written in Python, including Python's own package manager pip, a Glances Docker container, and the option to compile the application from source code.

Modes

Glances supports several operating modes for which you call the `ncurses` application with different parameters. In the simplest case, simply type the `glances` command at the prompt. The tool then reads the values from hardware and software, displays

them in the terminal, and continuously updates the process list (Figure 1).

In order to keep an eye on the server on the home network from anywhere on the LAN, call up Glances on this LAN with the `glances -w` command. The `-w` parameter activates browser mode, which does not require its own web server, but is based on the Bottle web framework.

You can now access the program on `http://<Server-IP>:61208` in the web browser from any other workstation on the web (Figure 2) and locally via a URL of `http://localhost:61208`. If you want to monitor the server via SSH from a workstation on the LAN, access the software via the terminal without browser mode.

In addition, Glances provides a client-server mode, which also allows the server to be called from any workstation on the network. To do this, launch the software on the server with the `glances -s` command and on the client by entering:

```
glances -c <Server-IP>
```

By default, Glances listens on port 61209 in server mode. If required, you can also define another port with the `-p` parameter.

To prevent unauthorized persons from accessing the Glances server and thus

gaining an overview of the available hardware capacities, there is also the option of setting a password on the server with the `--password` parameter, which the server then prompts you for when you access its IP.

Displays

Glances provides far more information than the usual system monitoring programs. In the upper part of the display, you will find information about the distribution, the kernel used, and the IP address of the system. Below this, you can see information about the CPU, the RAM, and the swap partition, as well as their current load values, listed in a table with several columns.

On the far right, you will also see information about the CPU kernel load and the operating time of the computer. Below that Glances provides periodically updated information about the network interfaces available on the system, the mass storage partitions, the occupancy of the system partition, and the installed sensors.

Shortcuts

Glances supplements these data with corresponding throughput rates, as well as with temperature values for sensors.

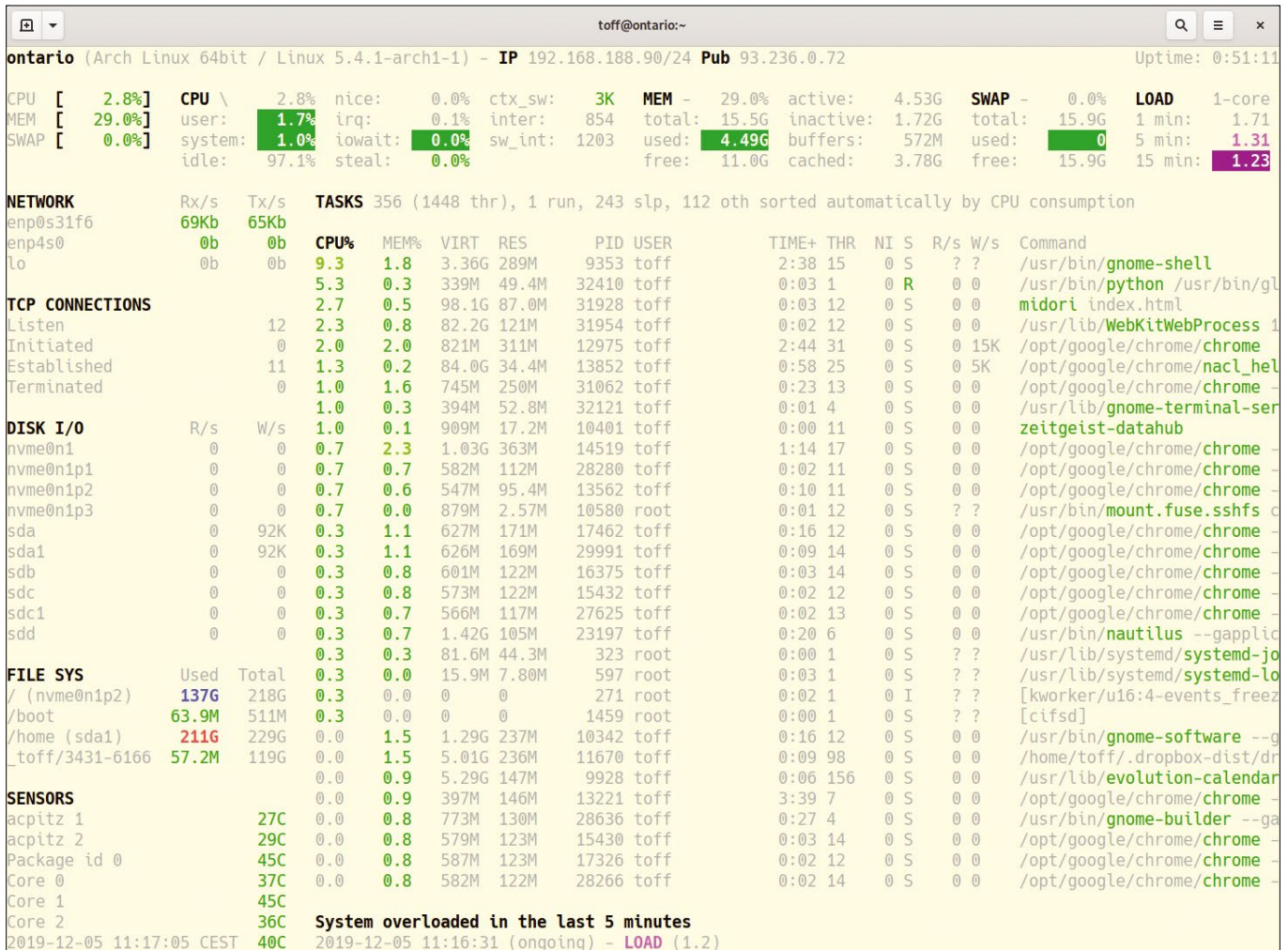


Figure 1: Glances continuously provides a wealth of information about the running system in the terminal.

To the right of this is a process display, which not only lists the individual processes one below the other, but also the CPU and memory usage per process in a separate column to the left. The individual displays are shown graphically, much like the load displays of htop or top.

Glances offers various possibilities to customize the appearance. For example, you can switch certain load displays on or off at the touch of a button. For example, pressing *N* removes the network interface information from the Glances window, while at the same

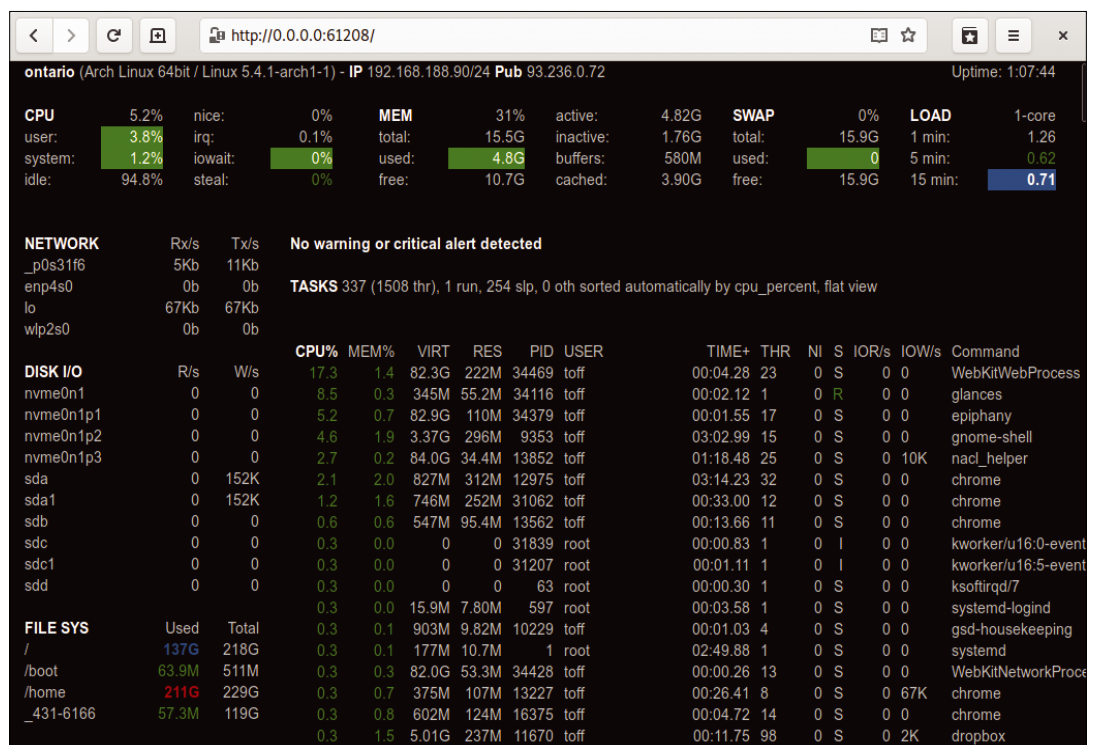


Figure 2: The integrated web server lets you retrieve data in a web browser from somewhere on the network.

```

1 #####
2 # Globals Glances parameters
3 #####
4
5 [global]
6 # Does Glances should check if a newer version is available on PyPI ?
7 check_update=true
8 # History size (maximum number of values)
9 # Default is 28800: 1 day with 1 point every 3 seconds
10 history_size=28800
11
12 #####
13 # User interface
14 #####
15
16 [outputs]
17 # Theme name for the Curses interface: black or white
18 curse_theme=black
19 # Limit the number of processes to display in the WebUI
20 max_processes_display=30
21
22 #####
23 # plugins
24 #####
25
26 [quicklook]
27 # Set to true to disable a plugin
28 # Note: you can also disable it from the command line (see --disable-plugin <pl
29 disable=False

```

Figure 3: If needed, you can configure even the details of Glances in the `glances.conf` configuration file.

time moving up the mass storage and I/O data. The developers provide detailed documentation on the keys that can be used interactively in the program [3].

Colors

In general, Glances highlights the utilization values of the hardware with a colored background, where this makes sense. In this way, you as an administrator can see immediately where problems occur. The software uses four colors: Values highlighted in green mean that the utilization of the component concerned is in the low to average range. Blue indicates an already above average load.

If values appear in violet, you should take a closer look at the component in question. Finally, red indicates a clear problem. In this case, software warnings also appear at the bottom of the window. If necessary, you can modify the assignment of the respective load

values to the colors in an individual configuration.

Configuration

The Glances configuration file is available in the `/etc/glances/` or `~/.config/glances/` folder. Some distributions, such as Arch Linux, don't even create the file, but simply save a template in `/usr/share/doc/glances/`.

`glances.conf` is plain text only; you can edit it with any text editor (Figure 3). The configuration file, which is extensive but quite clear-cut, summarizes the individual setting options in groups. Thanks to meaningful naming of the options, even inexperienced users can easily adapt the file to their own wishes.

You can use the `*_careful`, `*_warning`, and `*_critical` variables here to define thresholds for the hardware load's color display. This means that Glances will

warn you of even a relatively low load on critical systems, if so desired.

Data Export

Glances lets you export the acquired data for further processing in various formats, including JSON and CSV. To do this, launch the software with the `--export` parameter and specify the file name and path for the target file.

In some cases, data from Glances can also be transferred directly into databases. To do this, edit the configuration file accordingly to establish a connection to a database server. Detailed examples can be found in the documentation [4].

Conclusions

Glances is ideal for monitoring servers on smaller networks, but also cuts a fine figure on the desktop when it comes to finding system problems. You get a comprehensive list of the individual system states without having to enter many parameters manually or even start different programs.

The software also impresses with its simple configuration options using a text file, which also allows special hardware to be integrated. Also for administrators, who have to document the state of systems, Glances offers valuable help due to its export options to different file and database formats. ■■■

Info

- [1] Glances: <https://nicolargo.github.io/glances/>
- [2] Installation options: <https://glances.readthedocs.io/en/latest/install.html>
- [3] Documentation on interactive keys and parameters: <https://glances.readthedocs.io/en/latest/cmds.html>
- [4] Data export guide: <https://glances.readthedocs.io/en/latest/gw/index.html>

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

Security automation with rkhunter

On the Hunt

The Rootkit Hunter script efficiently checks for malware, with the potential to detect over 240 rootkits. *By Bruce Byfield*

Rootkit Hunter, or rkhunter, detects over 240 rootkits – pieces of malware designed to gain control of a system. However, while testing for rootkits may be rkhunter's main purpose, it is far from the only one. You can see the list of the names of the various tests run by the script by entering `rkhunter --list` (Figure 1) [1]. Mostly, the tests' names are self-explanatory. They include checks not only for rootkits, but also changed or deleted libraries and commands, hidden ports, loaded kernel modules, and several dozen other aspects of a system besides.

Rkhunter is written for generic Unix systems with a Bourne-type shell, such as Bash or ksh. Since its tests depend on online databases, it also requires an Internet connection. It is available in major Linux distributions and can be run from the command line, or as a cron job. Note that some distributions, such as Debian and its derivatives, may not install some of the Perl packages needed for a few of the tests. You can see what functionality may be missing by running `rkhunter --list` (Figure 2) and will then have to figure out which packages support the missing function-

ality. These packages, of course, may have different names depending on your distribution.

Setup and Configuration

If you install rkhunter from outside your distro's standard repositories, you can make sure that you always have the latest version by running `rkhunter --versioncheck` to help ensure your system's security. With most commands, I would always recommend that you not run the repository version, but rkhunter is so slow to release that in many cases the latest version is contained in a dis-

```

root@nanday:/etc# rkhunter --list

Current test names:
additional_rkts all apps attributes avail_modules deleted_files
filesystem group_accounts group_changes hashes hidden_ports hidden_procs
immutable ipc_shared_mem known_rkts loaded_modules local_host login_backdoors
malware network none os_specific packet_cap_apps passwd_changes
ports possible_rkt_files possible_rkt_strings promisc properties rootkits
running_procs scripts shared_libs shared_libs_path sniffer_logs startup_files
startup_malware strings susp_dirs suspscan system_commands system_configs
system_configs_ssh system_configs_syslog tripwire trojans

Grouped test names:
additional_rkts => possible_rkt_files possible_rkt_strings
group_accounts => group_changes passwd_changes
local_host => filesystem group_changes passwd_changes startup_malware system_
configs_ssh system_configs_syslog
malware => deleted_files hidden_procs ipc_shared_mem login_backdoors runni

```

Figure 1: The `--list` option shows the tests that rkhunter runs.


```
Perl module installation status:
perl command           Installed
File::stat             Installed
Getopt::Long           Installed
Crypt::RIPEMD160      MISSING
Digest::MD5            Installed
Digest::SHA            Installed
Digest::SHA1          MISSING
Digest::SHA256        MISSING
Digest::SHA::PurePerl MISSING
Digest::Whirlpool     MISSING
LWP                    Installed
URI                    Installed
HTTP::Status           Installed
HTTP::Date             Installed
Socket                 Installed
Carp                   Installed
```

Figure 2: Some Perl-based scripts may need to be installed separately.

tor's repository (see below). Currently, for example, even Debian, whose software versions often lag behind those of other distributions, has the latest rkhunter release in its official release.

No matter what your installation source, before running rkhunter for the

ing of the databases by adding:

```
APT_AUTOGEN="yes"
```

to

```
/etc/default/rkhunter
```

first time, you need to run `rkhunter --propupd` to ensure that the command's databases are up to date (Figure 3). You should also run `--propupd` whenever the system is updated. Otherwise, the log will contain false positives that will only waste your time. You can automate the updat-

You may also want to configure `/etc/rkhunter.conf` for your system. The field `MAIL-ON-WARNING="EMAIL"` can be modified to send a list of warnings to the address of your choice. You may also want to whitelist some common false positives by removing the `#` to uncomment these lines:

```
#ALLOWHIDDENIR=/dev/.udev
```

```
#ALLOWHIDDENIR=/dev/.static
```

```
#ALLOWHIDDENIR=/dev/.initramfs
```

After the first time you run rkhunter, additional whitelists (Figure 4) can be defined by adding the field `SCRIPTWHITELIST="FILE,FILE"` so that false positives are not flagged when the command is run. After any changes to `rkhunter.conf`, in some distros you can run `rkhunter -C` to check for any errors.

Running the Tests

If you install rkhunter from a distribution's repository, it can be run as soon as it is installed, although whitelisted files will be logged. If you install from

```
root@nanday:/etc# rkhunter --propupd
[ Rootkit Hunter version 1.4.6 ]
File updated: searched for 181 files, found 148
```

Figure 3: Use `--propupd` to keep rkhunter's databases current.

```
[12:30:10] Info: Found file '/bin/egrep': it is whitelisted for the 'script replacement' check.
[12:30:11] /bin/fgrep [ OK ]
[12:30:11] Info: Found file '/bin/fgrep': it is whitelisted for the 'script replacement' check.
```

Figure 4: Rkhunter flags known exceptions or whitelisted files that give false positives.

```
root@nanday:~# rkhunter --check
[ Rootkit Hunter version 1.4.6 ]

Checking system commands...

Performing 'strings' command checks
  Checking 'strings' command [ OK ]

Performing 'shared libraries' checks
  Checking for preloading variables [ None found ]
  Checking for preloaded libraries [ None found ]
  Checking LD_LIBRARY_PATH variable [ Not found ]

Performing file properties checks
  Checking for prerequisites [ OK ]
  /usr/sbin/adduser [ OK ]
  /usr/sbin/chroot [ OK ]
  /usr/sbin/cron [ OK ]
  /usr/sbin/groupadd [ OK ]
  /usr/sbin/groupdel [ OK ]
  /usr/sbin/groupmod [ OK ]
```

Figure 5: Rkhunter at the start of its run.

an outside source, however, configure the command as described above. In either case, to run the command, enter `rkhunter --check (-c)` (Figure 5). Rkhunter will begin to run its tests, although at several stages it will pause until you press the Enter key. As it runs, it may flag warnings, ranging from whitelists (files that list acceptable

files) to unusually large files. A running summary of results displays (Figure 6) as tests are done, although they may scroll too quickly to be easily read at some stages. Not to worry – you will want to study `/var/log/rkhunter` anyway (Figure 7). Some of the warnings may be false positives; for example, Firefox often has a larger than usual

configuration file (Figure 8). Take your time with the logfile, and make a list of any problems that you need to research to learn how to address.

This basic sequence can be modified with options, some of which have a long and a short form, and some which have only a long form. To start with, you can use `--disable` or `--enable` to select the tests to run in comma-separated lists (use the `--list` option to see which tests are available). Since the next step after running rkhunter is to examine the log, you can also use `--display-logfile` to show the log immediately after the completion of the command. Similarly, `--skip-keypress (-sk)` omits the pauses in the running of the command where you need to press the Enter key to continue. In addition, you can also suppress default features with commands like `--nocolors` and `--nolog` or set the directories to use with options like `configfile FILE` or `tmpdir FILE`.

```
System checks summary
=====

File properties checks...
  Files checked: 148
  Suspect files: 1

Rootkit checks...
  Rootkits checked : 479
  Possible rootkits: 4

Applications checks...
  All checks skipped

The system checks took: 6 minutes and 0 seconds

All results have been written to the log file: /var/log/rkhunter.log

One or more warnings have been found while checking the system.
Please check the log file (/var/log/rkhunter.log)
```

Figure 6: Rkhunter finishes with a summary of results.

```
[12:29:37] Running Rootkit Hunter version 1.4.6 on nanday
[12:29:37]
[12:29:37] Info: Start date is Tue Mar 31 12:29:37 PDT 2020
[12:29:37]
[12:29:37] Checking configuration file and command-line options...
[12:29:37] Info: Detected operating system is 'Linux'
[12:29:37] Info: Found O/S name: Debian GNU/Linux 10 (buster)
[12:29:37] Info: Command line is /usr/bin/rkhunter --check
[12:29:37] Info: Environment shell is /bin/bash; rkhunter is using dash
[12:29:37] Info: Using configuration file '/etc/rkhunter.conf'
[12:29:37] Info: Installation directory is '/usr'
[12:29:37] Info: Using language 'en'
[12:29:37] Info: Using '/var/lib/rkhunter/db' as the database directory
[12:29:37] Info: Using '/usr/share/rkhunter/scripts' as the support script directory
[12:29:37] Info: Using '/usr/local/sbin /usr/local/bin /usr/sbin /usr/bin /sbin /bin /usr/libexec' as the command directories
[12:29:37] Info: Using '/var/lib/rkhunter/tmp' as the temporary directory
[12:29:37] Info: No mail-on-warning address configured
[12:29:37] Info: X will be automatically detected
[12:29:37] Info: Found the 'basename' command: /usr/bin/basename
[12:29:37] Info: Found the 'diff' command: /usr/bin/diff
[12:29:37] Info: Found the 'dirname' command: /usr/bin/dirname
[12:29:37] Info: Found the 'file' command: /usr/bin/file
/var/log/rkhunter.log
```

Figure 7: After rkhunter runs, go to the log for details.

```
[12:34:45] Checking for suspicious (large) shared memory segments [ Warning ]
[12:34:45] Warning: The following suspicious (large) shared memory segments have been found:
[12:34:45] Process: /usr/lib/firefox-esr/firefox-esr PID: 2452 Owner: b
b Size: 7.1MB (configured size allowed: 1.0MB)
[12:34:45] Process: /usr/lib/firefox-esr/firefox-esr PID: 2452 Owner: b
b Size: 1.1MB (configured size allowed: 1.0MB)
[12:34:45] Process: /usr/lib/firefox-esr/firefox-esr PID: 2452 Owner: b
b Size: 7.1MB (configured size allowed: 1.0MB)
[12:34:45] Process: /usr/lib/firefox-esr/firefox-esr PID: 2452 Owner: b
b Size: 1.1MB (configured size allowed: 1.0MB)
```

Figure 8: Some warnings are false positives; rkhunter can be configured to ignore such warnings after its first run.

Running as a Cron Job

Rkhunter can be automated even more by setting it to run as a cron job. The cron job is best run with MAIL-ON-WARNING set in /etc/rkhunter.conf. Since rkhunter must be run as root, use the root account's crontab. Before beginning, use crontab -l to see if root already has a crontab, and, if so, back it up before beginning.

To add rkhunter to the crontab, enter crontab -e while logged in as root, and, if this is your first time editing it, choose a text editor to use. There are many ways to enter times and dates with crontab, but the easiest is to enter the minutes and hours using a 24 hour clock, followed by the command. For example, if you want to run rkhunter at 3am, when you are not using your computer, the cron job entry would look like this:

```
00 03 * * * /usr/bin/rkhunter Z
--cronjob --update --quiet
```

The three asterisks indicate unused fields. The --cronjob option runs rk-

hunter without colors and without pausing, while the --update option updates the databases and --quiet runs the command without output.

MAIL-ON-WARNING sets email notifications, and you can check the log later.

Release Lags

If you examine the rkhunter project, you may notice that the latest release, version 1.4.6 was released two years ago. In addition, recent traffic on the project forums is rarely more than a few posts per month from only half a dozen people or so. Such evidence may lead you to wonder how current rkhunter is, and whether there is recent malware that it does not cover.

However, this concern seems to be groundless. A web search immediately reveals that the long time between releases has done little to stop rkhunter's use. After 14 years of development, rkhunter is a mature script, with a comprehensive awareness of different intrusion methods. Quite simply, there may be little left to add to rkhunter. Moreover, even if the current version does not

cover a particular rootkit, rkhunter's other tests, such as changes in key files, can probably detect evidence of a new rootkit, if not the particular kit.

However, if this release schedule disturbs you, you may prefer to run an alternative such as chkrootkit. But while administrators should keep themselves aware of new rootkits, on the whole, rkhunter remains a useful tool, especially if you follow up on its checks and examine its log in detail. Use it in cautious but good health. ■■■

Info

[1] Rootkit Hunter:
<http://rkhunter.sourceforge.net/>

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 cofounder of Prentice Pieces, a blog about writing and fantasy at <https://prenticepieces.com/>.

Shop the Shop

shop.linuxnewmedia.com

Missed an issue?

You're in luck.

Most back issues are still available. Order now before they're gone!

shop.linuxnewmedia.com



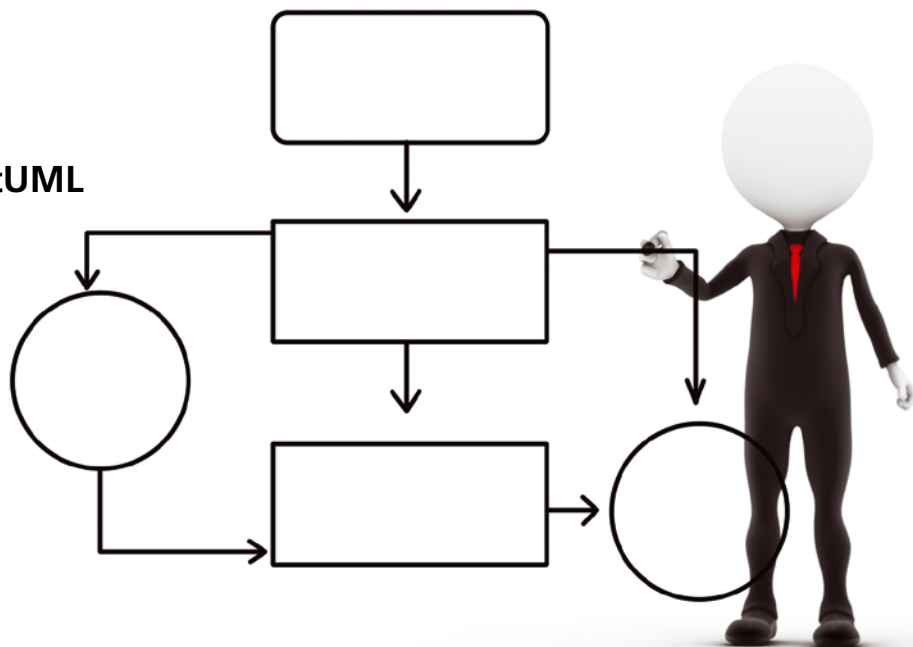
GET IT NOW!

SAVE TIME ON DELIVERY WITH OUR ALTERNATIVE PDF EDITIONS

ADS

Drawing diagrams with PlantUML

Picture This



With PlantUML, you can quickly create all kinds of diagrams using human-readable text and reuse them anywhere. *By Marco Fioretti*

Sooner or later, almost everybody needs to draw a diagram, whether to manage large amounts of information or organize group activities. For software developers and administrators (as well as business managers), diagrams based on the Unified Modeling Language (UML) make it easy to visualize complex systems, from software platforms to quality control departments, to see how they function.

UML stores drawings as plain text files, with a relatively intuitive syntax. If you are familiar with mind maps, you can often guess how to manually draw a diagram described by a UML file just by looking at it – even if you don't know UML! Because it is text-based, UML is extremely simple to generate with any software. You can produce hundreds of similar diagrams, with different parameters, by passing code generated on the fly by a simple shell script (or any other program) to a UML interpreter.

Additionally, UML is useful for efficiency, consistency, and version control. In a large project, UML interpreters can automatically find all the diagram descriptions in all of the project's files and generate the corresponding images in one fell swoop. The UML statements for a software algorithm or protocol's visual representation can be written as comments right above the code it visualizes in the software source file. Thanks to this integration, version control systems can store and track changes between different versions of many diagrams, just as

they do with software code. The same feature makes it easy to quickly check if a diagram is consistent with the code it represents. All of this makes UML really reusable.

PlantUML [1], an open source UML diagram tool, lets you quickly create many different types of UML diagrams from text-based descriptions. In addition, you can also generate several diagram types not included in the UML standard (Gantt, mind map, WBS). While some types of PlantUML diagrams, like timing and interface mock-ups, may only be interesting to software and hardware designers, the diagram styles shown here can benefit all users.

PlantUML is perfect for beginners and intermediate users. It runs everywhere Java runs, and it is easily included in scripts. While it offers a very basic graphical interface, I prefer to write diagrams in my favorite text editor and then launch PlantUML from the command line.

Installation

The fastest way to use PlantUML is via the official web server [2], but you also can easily install it on your computer. The only prerequisite is a working Java command, which is the core part of any Java Runtime Environment (JRE). For testing some diagram types, you may also need the Graphviz utility, but that is available as a binary package for most Linux distributions. On Linux, you can check if you have a working JRE by typing the following at the prompt:

```
#> java -version
```

On Ubuntu, if JRE is not installed, you will get a notification that “the program ‘java’ can be found in the following packages”:

```
* default-jre
... other Java packages...
```

In that case, just install the default JRE with:

```
#> sudo apt install default-jre
```

After installation (which consumed about 100MB of disk space on my computer), you can just download the Java container file, `plantuml.jar`, from the website and save it in a folder of your choice.

Getting Started

To create a diagram from instructions saved into a plain text file called `myuml-diagram.puml` (`.puml` is PlantUML's de-

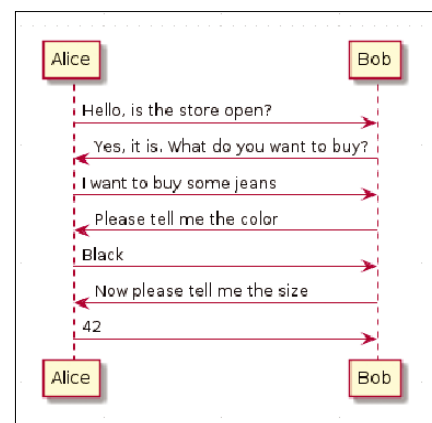


Figure 1: With a sequence diagram, you can show any structured transaction.

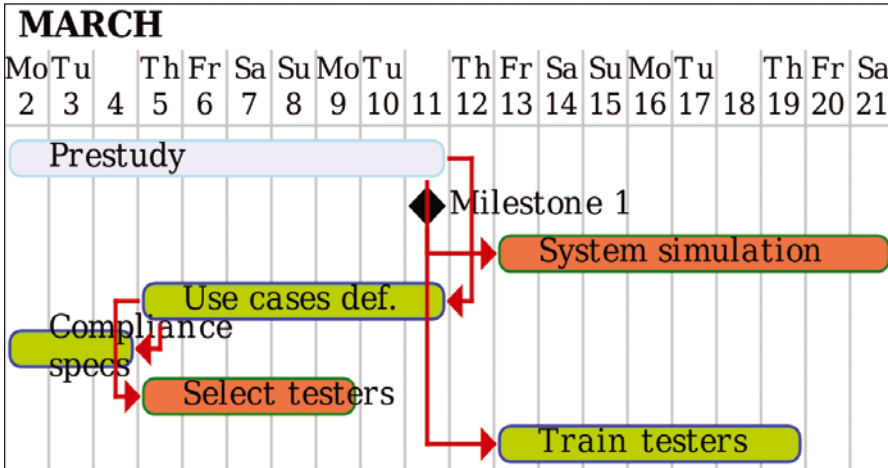


Figure 2: PlantUML’s Gantt diagrams may not be as fancy as those done in Microsoft Project, but they are much easier to create.

fault file extension), enter the following at the prompt or in a shell script:

```
java -jar /path/to/plantuml.jar ?
myumldiagram.puml
```

This command produces a PNG version of the diagram with the same name as the source file (in this case myumldiagram.png) in the same folder. Other supported output formats are SVG and LaTeX. You can assign a different name and location to a diagram if desired. With the -pipe option, PlantUML reads from standard input and “prints” the diagram on the standard output, which you can redirect wherever you want:

```
cat somefile.puml | java ?
-jar plantuml.jar ?
-pipe > /path/to/somefile.png
```

It is possible to process multiple UML source files with one call. If you give PlantUML a folder instead of a file, all the text files in that folder will be processed. Alternatively, you can use shell wildcards to only parse files with certain names or extensions:

```
java -jar /path/to/plantuml.jar ?
"source/*/*.c"
```

To put all the generated images into a common folder, pass its absolute path to PlantUML with the -o switch.

Even the simple examples shown in this article can take two to three seconds to run. If you plan to simultaneously process many UML files, take advantage

of the -failfast2 and -failfast options. -failfast2 checks if there are any errors in any of the files passed to the program and stops it without generating a diagram. -failfast parses the code, but stops as soon as it finds one error. You must decide which option best fits your needs.

By default, PlantUML saves each diagram’s full UML code inside the PNG file’s metadata section, making it very easy to retrieve. The following command

```
java -jar /path/to/plantuml.jar ?
-metadata statediagram.png > ?
statediagram.puml
```

will save all the UML statements that generated statediagram.png into a text file with the same base name, which allows quick changes to complex diagrams even if you did not write the code or have their UML description! A similar option, -checkmetadata, regenerates a diagram only if the source code it contains does not match the corresponding .puml file.

While the command-line switches mentioned here will be enough for most of your diagramming needs, you can learn about PlantUML’s other command-line options in the documentation [3] or launch PlantUML with the -help

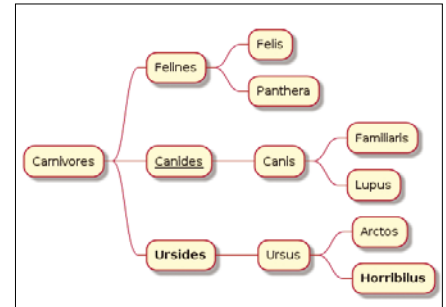


Figure 3: A taxonomy mind map diagram that uses asterisks for indentation.

switch. You can also use the -language switch to list all the UML keywords supported by PlantUML.

Sequence Diagrams

Sequence diagrams are frequently used in software and telecom documentation to describe exchanges between programs or communication protocols. However, as Listing 1 shows, they can describe any type of structured transaction (Figure 1). Listing 1 shows the UML source code for Figure 1.

You can see UML’s intuitiveness in Listing 1: Diagrams (with the exceptions discussed below) are enclosed between @startuml and @enduml statements; each line defines one element (in Figure 1, one transaction step). The direction of each step (i.e., if Alice is speaking to Bob, or vice versa) is given by the name order (line 2) or the arrow orientation (line 7). You can also draw sequencing exchanges among more than two entities.

Gantt Diagrams

You can use PlantUML instead of applications like Microsoft Project to make basic Gantt diagrams, with deadline and task dependencies. If you feed the code in Listing 2 to PlantUML, you will get a Gantt diagram like the one shown in Figure 2.

Listing 1: Sequence Diagram

```
01 @startuml
02 Alice -> Bob : Hello, is the store open?
03 Bob -> Alice: Yes, it is. What do you want to buy?
04 Alice -> Bob : I want to buy some jeans
05 Alice <- Bob : Please tell me the color
06 Alice -> Bob : Black
07 Alice <- Bob : Now please tell me the size
08 Alice -> Bob : 42
09 @enduml
```

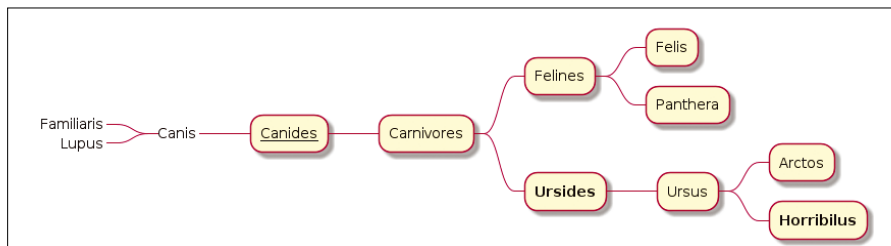


Figure 4: This mind map has the same information as shown in Figure 3, but it uses dashes and underscores for additional formatting.

Listing 2: Gantt Diagram

```
01 @startgantt
02 scale 4
03 Project starts the 2020/03/02
04 [Prestudy] lasts 10 days and is colored in Lavender/LightBlue
05 [Milestone 1] happens at [Prestudy]'s end
06 [System simulation] lasts 9 days and is colored in Coral/Green and starts 1 day
    after [Prestudy]'s end
07 [Use cases def.] lasts 7 days and ends at [Prestudy]'s end
08 [Compliance\nspecs] lasts 3 days and ends at [Use cases def.]'s start
09 [Select testers] is colored in Coral/Green
10 [Select testers] lasts 5 days and starts at [Use cases def.]'s start
11 [Train testers] starts 1 days after [Milestone 1]'s end and lasts 7 days
12 @endgantt
```

While PlantUML diagrams are not as polished as Microsoft Project diagrams, the source code can be quickly written (even on your smartphone) in human-readable text. Gantt diagrams use their own beginning (line 1) and end (line 12) keywords. In addition, using `scale` (line 2) makes the diagram four times bigger than PlantUML's default minimum size (note: this syntax does not work the same for all diagrams).

Mind Maps

Mind maps are an excellent way to represent the relationship between concepts or objects. To draw mind maps in PlantUML, you need to enclose all the mind map's elements between the keywords `@startmindmap` and `@endmindmap`, starting with the main element and indenting the following elements with asterisks or dashes. Figures 3 and 4 show two mind map layouts, with their respective source code in Listings 3 and 4. Functionally speaking, Figures 3 and 4 are identical, with the only differences being orientation and boxing. In Figure 4, the root element (*Carnivores*) is in the center, because using dashes (lines 6 to 9, Listing 4) instead of asterisks (lines 6 to 9, Listing 3) forces the whole *Canides* section to the

left. The underscores in lines 7 to 9 in Listing 4 tell PlantUML to not draw boxes around those elements. In mind maps, you can also use basic HTML tags for formatting, such as the underline tag (line 6 in Listings 3 and 4) and the bold face tag (lines 10 and 13 in Listings 3 and 4).

WBS Charts

PlantUML uses the Work Breakdown Structure (WBS) format for organizational charts. Figure 5 (source code shown in Listing 5) shows a company

Listing 3: Mind Map Diagram

```
01 @startmindmap
02 * Carnivores
03 ** Felines
04 *** Felis
05 *** Panthera
06 ** <u>Canides</u>
07 *** Canis
08 **** Familiaris
09 **** Lupus
10 ** <b>Ursides</b>
11 *** Ursus
12 **** Arctos
13 **** <b>Horribilus</b>
14 @endmindmap
```

org chart, but you can use this type of diagram for other things, like a manufacturing process or a school syllabus.

The biggest challenge with WBS diagrams in UML is balancing the diagram visually (avoiding empty areas alongside crowded areas). In general, you assign hierarchy with indentation. Visual balancing is achieved by changing the section order (when possible) and by using dashes instead of plus characters to change each box's orientation (Listing 5, lines 10, 13, 16, and 19). This syntax is compatible with Emacs' Org Mode. However, WBS diagrams are still a beta feature in PlantUML, so their look and feel, if not their syntax, may change in future versions.

Flow Diagrams

Flow diagrams show the several steps that make a task or algorithm and can be used for anything from statistical analyses to cake recipes. Figure 6, which is taken from the PlantBuddy GitHub site [4], shows a PlantUML flow diagram. While the language in Figure 6's source code (Listing 6) is intuitive, you need to know how to write a `while` cycle (lines 5 and 13) and how to use markers to delimit each phase of the flow. You use a colon at the beginning of a phase and semicolon at the end.

Diagram Formatting

UML and PlantUML offer many options for controlling text formatting, as well as style and appearance.

According to the manual [5], the `scale` command enlarges the generated image in several ways, including but

Listing 4: Alternative Mind Map Diagram

```
01 @startmindmap
02 * Carnivores
03 ** Felines
04 *** Felis
05 *** Panthera
06 -- <u>Canides</u>
07 ---_ Canis
08 ----_ Familiaris
09 ----_ Lupus
10 ** <b>Ursides</b>
11 *** Ursus
12 **** Arctos
13 **** <b>Horribilus</b>
14 @endmindmap
```

Listing 5: WBS Diagram

```

01 @startwbs
02 + Board of Directors
03 ++ <u>Legal/Finance</u>
04 ++ <b>R&D</b>
05 +++ Software
06 ++++ Design
07 ++++ Testing
08 +- Hardware
09 ++++ Digital Circuits
10 +++- PCB Boards
11 ++++ Analog Circuits
12 +++ System design
13 +-+ Modeling
14 ++++ Simulation
15 ++ <b>Production</b>
16 +- Procurement
17 +++ Manufacturing
18 ++++ Mechanics
19 ---- Electronics
20 @endwbs
    
```

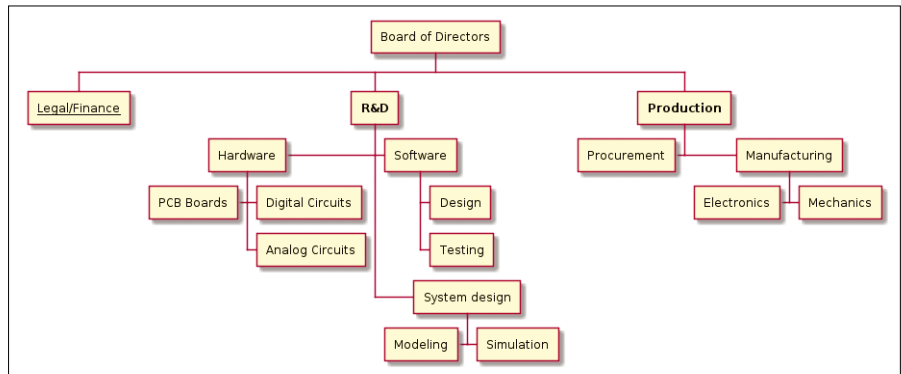


Figure 5: PlantUML’s WBS diagrams are the best choice for org charts or any other hierarchical structure.

not limited to these options:

```

scale 1.5
scale 200 width
scale 1400*700
    
```

However, this option did not work as documented on several of my diagrams.

To format text, PlantUML supports both basic HTML tags like bold and underline, and their equivalents in Creole format, such as:

```

@startuml
This is bold
This is italics
This is "monospaced"
This is --stroked--
This is underlined
This is ~waved~
    
```

The most important formatting command in PlantUML, `skinparam` (see [5] for its many attributes), lets you set fonts, background and text color, title properties, and much more. As an example, you can customize UML diagrams with `skinparam` to create real templates as shown in Listing 7.

`skinparam`’s only drawback is that the more you use it, the less portable your diagrams may become. This is especially true with fonts, which may not be available on all systems that will generate diagrams from your UML code.

Diagram Metadata

Besides the actual diagram, UML files can define several widgets that give context, structure, and generally make the diagrams themselves easier to understand. For brevity, Figure 7 shows all of these elements, plus it introduces a few other PlantUML features. The source code for Figure 7 is shown in Listing 8.

In Listing 8, the actual diagram is shown in lines 20 to 29; again, you can easily understand the source code by comparing it carefully with the drawing. Lines 2 to 18 demonstrate how to define all the widgets I mentioned, from header and footer to title, legend, and captions. Line 4 requires an explanation. While the `size:30` statement is self-explanatory, the `&bullhorn` and `&star` keywords tell PlantUML to insert the corresponding icons from the Open Iconic graphic library [6] (integrated in PlantUML) in their place.

Listing 6: Flow Diagrams

```

01 @startuml
02
03 :read moisture level;
04 :read humidity + temperature;
05 while (sensor readings) is (invalid)
06 :wait 2s;
07 :read moisture level;
08 :read humidity + temperature;
09 :increase counter;
10 if (MAX_READING_RETRIES reached)
11   then (yes)
12     :send controller to deep sleep;
13   endif
14 :submit sensor data;
15
16 @enduml
    
```

Custom Images

In addition to the integrated icons, you can also create custom images. Listing 9 shows the source for embedding a custom image, and Figure 8 shows its output.

To do this, define a `sprite` (a graphical element) with a label (tux in my example) and associate it with an actual image on your hard drive. Next, you can place the image where you want by calling its label with the syntax shown in line 3 of Listing 8. In addition to regular images, you may define and embed custom sprites [7], which are monochrome images that are defined similarly to tra-

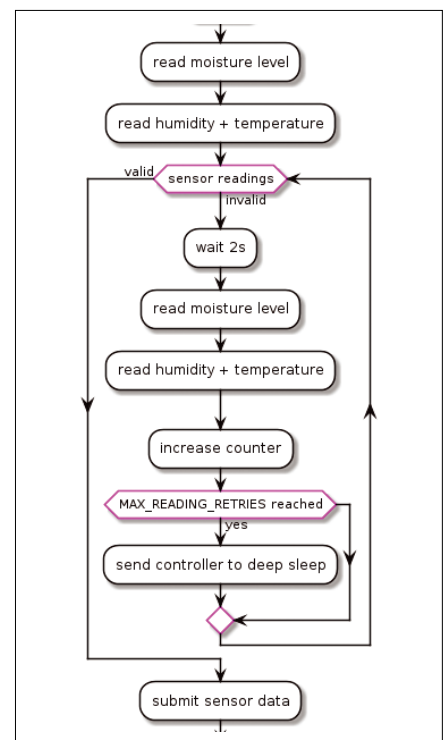


Figure 6: Part of a complex flow diagram, created on the fly from PlantBuddy’s GitHub site [4]. © PlantBuddy.

Listing 7: skinparam Customizations

```
01 skinparam backgroundColor transparent
02 skinparam monochrome true
03
04 skinparam titleBorderRoundCorner 15
05 skinparam titleBorderThickness 2
06 skinparam titleBackgroundColor Aqua-CadetBlue
07
08 skinparam classFontColor red
09 skinparam classFontSize 10
10 skinparam classFontName Aapex
```

ditional ASCII art right inside your UML code:

```
startuml
sprite $fool {
```

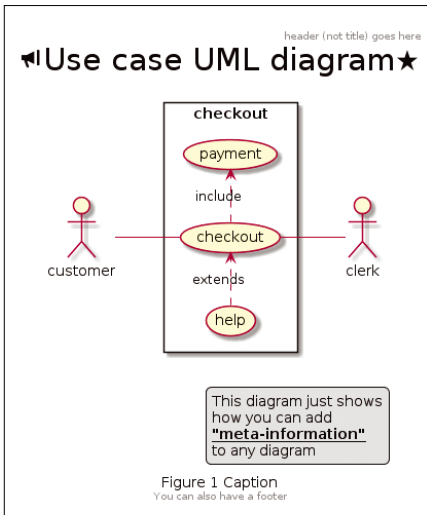


Figure 7: With PlantUML, you can add legends, headers, footers, and more.

```
FFFFFFFFFFFFFFF
F0123456789ABCF
F0123456789ABCF
FFFFFFFFFFFFFFF
}
```

This format is less esoteric than it may seem. Each hexadecimal digit inside the curly braces corresponds to one pixel of the image, and its value corresponds to the gray level of that pixel, with 0 being white, and F being

black. For further details on embedded sprites, see the PlantUML guide [5].

UML Programming

UML's creators have given it several capabilities of a real programming language.

You can include comments in your UML files to make them more readable. Single line comments start with one sim-

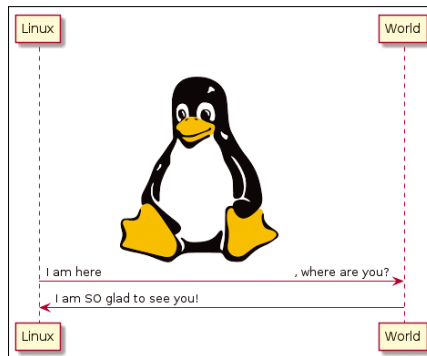


Figure 8: You can easily embed any image you want in your diagrams.

ple quote ('), and multiline comments are enclosed by '/' and '/' tags.

Lines that start with exclamation marks (!) are preprocessing directives, with which you can give PlantUML general instructions or make it interact with your system. The following

```
!log Now generating a Gantt Diagram
```

tells PlantUML to log a message, by writing it to its standard output.

PlantUML also uses variables and functions. PlantUML variables can only contain strings or integer numbers; it is good practice to give them names starting with a dollar sign (\$), like Perl. Besides storing values for later use, variables can also be used for simple flow control, to decide what to draw where:

```
Alice -> Bob : Are you free tonight?
!if ($day == "Saturday")
Alice <- Bob : yes
!else
Alice -> Bob : no, I'm sorry
!endif
```

PlantUML supports three different types of functions: built-in, void, and return functions. Built-in functions, which have names starting with %, include operators like %strlen (string length) or %substr (substring extraction). Other built-in functions give access to system information, like file location or the current date.

Void and return functions are recognizable, because, like variables, their names must start with a dollar sign (\$).

Listing 8: Adding Widgets

```
01 @startuml
02 caption Figure 1 Caption
03 title
04 <size:30>&bullhorn>Use case UML diagram<star></size>
05 endtitle
06
07 header
08 header (not title) goes here
09 endheader
10
11 center footer You can also have a footer
12
13 legend right
14 This example shows
15 how you can add
16 <u><b>"meta-information"</b></u>
17 to any diagram
18 endlegend
19
20 left to right direction
21 skinparam packageStyle rectangle
22 actor customer
23 actor clerk
24 rectangle checkout {
25 customer -- (checkout)
26 (checkout) .> (payment) : include
27 (help) .> (checkout) : extends
28 (checkout) -- clerk
29 }
30 @enduml
```



The difference is simple between void and return functions and summarized in Listing 10.

Void functions directly insert something (normally one statement) into the UML diagram description. In Listing 10, the `$warning` function (line 1) will print a warning from user `$source` to user `$recipient` whenever it is called, obviously replacing `$source` and `$recipient` with the current values of those variables. Return functions perform a calculation or string processing and return the result (lines 4 and 5); they are often called from other functions.

In both void and return functions, you can define local variables, which are not visible from outside the function, and set default values for the arguments.

Code Management

In UML, as in software in general, reuse is almost always good, and duplication is almost always bad. You can put all the configuration and formatting functions that you regularly use in one file and make PlantUML load it as follows:

```
java -jar /path/to/plantuml.jar 
-config "./config.cfg" dir1
```

To do the same thing with the actual drawing code (e.g., the initial part of a sequence exchange or flow diagram) that you want to duplicate in other diagrams, you can write the corresponding code once (maybe packaging it as a function) in one file. Then `!include` that file in other files every time you need it as follows:

```
@startuml
!include common-diagrams-section.iuml
.. your other UML code here
@enduml
```

By doing this, any change in that single file will be seen and reloaded by all the files that include it the next time you run PlantUML, just as if you had copied and pasted `common-diagrams-section` in each file. The `!include` directive supports URLs, so you may even load code directly from the Internet or your company network. By default, a file can only be included once. You may include the same file several times in your code with the `!include_many` directive, but think twice before doing it: It may make your code unmanageable.

Syntax-wise, you may also put several independent blocks of common code in the same file, each within its own `@startuml/@enduml` statements, and identify each of those blocks by their number. A directive like `!include myuml.txt!1` will then load only the *second* (numbering starts from 0) block within `myuml.txt`. To improve code readability, you can assign names to a block

```
@startuml(id=SOME_IDENTIFIER)
```

and then include only that block with this statement

```
!include foo.txt!SOME_IDENTIFIER
```

Pros and Cons

With the basics covered, there are some general pros and cons to using PlantUML for creating diagrams.

PlantUML uses plain text, and very simple text at that. On one hand, this makes it extremely quick to write, without wasting time by clicking on countless options or dragging around lines and boxes.

On the other hand, this efficiency results in some loss of control. A PlantUML diagram is “inferred by a deterministic algorithm in the rendering process” [1]. Therefore, describing diagrams with text instead of just drawing them on your computer is like writing Markdown instead of using a word processor. While you can focus on your diagram’s structure, the algorithms baked into PlantUML do the actual drawing. If the developers of the next PlantUML version decide to change those algorithms (or some of the graphic libraries), all your diagrams could assume a new look and feel when you upgrade, whether you like it or not.

While this may be a showstopper for some users, don’t let it scare you, because there is nothing as reusable as a

Listing 9: Embedding Images

```
01 @startuml
02 sprite tux linuxtux.png
03 Linux->World : I am here <$tux>, where are you?
04 Linux<-World : I am SO glad to see you!
05 @enduml
```

Listing 10: Void and Return Functions

```
01 !function $warning($source, $recipient)
02 $source --> $recipient : Warning! You have been hacked!
03 !endfunction
04 !function $halfvalue($a)
05 !return $a/2
06 !endfunction
```

PlantUML diagram, and it has so many uses [8]. You can automatically create or process UML code with any major programming language. Above all, you can write, copy, and paste UML code in LibreOffice, Microsoft Office, WordPress, Etherpad, MediaWiki, and many other editors or content management systems. With the right plugins, all those programs will read that code and convert it into embedded diagrams without a hitch. What’s not to like? ■■■

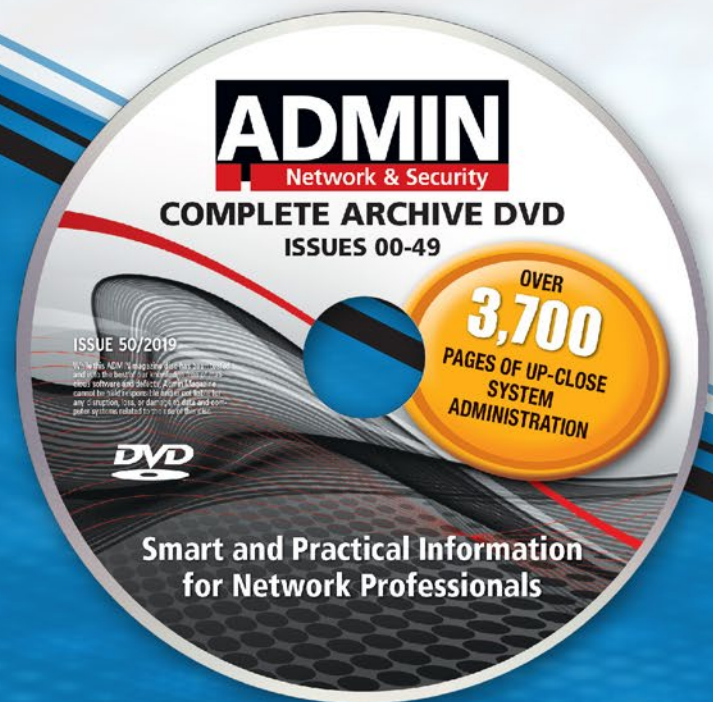
Info

- [1] PlantUML: www.plantuml.com
- [2] PlantUML web server: www.plantuml.com/plantuml
- [3] PlantUML command-line options: <https://plantuml.com/command-line>
- [4] PlantBuddy diagram: <https://github.com/anoff/plantbuddy>
- [5] PlantUML Reference Guide: <http://plantuml.com/guide>
- [6] Open Iconic: <https://useiconic.com/open>
- [7] Custom sprites: <https://plantuml.com/sprite>
- [8] Using PlantUML: <https://plantuml.com/running>

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://freenknowledge.eu>).





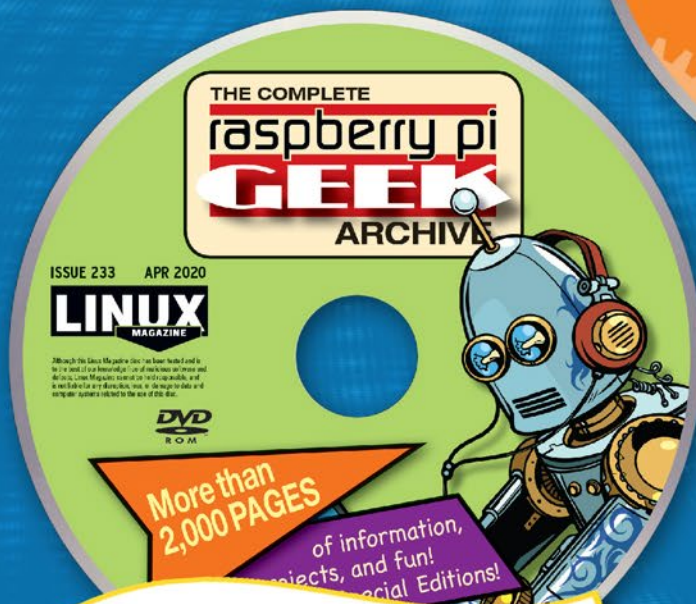
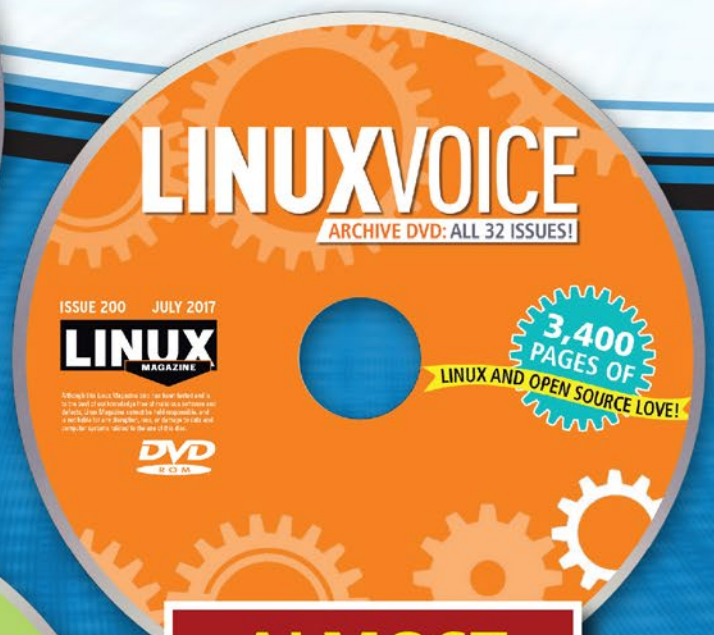
Complete Your Open Source Library with Archive DVDs!

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



ALMOST SOLD OUT!

New Release!

All-in-one performance analysis with standard Linux tools

What's Going On?

Experienced sys admins always use the same commands to analyze problematic system loads on Linux. Mike Schilli bundles them into a handy Go tool that shows all the results at a glance.

By Mike Schilli

When the performance analysis book by Brendan Gregg [1], which I had long been waiting for, appeared recently, I devoured it greedily. I even had access to a prerelease so that in January I was able to give the readers of *Linux Magazine* some tips on the kernel probes for throughput measurement by means of the Berkeley Packet Filters [2]. Performance guru Gregg also explains at the beginning of the book how he often uses classic command-line tools to find out what is causing a struggling server to suffer.

10 Commands

He starts off with a list of 10 command-line utilities (Listing 1) [3], which every

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.



Unix system understands, checks everyday things – such as how long the system has been running without rebooting (`uptime`) or whether there is anything noticeable in the system log (`dmesg`). The `vmstat` command looks for processes waiting for their turn on a CPU. It also checks if all RAM is occupied and the system is wildly swapping. Similarly, `free` shows the available memory.

Entering `pidstat` visualizes how processes are distributed on the system's CPUs; `iostat` determines whether the data exchange with the hard disk is the bottleneck. The utilization of individual CPUs is illustrated by `mpstat`, which also shows whether a single process is permanently blocking an entire CPU. The `sar` commands collect statistics about the network activity on a regular basis, including the throughput data. The best known of the tools, `top`, gives an overview of RAM usage and running processes, but according to Gregg, should be called last.

As the Go language offers both excellent interaction with external processes and can conjure up fast and attractive terminal UIs, my idea was to fire off all

the analysis commands more or less at the same time and then to display the neatly structured results in separate panes of the same terminal window (Figure 1).

Listing 2 shows the `runit()` function, the central part of the newly created command-line utility `Greggalizer` – the name I gave the analysis program in honor of Gregg. The function receives a command in the form of an array of strings, executes it, and returns the con-

Listing 1: brendan-gregg-commands

```
# uptime
# dmesg | tail
# vmstat 1
# mpstat -P ALL 1
# pidstat 1
# iostat -xz 1
# free -m
# sar -n DEV 1
# sar -n TCP,ETCP 1
# top
```

Lead Image © Maksym Yemelyanov, 123RF.com



tents of its standard output to the caller. This comes courtesy of the `exec` pack-

age's `Command()` function in the Go standard library, which accepts a command

with parameters. The `Command()` call executes the requested program with the specified arguments and returns an object, on which the code then calls the `Output()` method to retrieve its output as a character string.

Since various Unix command-line tools structure their text output using tabs, but the widgets in the terminal UI cannot cope with these special characters, line 17 defines a regular expression that matches and later replaces tabs with spaces. Since regexes expect the tab character to be `\t` and the expression is double quoted, Listing 2 needs to double the backslash (`\\t`) to preserve the single backslash within double quotes.

Go has to compile regular expressions. However, nothing can go wrong with a single tab, so line 17 uses `MustCompile()`, which does not return any error code, but would blow up in your face if a regex failed to compile. `ReplaceAllString()` then replaces all tabs in the out byte array with spaces, and `runit()` returns the result as a string to the caller.

Drawing the Test Grid

Listing 3 shows the `main()` program, which starts the Greggalizer. In the initial import statements, the code pulls in the `Termdash` project from the GitHub server, to draw and manage the

Listing 2: `runit.go`

```
01 package main
02
03 import (
04     "fmt"
05     "os/exec"
06     "regexp"
07 )
08
09 func runit(argv []string) string {
10     out, err := exec.Command(
11         argv[0], argv[1:]...).Output()
12
13     if err != nil {
14         return fmt.Sprintf("%v\n", err)
15     }
16
17     r := regexp.MustCompile("\\t")
18     return r.ReplaceAllString(
19         string(out), " ")
20 }
```

```
Greggalizer
[/usr/bin/uptime]
18:11:25 up 24 days, 2:47, 3 users, load average: 0.43, 0.83, 0.72

[/bin/bash -c dmesg | tail -10]
[2076572.988728] hid-generic 0003:04F2:0939.0046: input.hidraw4: USB HID v1.11 Mouse [PixArt USB Optical Mouse] on usb-0000:00:14.0-2.2/input0
[2079528.077120] usb 3-2.2: USB disconnect, device number 71
[2079877.724834] usb 3-2.2: new low-speed USB device number 72 using xhci_hcd
[2079877.832847] usb 3-2.2: New USB device found, idVendor=04F2, idProduct=0939
[2079877.832855] usb 3-2.2: New USB device strings: Mfr=1, Product=2, SerialNumber=0
[2079877.832860] usb 3-2.2: Product: USB Optical Mouse
$

[/usr/bin/vmstat 1 1]
procs-----memory-----swap-----io-----system-----cpu-----
 r  b   supd  free  buff  cache  si  so   bi  bo   in  cs  us  sy  id  wa  st
0  0 3814576 144028 92720 1470876 7  8  36  76  2  2  1  0 98  1  0

[/usr/bin/mpstat -P ALL]
Linux 4.4.0-171-generic (mybox) a02/01/2020 a_x86_64_a(4 CPU)
06:11:25 PM CPU      %usr   %nice    %sys  %iowait    %irq   %soft  %steal   %guest   %gnice   %idle
06:11:25 PM all      0.66    0.01    0.22    1.23    0.00    0.01    0.00    0.00    0.00    97.87
06:11:25 PM 0        0.69    0.01    0.23    1.25    0.00    0.00    0.00    0.00    0.00    97.81
06:11:25 PM 1        0.70    0.01    0.24    1.79    0.00    0.00    0.00    0.00    0.00    97.25
06:11:25 PM 2        0.64    0.02    0.21    0.22    0.00    0.05    0.00    0.00    0.00    98.86
$

[/usr/bin/pidstat 1 1]
Linux 4.4.0-171-generic (mybox) a02/01/2020 a_x86_64_a(4 CPU)
06:11:25 PM UID      PID      %usr  %system  %guest   %CPU  Command
06:11:26 PM 1000     911      0.99   0.00    0.00   0.99  1 pidstat
06:11:26 PM 119      1918    0.00   0.99    0.00   0.99  0 nagios3
06:11:26 PM 1000     2501    0.99   0.00    0.00   0.99  1 compiz
06:11:26 PM 1000     30520   2.97   1.98    0.00   4.95  0 chromium-browser
06:11:26 PM 1000     30531   0.99   0.00    0.00   0.99  0 chromium-browser
$

[/usr/bin/iostat -xz 1 1]
Linux 4.4.0-171-generic (mybox) a02/01/2020 a_x86_64_a(4 CPU)
avg-cpu:  %user   %nice %system %iowait  %steal   %idle
           0.66    0.01    0.23    1.23    0.00    97.87

Device:            rrqm/s   wrqm/s     r/s     w/s    rkB/s    wkB/s avgrq-sz avgqu-sz   await r_await
w_await  svctm  %util
$

[/usr/bin/free -m]
              total        used         free   shared  buff/cache   available
Mem:           3868         2201          139       1013        1527         344
Swap:          4096         3725           371

[/usr/bin/sar -n DEV 1 1]
Linux 4.4.0-171-generic (mybox) a02/01/2020 a_x86_64_a(4 CPU)
06:11:26 PM IFACE   rxpck/s   txpck/s   rxkB/s   txkB/s   rxcmp/s   txcmp/s  rxmcast/s  Zifutil
06:11:27 PM lo         0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00
06:11:27 PM wlan0      0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00
06:11:27 PM docker0    0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00
06:11:27 PM eth0       0.00     0.00     0.00     0.00     0.00     0.00     0.00     0.00
$

[/usr/bin/sar -n TCP,ETCP 1 1]
Linux 4.4.0-171-generic (mybox) a02/01/2020 a_x86_64_a(4 CPU)
06:11:27 PM active/s  passive/s   iseg/s   oseg/s
06:11:28 PM 0.00      0.00      0.00     0.00

06:11:27 PM atptf/s   estres/s  retrans/s  isegerr/s  orsts/s
06:11:28 PM 0.00     0.00     0.00     0.00     0.00
$
```

Figure 1: The Greggalizer fires off all of Listing 1's performance analysis commands and displays the results in separate window panes.

terminal UI. The array of string slices in the `commands` variable starting in line 16 define the various commands that the program will run, along with their parameters.

Some commands, such as `pidstat`, accept both an update interval and – optionally – the maximum number of

iterations to perform their task. For example, `pidstat 1` prints out tasks currently being processed by the kernel, as an infinite loop in one second intervals. A second parameter specifies the maximum number of calls; `pidstat 1 1` terminates after the first result, just like the Greggalizer wants.

You might have noticed that the top command is missing from the list; this is because it uses terminal escape sequences for its display, similar to the Greggalizer's terminal UI. Its output would have needed to be preprocessed and has for that reason been excluded. Also, because `exec` can only execute

Listing 3: greggalizer.go

```

01 package main
02
03 import (
04     "context"
05     "fmt"
06     "github.com/mum4k/termdash"
07     "github.com/mum4k/termdash/cell"
08     "github.com/mum4k/termdash/container"
09     "github.com/mum4k/termdash/linestyle"
10     "github.com/mum4k/termdash/terminal/termbox"
11     "github.com/mum4k/termdash/terminal/terminalapi"
12     "github.com/mum4k/termdash/widgets/text"
13 )
14
15 func main() {
16     commands := [][]string{
17         {"/usr/bin/uptime"},
18         {"/bin/bash", "-c",
19             "dmesg | tail -10"},
20         {"/usr/bin/vmstat", "1", "1"},
21         {"/usr/bin/mpstat", "-P", "ALL"},
22         {"/usr/bin/pidstat", "1", "1"},
23         {"/usr/bin/iostat", "-xz", "1", "1"},
24         {"/usr/bin/free", "-m"},
25         {"/usr/bin/sar",
26             "-n", "DEV", "1", "1"},
27         {"/usr/bin/sar",
28             "-n", "TCP,ETCP", "1", "1"},
29     }
30
31     t, err := termbox.New()
32     if err != nil {
33         panic(err)
34     }
35     defer t.Close()
36
37     ctx, cancel := context.WithCancel(
38         context.Background())
39
40     widgets := []container.Option{
41         container.ID("top"),
42         container.Border(linestyle.Light),
43         container.BorderTitle(
44             " Greggalizer ")}
45
46     panes := []*text.Text{}
47
48     for _, command := range commands {
49         pane, err := text.New(
50             text.RollContent(),
51             text.WrapAtWords())
52         if err != nil {
53             panic(err)
54         }
55
56         red := text.WriteCellOpts(
57             cell.FgColor(cell.ColorRed))
58         pane.Write(
59             fmt.Sprintf("%v\n", command), red)
60         pane.Write(runit(command))
61
62         panes = append(panes, pane)
63     }
64
65     rows := panesSplit(panes)
66
67     widgets = append(widgets, rows)
68
69     c, err := container.New(t, widgets...)
70     if err != nil {
71         panic(err)
72     }
73
74     quit := func(k *terminalapi.Keyboard) {
75         if k.Key == 'q' || k.Key == 'Q' {
76             cancel()
77         }
78     }
79
80     err = termdash.Run(ctx, t, c,
81         termdash.KeyboardSubscriber(quit))
82     if err != nil {
83         panic(err)
84     }
85 }

```

simple executable programs with arguments, it cannot process commands like `dmesh | tail -10` directly. Two commands linked by a pipe can only be understood by the shell. Therefore, line 18 simply uses `bash -c` to pass the whole command to a bash shell as a string for execution.

In line 31, `termbox.New()` defines a new virtual terminal, which the `defer` call in line 35 neatly collapses when the program ends. The `widgets` slice in line 40 defines the widget panes in the window and populates the UI with the main "top" widget, which draws a frame and writes the program title at the top.

The `panes` slice in line 46 defines pointers to the various stacked text widgets in the top window. The `for` loop from line 48 creates a text widget with scrolling content for each of the commands. This means that the widgets can handle longer output from chatty commands without going haywire or losing content. The user can scroll up the output with the mouse wheel if the content size exceeds the dimensions of the widget.

Stacking Building Blocks

Into each of these text widgets, line 58 first writes the name of the scheduled command in red and then passes the command to `runit()` to have it executed. It then intercepts the output and feeds it to the text widget. All the widgets end up in the `panes` slice, each of them appended

at its end thanks to the `append` command (line 62).

Line 69 then uses the `...` operator to individually pass the elements in the slice to the `container.New()` function courtesy of the `termdash` library. The `Run()` function in line 80 builds and manages the UI until the user presses `Q` to terminate the endless event loop. The keyboard handler intercepts this event starting at line 74 and calls the `cancel()` function of the background context previously defined in line 37, which pulls the rug out from under the terminal UI's processing loop.

But how does the graphical user interface stack the individual widgets on top of each other, while giving each one the same amount of space, no matter how many commands the user defines? A trick is needed here, because as a layout method for vertically stacking two widgets, `termdash` only supports the `SplitHorizontal` function. It accepts two widgets and a percentage value that determines how much space the upper widget gets in relation to the lower one.

Figure 2 shows how any number of widgets can be stacked in steps of two: At the top of each partial stack is the conglomerate of all previously stacked widgets, and at the bottom the new widget that the algorithm attaches. The percentage value, which determines the ratio of the upper widget height to the lower one, needs to change dynamically, depending on the number of wid-

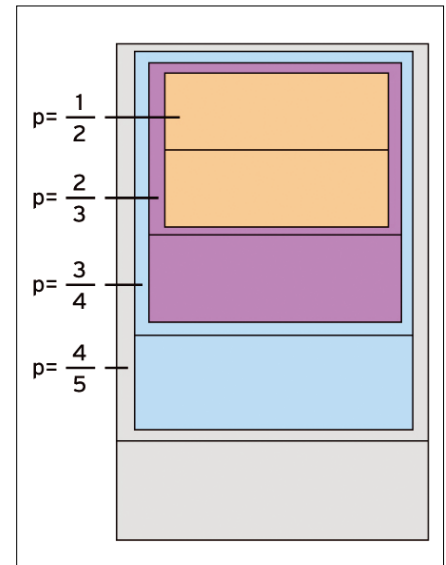


Figure 2: Building up the panel structure in steps of two.

gets already in the group, so that in the end all individual widgets really appear to be the same size.

If there is only one widget at the top, it gets exactly 50 percent of the space (orange box), just like the one at the bottom. But if there are already three widgets on top and one is added at the bottom, the widget group on top gets 75 percent of the space and the new widget 25 percent (blue box). Accordingly, the function `panesSplit()` from Listing 4 takes a slice of text widgets and initializes the resulting group widget rows by adding the first text widget.

The `for` loop then iterates over the remaining widgets to be packed start-

Listing 4: pane-splitter.go

```

01 package main
02
03 import (
04     "github.com/mum4k/termdash/container"
05     "github.com/mum4k/termdash/widgets/text"
06     "github.com/mum4k/termdash/linestyle"
07 )
08
09 func panesSplit(
10     panes []*text.Text) container.Option {
11     var rows container.Option
12
13     if len(panes) > 0 {
14         rows =
15             container.PlaceWidget(panes[0])
16         panes = panes[1:]
17     }
18
19     for idx, pane := range panes {
20         itemsPacked := idx + 2
21
22         rows = container.SplitHorizontal(
23             container.Top(rows),
24             container.Bottom(
25                 container.PlaceWidget(pane),
26                 container.Border(
27                     linestyle.Light),
28             ),
29             container.SplitPercent(
30                 100*(itemsPacked-1)/itemsPacked,
31             )
32         }
33
34     return rows
35 }

```


The sys admin's daily grind: traceroute

Advanced Tracing

Like every admin, Charly regularly uses the classic traceroute tool. If unfriendly digital natives interfere with an ICMP filter, he simply switches to a clever alternative like LFT. *By Charly Kühnast*

Practically every admin uses the classic traceroute tool at more or less regular intervals. This gets me all the more irritated when I find myself in a hotel with a WiFi network where the admin has completely disabled ICMP. Apart from the fact that this causes more trouble than benefits in what is by definition a

public network, it can be easily circumvented.

The first version of traceroute was written in 1988 by a certain Van Jacobsen – Van is his first name, not an honorific. To be able to trace the path of packets through the web, Jacobsen came up with a clever method. He sent test packets through the Internet to a

defined destination and increased the time to live (TTL) value for each packet.

The first packet is assigned a TTL of one. Each router that transports the packet further reduces the TTL by one. Once the TTL reaches a value of zero, the router sends it back with an ICMP TTL exceeded message. By successively increasing the TTL, Jacobsen got the packets back from routers that were further and further away and was able to follow the path of the packet until it finally reached its destination.

This does not work if the remote peer suppresses ICMP messages. However, traceroute has evolved over the years. It has been able to use an alternative TCP-based method that relies on TCP SYN packets for quite some time. Figure 1 shows two traceroutes to the same destination, the BBC web server (*bbc.co.uk*). The first call gets stuck at some point, probably due to an ICMP filter. The second one uses TCP SYN packets – it gets to its destination unhindered.

Alternative traceroute tools, such as MTR [1], which continuously repeats the trace and thus helps to detect occasional packet losses, take things one step further. Another very interesting tool is Layer Four Traceroute (LFT [2]). It can handle other transport methods and thus makes it through most firewalls. In addition, it can output whose network blocks the packet is passing through, including the number of the autonomous system responsible for it (Figure 2).

It is therefore worthwhile to take a closer look at the different traceroute variations – if only to keep your blood pressure down during your next hotel stay. ■■■

Info

[1] “Sys Admin’s Daily Grind: Step Counter” by Charly Kühnast, *Linux Magazine*, issue 119, October 2010, p. 47

[2] LFT: <http://freshmeat.sourceforge.net/projects/LFT>

```
root@glas:~# traceroute bbc.co.uk
traceroute to bbc.co.uk (151.101.0.81), 30 hops max, 60 byte packets
 1 192.168.1.254 (192.168.1.254) 0.345 ms 0.379 ms 0.486 ms
 2 100.72.0.1 (100.72.0.1) 2.457 ms 2.376 ms 2.401 ms
 3 100.127.1.6 (100.127.1.6) 6.542 ms 6.216 ms 6.346 ms
 4 100.127.1.7 (100.127.1.7) 7.186 ms 8.163 ms 7.083 ms
 5 100.127.1.11 (100.127.1.11) 6.005 ms 6.374 ms 6.139 ms
 6 185.22.45.30 (185.22.45.30) 10.169 ms 6.764 ms 7.304 ms
 7 * 62.140.26.189 (62.140.26.189) 7.180 ms 7.098 ms
 8 ae-1-3107.edge5.Frankfurt1.Level3.net (4.69.163.18) 6.770 ms 7.547 ms
03 ms
 9 * * *
10 * * *
11 * * *
12 * * *
13 * * *
14 *^C
root@glas:~# traceroute -T bbc.co.uk
traceroute to bbc.co.uk (151.101.64.81), 30 hops max, 60 byte packets
 1 192.168.1.254 (192.168.1.254) 0.407 ms 0.584 ms 0.717 ms
 2 100.72.0.1 (100.72.0.1) 3.442 ms 3.450 ms 3.426 ms
 3 100.127.1.6 (100.127.1.6) 7.772 ms 7.351 ms 7.736 ms
 4 100.127.1.7 (100.127.1.7) 8.035 ms 8.688 ms 9.156 ms
 5 100.127.1.11 (100.127.1.11) 8.160 ms 7.825 ms 7.447 ms
 6 185.22.45.30 (185.22.45.30) 11.193 ms 7.460 ms 7.222 ms
 7 62.140.26.189 (62.140.26.189) 7.380 ms * *
 8 * * *
 9 213.19.200.114 (213.19.200.114) 12.973 ms 13.240 ms 13.814 ms
10 151.101.64.81 (151.101.64.81) 16.583 ms 16.396 ms 13.926 ms
root@glas:~#
```

Figure 1: Where the classic traceroute fails, a simple -T (for TCP-SYN) often does the trick.

```
root@glas:~# lft -A -N bbc.co.uk:443

Tracing _____

TTL LFT trace to 151.101.192.81:443/tcp
 1 [AS198949] [PRIVATE-ADDRESS-CBLK-RFC1918-IANA-RESERVED] 192.168.1.254 24.5ms
 2 [ASN?] [SHARED-ADDRESS-SPACE-RFCTBD-IANA-RESERVED] 100.72.0.1 24.2ms
 3 [ASN?] [SHARED-ADDRESS-SPACE-RFCTBD-IANA-RESERVED] 100.127.1.6 24.2ms
 4 [ASN?] [SHARED-ADDRESS-SPACE-RFCTBD-IANA-RESERVED] 100.127.1.7 24.2ms
 5 [ASN?] [SHARED-ADDRESS-SPACE-RFCTBD-IANA-RESERVED] 100.127.1.11 24.2ms
 6 [AS60294] [RIPE-185/DE-DGNO-20130326] 185.22.45.30 24.2ms
 7 [AS3356] [RIPE-C3/FRANKFURT-CUSTOMER-SERIAL-LINKS5] 62.140.26.189 600.0ms
** [neglected] no reply packets received from TTL 8
 9 [AS3356] [RIPE-213/AMSTERDAM--CUSTOMER-LINKS] 213.19.200.114 24.2ms
10 [AS54113] [SKYCA-3] [target] 151.101.192.81:443 24.3ms

root@glas:~#
```

Figure 2: Knows where it's going: LFT makes it through most firewalls and returns the network blocks it has passed through.



Universal Package Systems and competing standards

Status Report

Billed as the future of package management, universal package systems like Snappy and Flatpak have failed to live up to their promise. *By Bruce Byfield*

Remember universal package systems? Although AppImage [1], the earliest universal package system, was first released in 2004, the concept did not capture much attention until a decade later, when Canonical released Snappy [2] and Red Hat released Flatpak [3]. Each was presented as the next generation of package managers, usable by any distribution, and as a means to reduce the number of rival technologies. Yet in 2020, both Snappy and Flatpak have receded into the background, and the deb and RPM package management systems continue to dominate Linux, leaving the question of why Snappy and Flatpak did not fulfill their promises.

Two quick searches on DistroWatch reveal that, out of the 273 active distros listed, 39 support Flatpak [4], and 35 support Snap packages [5]. At first, those may sound like respectable numbers, until you realize that a much more arcane deviation from the norm, like distros that do not ship systemd, can boast 99 distros. Moreover, those figures con-

sist mainly of major distros that support Flatpak and Snap – often both – but still depend primarily on traditional package managers.

Theory vs. Practice

A serious drawback to universal packages is that, to be truly universal, they require that each distribution be structured the same as others. Despite efforts like the Linux Standard Base, this requirement is simply not met. Many distros continue to place key files in different positions. For this reason, the promise that universal packages would reduce the amount of work needed to ship packages has no practical chance of being realized.

Similarly, although still in discussion, Alexander Larsson, one of the original Flatpak developers, has championed the placement of Flatpak in containers, which has also fallen short of theory. For one thing, containers are optional, and the level of security often varies. Just as importantly, a study at North Carolina

State University in 2017 [6] showed that over 356,000 community-contributed container images averaged 156 vulnerabilities, while 3,800 official images averaged 76. In most cases, these vulnerabilities were rated high severity. In other words, while containers might be secure in themselves, what is in them may not be secure.

As Adrian Coyle, who summarized the North Carolina State study, pointed out, new packages often perpetuate vulnerabilities by borrowing dependencies from older ones for convenience. In fact, even a containerized package that is up to date when created may later prove to have vulnerabilities but continue to be used. With the introduction of automatic updates and tools such as Docker Security Scanning, these problems may be mitigated, but, even so, much still depends upon the conscientiousness of a package's maintainer. Consequently, the promised advantages of universal packages often are yet to be realized.

Lead image © Sebastian Duda, 123RF.com

Going Against Custom

The technical challenges are only part of the reason for the lukewarm reception of universal package managers. It is true that the deb and RPM package managers were designed in eras of limited memory, rather than today's abundance. However, for the average user, that hardly matters. In fact, when installing on older or limited systems, like many bottom-of-the-line laptops, the efficient memory use can still be relevant. Snaps, for example, may be better suited for use with containers, but, overall, the incentive to move away from traditional packages simply hasn't been there once the universal package manager's novelty subsided.

Part of the problem may be infrastructure. When universal packages were introduced, the rationale was that they would be built by upstream developers. By removing the distributions from the delivery, in theory, packages could get into the hands of users more quickly. The trouble is that was a new role for upstream developers, and one they have not always undertaken. This should not be surprising, because upstream developers' operations are simply not geared for it. Often, they may not have the numbers to provide such service. As a result, packaging has remained largely in the hands of distro developers, who have the experience and infrastructure for it. While distro developers appear perfectly willing to produce universal packages of different formats, the role tends to be a side-show, secondary to maintaining traditional packages.

Even more importantly, despite what the makers of universal packages maintain, a functional distribution is not a matter of technology so much as policy – specifically, of quality control. Josh Triplett [7], a long time Debian contributor, explains the reason for Debian's dominance among currently active distributions: “Debian without the .deb format would still be Debian; Debian without Debian Policy would just be SourceForge or rpmfind” – that is, repositories of packages and source code that you could find on a web page with no overall quality control.

The Debian Policy Manual [8] is a lengthy, much revised document that explains what goes in to both a Debian

package and a general release. It explains what a package must or may contain, and how it interacts with other packages. It includes where files and logs should be placed, and dozens of other details. No other distribution is so specific about such matters, although almost all have similar documents.

As a Debian package moves from the Unstable repository to Testing and Stable, it is closely examined for compliance with Debian Policy. Moreover, as John Goerzen [9], another veteran Debian contributor, notes, this initial quality control is reinforced throughout a package's life history through “*unattended-updates, needrestart, debsecan, and debian-security-support* [...] Debian's security team generally backports fixes rather than just say ‘here's the new version,’ making it very safe to automatically apply patches. As long as I use what's in Debian Stable, all layers mentioned above [everything] will be protected using this scheme.”

Originally, universal packages lacked most of these quality control measures. Today, ones like automatic updates are mostly standard, but the kind of quality control offered by Debian is not instituted over night. Quality control in major distros like Debian is the result of decades, and universal packages cannot be expected to equal them in a few years, especially since quality control is a specialist role that many developers do not favor.

The Future of Universal Packages

Universal packages do have one advantage: They make having multiple versions of a package on the same system easier. However, when traditional packages are numbered, multiple versions of libraries and even desktop packages like LibreOffice can coexist on the same system. Besides, multiple versions are a special case that do not affect many users unless they mix package repositories. Nor is there any reason why users should not mix package systems as they choose.

In a sense, universal packages are a modern version of static tarballs, which include all the dependencies needed to install a package. However, the majority of distributions rejected that model for package management years ago, and the

improvements offered by universal packages are not enough to make them preferable to traditional systems.

Neither Flatpak nor Snap are about to go away, especially since they are backed by major Linux corporations. Free software has never been slow to use new technologies, and universal packages are no exception. Still, their current status is far from “the future of application deployment” promised on Flatpak's front page or “the new bullet-proof mechanism for app delivery and system updates” [10] announced by Snappy. Instead, as a much-reprinted famous xkcd comic observed [11], the attempt to reduce the number of competing standards has only added to the confusion, and the benefit is small. ■■■

Info

- [1] AppImage: <https://appimage.org/>
- [2] Snappy: <https://snapcraft.io/>
- [3] Flatpak: <https://www.flatpak.org/>
- [4] 39 support Flatpak: <https://distrowatch.com/search.php?ostype=All&category=All&origin=All&basedon=All¬basedon=None&desktop=All&architecture=All&package=Flatpak&rolling=All&isosize=All&netinstall=All&language=All&defaultinit=All&status=Active#simple>
- [5] 35 support Snap packages: <https://distrowatch.com/search.php?ostype=All&category=All&origin=All&basedon=All¬basedon=None&desktop=All&architecture=All&package=Snap&rolling=All&isosize=All&netinstall=All&language=All&defaultinit=All&status=Active#simple>
- [6] North Carolina State University Study 2017: <http://dance.csc.ncsu.edu/papers/codaspy17.pdf>
- [7] Josh Triplett: <https://lists.debian.org/debian-devel/2016/06/msg00287.html>
- [8] Debian Policy Manual: <https://www.debian.org/doc/debian-policy/>
- [9] John Goerzen: <https://changelog.complete.org/archives/date/2017/04/29>
- [10] “the new bullet-proof mechanism for app delivery and system updates”: <https://www.markshuttleworth.com/archives/1434>
- [11] xkcd comic: <https://xkcd.com/927/>

Tangram integrates social media services in a single app

Convenient Connection

Tangram lets you track social media portals like Facebook and Twitter, as well as web-based messengers like Whatsapp and Telegram, in a single application window.

By Christoph Langner

Anyone who uses more than just a handful of online services can quickly lose track of all the open social media portals in their browser. In addition, the proliferation of Like buttons endangers your privacy, allowing Facebook and other companies to follow you wherever you go. Your social media presence becomes even more complicated if you use more than one account, for example, one for business and one for private activities. Providers generally do not support seamless switching between the different accounts.

The desire for more systematic control of social media accounts and web applications has given rise to a new class of desktop tools that offer unified management of web applications through a single user interface. Tangram [1] is a promising candidate that is built on the Gnome libraries and the Gnome web browser (formerly known as Epiphany). The goal of the Tangram project is to “... improve integration of web applications into the desktop, especially the Gnome desktop.”

Tangram displays all your web applications in a unified window. The different applications appear as tabs, allowing you to move effortlessly among the services, or you can find the application you're looking for in an easy navigation menu.

Flatpak

Flatpak lets developers bundle their application into containers that can be installed across multiple distributions. You will, however, need to configure Flatpak support. The Flatpak project website explains the procedure required for configuring support in most common Linux variants [7].

Tangram is similar to programs like Franz [2] or Rambox [3]. These applications act as social media browsers and messaging apps, making it easier to separate your web work from personal communication. Both Franz and Rambox implement the idea well, but under the hood, they both use the Electron framework [4], which adds a large amount of ballast and complexity. Franz and Rambox are both commercial tools that offer free community editions. Tangram, on the other hand, is an all free project intended specifically for the Gnome environment, although users on other desktops can also run it.

You won't find Tangram in the package sources of major Linux distributions [5]; however, the developers have published a Flatpak [6] on Flathub to facilitate the installation on most systems (see the “Flatpak” box). If you have Arch Linux, you can alternatively install Tangram via the Arch user repository. The AUR helper, Yay, lets you set up the program with a single command:

```
yay -S tangram
```

Looking Around

When first launched, Tangram looks like a simple browser. The application window is empty for the most part, al-

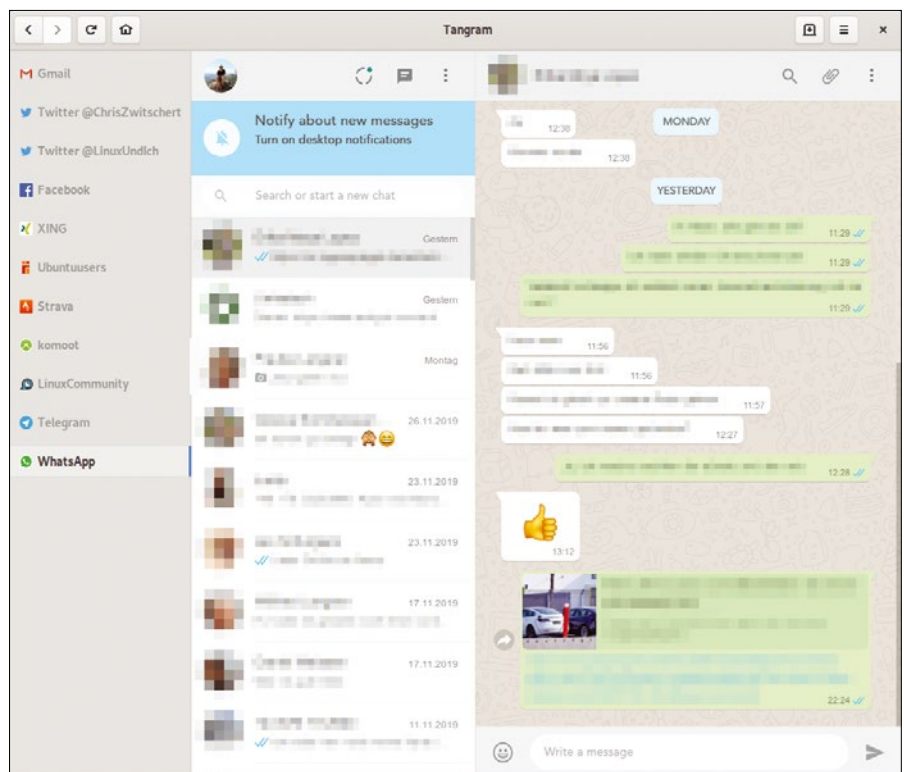


Figure 1: Tangram is a single application that lets you combine social media services like Facebook and messengers in the style of Whatsapp or Telegram.



Figure 2: Tangram gives you complete freedom in the choice of services. You can even integrate forums and news portals with comment functions.

though you will see an address bar and a *Done* button. To add a service, type its URL into the address line in the usual way, and press Enter to open the page. Then log in and go to the main page of the service, such as its dashboard or profile page.

Then click on the green *Done* button in the Tangram user interface to firmly anchor the page in Tangram. If necessary, you can still edit the name and URL of the page manually. *Add* lets you close the setup dialog; Tangram now stores the service as a tab on the left side of the sidebar.

Repeat the process for all of your web services, including Facebook, Twitter, Instagram, Gmail, or your employer's web-based email portal or wiki (see also the "Whatsapp" box). Click on the plus icon in the window bar to add a new site. The sidebar will gradually fill up with the desired services (Figure 1).

The individual entries work like tabs in a web browser. If required, you can shift the position to the header or to any other border of the application window (Figure 2). Tap on the Hamburger menu and select the *Tabs position* entry.

When you click on a link, Tangram opens the page in its own application window. The navigation icons in the window bar let you jump backwards or

forwards in the history. The Home button takes you to the start page of the service you initially configured.

If you right-click on a link, a context menu opens. The *Open link* entry is equivalent to a normal left click. The *Open link in new tab* option lets you open the selected link in an external web browser.

Strictly Isolated

Tangram passes notifications to the desktop messaging system (Figure 3). It is important to make sure that the desired services support the function. For example, for Gmail, you first have to activate the notifications in the *Desktop Notifications* section of the web portal settings. In our lab, notifications worked for Gmail and Telegram, but not for Twitter, Facebook, or Whatsapp.

Unlike a web browser, Tangram strictly separates the individual tabs from each other. So the Facebook tab doesn't know that you are also logged in to Twitter. This design reinforces your privacy and also lets you create multiple accounts for the same service in Tangram. To differentiate between the accounts, simply add the account name to the tab name. On the downside, for services that use the same credentials, such as Gmail,

Whatsapp

In our tests, all of the common web services, from Facebook to Twitter and Telegram, worked fine. Only Whatsapp proved a little stubborn: Instead of the QR code necessary for linking to a mobile phone, Tangram only shows a rotating circle. The code appears, but it takes several minutes for the system to get organized [8].

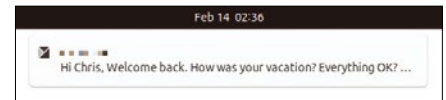


Figure 3: Tangram forwards notifications about new messages to the desktop environment. This service does not work with all services.

Google Calendar and Contacts, you will need to log in multiple times.

Conclusions

Tangram offers a lean and powerful alternative to the Electron apps Franz and Rambox. The program integrates smoothly with the desktop, and it saves resources, because the software does not have the overhead of Javascript ballast. The Tangram application does still have room for improvement. For example, developers should provide DEB and RPM packages, in addition to the Flatpak, until Tangram finds its way into the repositories of the major Linux distributions.

Another drawback is the somewhat reticent notifications display. It would be ideal if every service could tell you when a new message arrives, but currently notification only works on some of the platforms. ■■■

Info

- [1] Tangram: <https://github.com/sonnyp/Tangram>
- [2] Franz: <https://meetfranz.com/>
- [3] Rambox: <https://rambox.pro/>
- [4] Electron: <https://electronjs.org>
- [5] Repology: <https://repology.org/project/tangram/versions>
- [6] Flathub: <https://flathub.org/apps/details/re.sonny.Tangram>
- [7] Setting up Flatpak: <https://flatpak.org/setup>
- [8] "Whatsapp Doesn't Work": <https://github.com/sonnyp/Tangram/issues/66>

A modern library interior with a curved staircase and bookshelves. The scene is brightly lit, with a blue overlay at the bottom. The text is centered on the blue overlay.

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

FREE DVD! ALL THE SOFTWARE YOU NEED!

THOUSANDS OF FREE TOOLS IN EASY REACH!

SWITCH TO LINUX NOW! 2019 Edition

GETTING STARTED WITH LINUX

Learn how to set up a Linux system to:

- Listen to Music
- Surf the Web
- Play Games
- Process Photos
- and Much More!

MORE Powerful, MORE Secure, MORE Fun!

JOIN THE **LINUX REVOLUTION!** **LINUX** SPECIAL

WWW.LINUX-MAGAZINE.COM

300+ BEST BASH COMMANDS **SAVE 15%** on Linux Certification See details inside

LINUX SHELL HANDBOOK

2019 Edition

SUPERCHARGE YOUR LINUX SKILLS

Travel Light with fast and graceful keyboard commands

Power at Your Fingertips

- Manipulate text strings
- Pipe and redirect output
- Monitor processes
- Manage users and groups
- Create easy automation scripts

Keep this comprehensive guide as a permanent command reference!

WWW.LINUX-MAGAZINE.COM

FREE DVD! LibreOffice Full Version

Includes full versions for Windows, macOS, and Linux

2019 Edition

DISCOVER LibreOffice

Free Office Suite

Find out why 75 million users have stopped paying for office software!

Edit and Save MS Office Files!

Create your own:

- Word processing docs
- Spreadsheets
- Presentations
- Databases

Use open file formats that other programs can read and import!

Also inside:

- Extensions
- Macros
- Templates
- and more

WWW.LINUX-MAGAZINE.COM

LINUX SPECIAL **101 COOL LINUX HACKS** **2020 EDITION**

101 COOL LINUX HACKS

Inspirational tricks and shortcuts for Linux geeks

- Repair your bootloader
- Cure the Caps Lock disease
- Tricks with terminal output
- Disable your webcam and mic
- Run C one-liners in the shell
- Undelete lost files
- Ignore case in file names

PHONING IN Sync your phone with a Linux desktop

QUICK SWITCH Change to a second distro using chroot

LINUX SPECIAL

WWW.LINUX-MAGAZINE.COM

Reenvisioning SSH with ShellHub

21st Century SSH

ShellHub offers an innovative approach to remote access with minimal reconfiguration of a firewall. *By Chris Binnie*

Secure Shell (SSH) is one of the most popular Linux services. With the global IPv4 address space shortage, it often becomes necessary to SSH into machines that are sitting behind a NAT-enabled router.

Rather than opening up network ports and then forwarding traffic individually to all your specific LAN devices, clearly it would be much better to access devices via a centralized point.

A new, natty piece of software called ShellHub [1] solves this headache nicely. ShellHub creates an SSH server inside your local network, allowing you to forward inbound SSH traffic to your other machines without having to mess around with the individual port forwarding settings for all your devices. Think of ShellHub as an alternative to the popular `sshd` daemon (OpenSSH) on your LAN.

In Closer

Figure 1 is a simple schematic showing the ShellHub architecture. From outside the LAN, you connect to the ShellHub server, either from the command line or a browser window. The firewall/router is configured to pass traffic to the predefined static IP address and port number of the ShellHub server on the LAN, performing network address translation as required.

A user who is connected to the ShellHub server can then use the ShellHub web interface (or a command-line interface) to initiate connections with

other devices on the LAN. The remote user can thus connect with all devices even though the ShellHub server itself is the only system requiring special firewall attention.

On Your Marks

The excellent ShellHub uses Docker Engine and Docker Compose, taking a microservices approach to make the software easier to develop and maintain.

Consult your package manager's documentation for more on how to set up ShellHub for your Linux distro. I'm using Linux Mint atop Ubuntu 18.04.

The first step is to set up Docker:

```
$ apt install docker.io docker-compose
```

To make sure that Docker Engine starts when the machine reboots, run the following command:

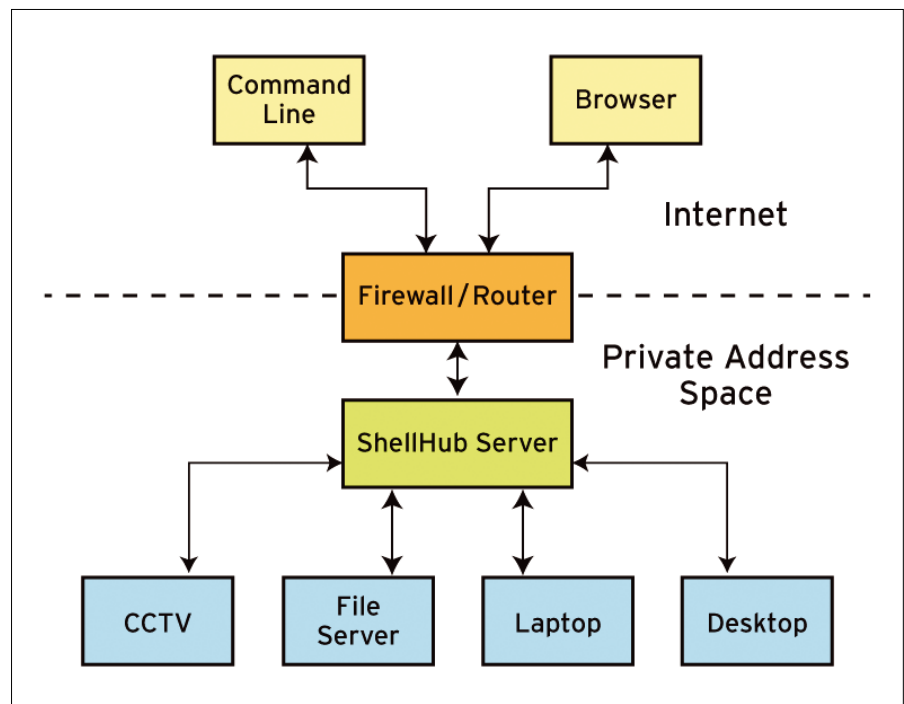


Figure 1: A remote user just has to access the ShellHub server to obtain access to other devices on the LAN.

Docker Hub Login

If you haven't logged into Docker Hub for a while, you may have missed a slight change in policy over the last few months.

Docker now places more emphasis on regular logins (even for public images). As a result, you may need to login again. Reset your password online if required, and then run the login command:

```
$ docker login
```

```
$ systemctl enable docker
```

The next task is cloning ShellHub's GitHub repository [2] and selecting the correct branch (and therefore version):

```
$ git clone -b v0.0.4 https://github.com/shellhub-io/shellhub.git
shellhub-v0.0.4
```

ShellHub development is proceeding quickly, but be aware that ShellHub is still in Beta. I was running version 0.0.2 just a couple of weeks ago and already it is at version 0.04, so make sure you get the latest and most stable version.

Go to the cloned directory next by using this command:

```
$ cd shellhub-v0.0.4
```

Before proceeding, fire up the excellent keygen tool, which is included in the repository's `bin/` directory:

```
$ ./bin/keygen
```

and create an RSA key pair.

Docker Inside

The ShellHub service is built on Docker; you'll need a working Docker [3] configuration to set up ShellHub. You're advised to exercise some patience (anything up to 15 minutes apparently) when running Docker Compose to bring up ShellHub.

Try the following command:

```
$ docker-compose up -d
```

If this command doesn't immediately work, see the box entitled "Docker Hub

Login." For more help with Docker Compose issues, see "There May Be Trouble Ahead."

Starting Up

After running the command to bring up the resources listed in the ShellHub Docker Compose configuration, you'll see lots of Docker-esque output as containers are created and started. After putting the kettle on and coming back a few minutes later, you should be greeted with some logging information, right at the end of the output:

```
Creating shellhub-v002_emq_1... done
Creating shellhub-v002_ssh_1... done
Creating shellhub-v002_ws_1... done
Creating shellhub-v002_gateway_1 ... done
```

To check whether the installation was successful, run the following command to see which containers are now running:

```
$ docker ps
```

The seven containers that run a ShellHub server are shown in Figure 2. They include a MongoDB container and con-

tainers running from images with names such as:

```
shellhubio/shellhub-gateway:v0.0.4
shellhubio/shellhub-ws:v0.0.4
shellhubio/shellhub-ssh:v0.0.4
shellhubio/shellhub-ui:v0.0.4
shellhubio/shellhub-api:v0.0.4
```

The next step is adding a user to our shiny, new ShellHub server. From the same directory as in the cloned GitHub repository, run the following command:

```
$ ./bin/add-user chrisbinnie
nothingtoseehere

User added: chrisbinnie
Tenant ID: fca338ad-8801-4a917-8487-5de263dadbd2
```

The username in this case is *chrisbinnie* and the password is *nothingtoseehere*.

Connecting

The moment of truth has arrived. Open up a browser and navigate to the following HTTP page:

There May Be Trouble Ahead

If you receive an error when Docker Compose tries to start up, you might need to get the Docker Compose version required by your ShellHub version. On the Docker website, the compatibility matrix [4] mentions that, for version 3.7 to be used in the Docker Compose file (v3.7 is the version that my ShellHub uses), you need the following:

- Docker Compose (version in the `docker-compose.yml` file): version 3.7
- Docker Engine: 18.06.0+

To check your Docker Engine version, you can run the following command:

```
$ docker version
```

And, for the Docker Compose version, use this command:

```
$ docker-compose version
```

As ShellHub is up-to-date, I took the following steps to update Docker Compose. First run the following command to remove the old version:

```
$ apt purge docker-compose
```

From the Docker Compose GitHub repos-

itory [5], download the source code for the latest version (version 1.25.4 at the time I wrote this article):

```
$ curl -L "https://github.com/docker/compose/releases/download/1.25.4/docker-compose-$(uname -s)-$(uname -m)" -o /usr/bin/docker-compose
```

The eagle-eyed among you will see from the long, somewhat unwieldy command that the resulting Docker Compose release is saved within the directory `/usr/bin` on my local machine (adjust this location to your needs). Once it's been downloaded, you can make the file executable with:

```
$ chmod +x /usr/bin/docker-compose
```

Now, from inside the `shellhub-v0.0.4` directory in the cloned GitHub repository file path, run the following command to bring Docker Compose up again (you might have to run the `docker login` command first):

```
$ docker-compose up -d
```

Ta-da! The screeds of output as our ShellHub server is brought up confirm that Docker Compose is happy.

CONTAINER ID	IMAGE	NAMES	COMMAND	CREATED	STATUS	PORTS
207671d0393a	shellhubio/shellhub-gateway:v0.0.2	shellhub-v002_gateway_1	"/usr/local/openrest_..."	14 seconds ago	Up 13 seconds	0.0.0.0:80->80/tcp
29ad0072f6df	shellhubio/shellhub-ws:v0.0.2	shellhub-v002_ws_1	"/bin/sh -c /ws"	15 seconds ago	Up 14 seconds	
43e49b45be25	shellhubio/shellhub-ssh:v0.0.2	shellhub-v002_ssh_1	"/bin/sh -c /ssh"	16 seconds ago	Up 15 seconds	0.0.0.0:22->2222/tcp
55cea3b769a5	emqx/emqx:v3.1.1	shellhub-v002_emq_1	"/usr/bin/docker-ent_..."	19 seconds ago	Up 16 seconds	4369/tcp, 5369/tcp, 6369/tcp, 8080/tcp, 8083-8084/tcp, 8883/tcp, 0.0.0.0:1883->1883/tcp,
9.0.0.0:18083->18083/tcp, 11883/tcp	shellhubio/shellhub-ui:v0.0.2	shellhub-v002_ui_1	"nginx-debug -g 'dae_..."	19 seconds ago	Up 17 seconds	80/tcp
e9b066eebdad	shellhubio/shellhub-api:v0.0.2	shellhub-v002_api_1	"/bin/sh -c /api"	20 seconds ago	Up 18 seconds	
cb45f0b5a69c	mongo:3.4.19	shellhub-v002_mongo_1	"docker-entrypoint.s_..."	6 minutes ago	Up 6 minutes	27017/tcp

Figure 2: Seven containers are running the ShellHub server, courtesy of Docker Compose's ShellHub config.

```
http://localhost/login
```

Figure 3 shows the slick-looking login screen that awaits.

Figure 4 shows the main window

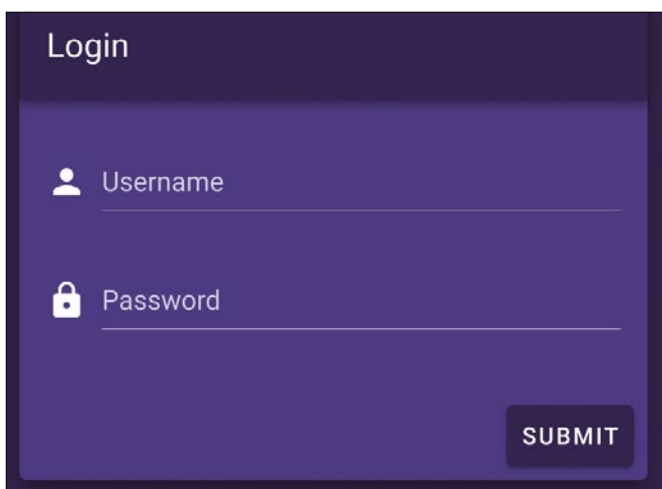


Figure 3: Happiness is a login screen that accepts your username and password.

that appears once you have logged in. On the left-hand side are a few options; the rest of the screen shows you the status of devices that are connected to your ShellHub server.

I'll cheat a little here in order to stay focused on the dashboard and the UI. Instead of installing a ShellHub agent (the equivalent of the SSH client) on another device, I'll install it on the ShellHub server itself and then explain what to do with other machines later on.

The process of installing a ShellHub agent is re-

markably simple. Click the *ADD DEVICE* link on the dashboard, and you'll be presented with a *curl* command, as shown in Figure 5.

This command essentially passes a tenant ID to ShellHub and then asks the *localhost* (the ShellHub server name, which I'll change in a minute) to install the ShellHub agent so that it can be accessed over SSH.

Then check the ShellHub dashboard again to see if the ShellHub agent has been registered. Figure 6 shows that a device is registered to the ShellHub server successfully.

Oh So Pretty

The *View All Devices* link in Figure 6 lets you open the Devices dialog (Figure 7). The command-line icon in lower-right corner of the Devices dialog lets you open an SSH prompt. As you can see in Figure 7, devices are referenced by MAC

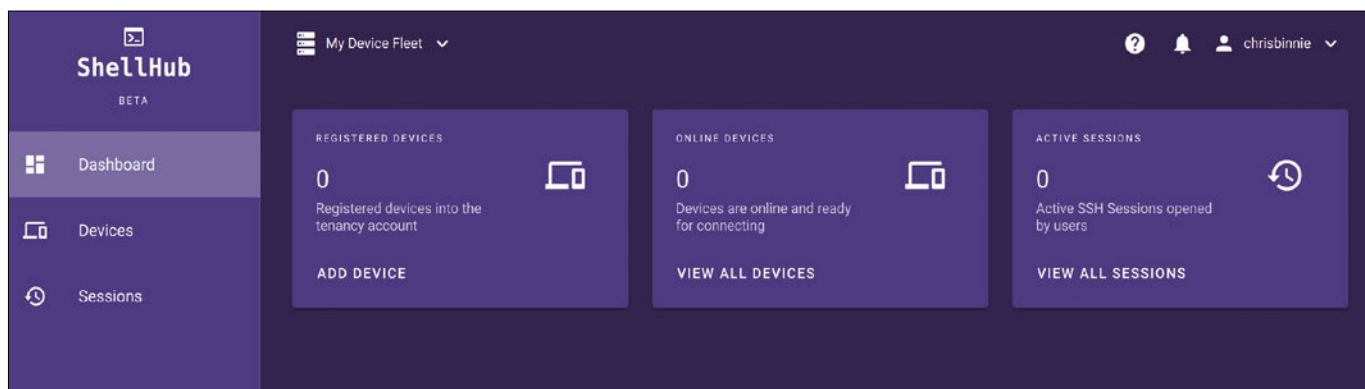


Figure 4: A dashboard view, once logged in, courtesy of the nicely designed ShellHub UI.

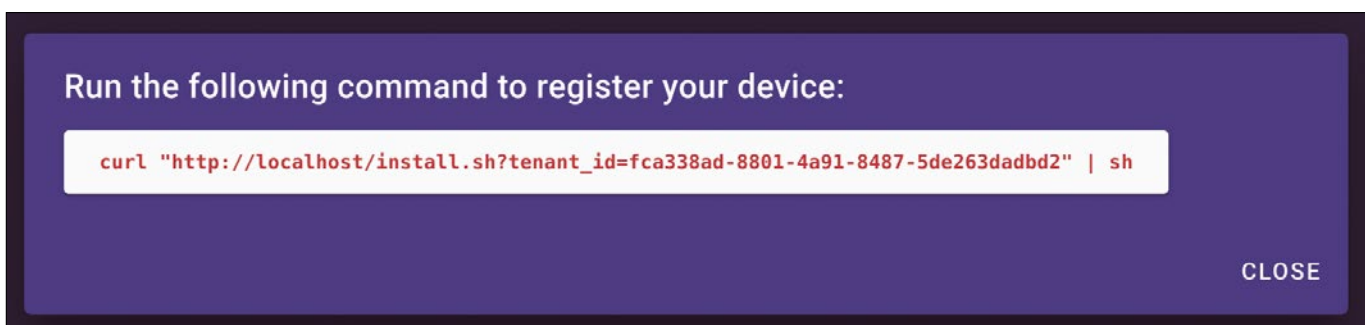


Figure 5: Simplicity incarnate. Run this command to register a device you want SSH access to with ShellHub. Note that *localhost* needs changing; I'll come back to that later.

The dashboard shows three sections:

- REGISTERED DEVICES:** 1 Registered devices into the tenancy account. Button: ADD DEVICE
- ONLINE DEVICES:** 1 Devices are online and ready for connecting. Button: VIEW ALL DEVICES
- ACTIVE SESSIONS:** 0 Active SSH Sessions opened by users. Button: VIEW ALL SESSIONS

Figure 6: Excellent, I've connected a client (the ShellHub agent) to the ShellHub server. What was previously shown as 0 devices now shows as 1.

The 'Devices' page displays a table with the following data:

Online	Name	Operating System	SSHID	Actions
	3c-6a-a7-15-23-7c	Linux Mint 19	chrisbinnie.3c-6a-a7-15-23-7c@localhost	

Figure 7: After clicking on the *View All Devices* section shown in Figure 6, you're presented with the ability to open up an SSH prompt.

address, but you have the option of giving each device a handy nickname.

If you click the little shell prompt icon, you'll be prompted for a username and password. These credentials are for

the machine's users directly and not the ShellHub server's usernames.

Figure 8 shows that the browser-based SSH client works well (I'm using Google Chrome on Linux Mint).

Note that the SSHID column in Figure 7 allows you to SSH directly over the command line to a machine connected to ShellHub. Simply click the *copy* box and then enter the credentials. See the

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

Now available on ZINIO: bit.ly/Linux-Pro-ZINIO



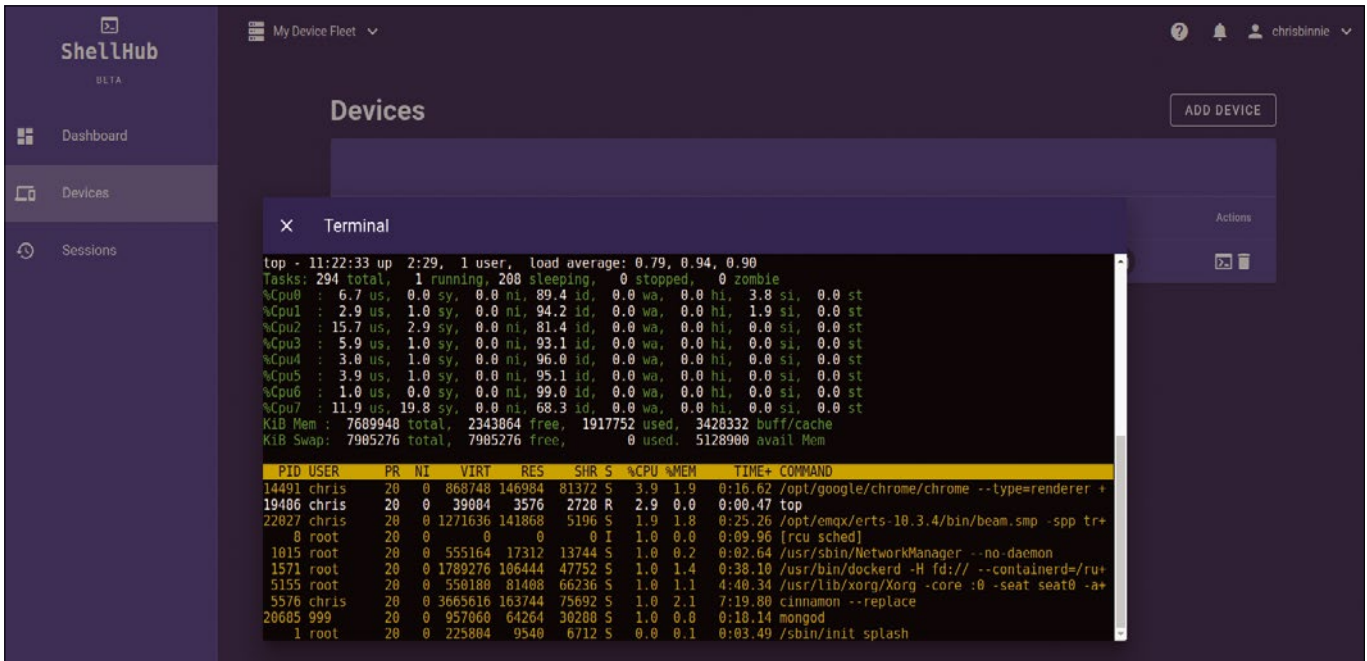


Figure 8: Running the `top` command in a terminal window, within a *Browser* tab sitting atop a desktop environment.

project docs for more on connecting to another device [6].

From the command line, the connections launched through the ShellHub GUI to devices on the LAN have the following format:

```
$ ssh <USER>@<SSHID>
```

For example:

```
$ ssh linux-user.2
3c-6a-a7-15-23-7c@localhost
```

With a little trial and error, you should be able to quickly get used to the format of the SSH command from all popular SSH clients. If you want to connect a device later on but aren't sure of the ID of that ShellHub's agent, the docs suggest running this command:

```
$ docker exec shellhub agent info
```

Then look for the `sshid` entry in the JSON that is output:

```
sshid: "namespace.2
device_name@address"
```

Network Tweaks

To connect to your ShellHub server from outside the LAN, assign a static IP address to the machine running ShellHub. Then

follow the router's manual for how to configure port forwarding on TCP port 22 for traffic addressed to the ShellHub server. (TCP port 22 is the default port for SSH, but you can also change the port assignment using the Docker Compose YAML file.) You only need to configure remote access through the firewall to the ShellHub server, and then you can connect to other devices on your LAN via ShellHub.

I made a point of mentioning the `localhost` change that you'll need to make. Aside from the simple router change, the other thing to do is, where possible, add a local alias for the ShellHub server's IP address in the `/etc/hosts` file on any devices that are running a ShellHub agent. Or, you could, of course, create a DNS entry somehow that points at the static LAN IP address. Adding a reference to the ShellHub server in the host's file or a DNS record will allow you to address the server by hostname rather than IP address.

Conclusion

Now that you have walked through the quick and simple setup of ShellHub, I hope you will be tempted to use it. It's slick, lightweight, and under active development. ShellHub lets you access multiple devices on your LAN through SSH with minimal firewall complications. You could also use ShellHub within the LAN itself, with no public

access, if you just need a way for local devices to communicate.

Centralized management via a nicely crafted user interface offers easier admin for multiple devices. ShellHub provides a clever route to fixing a problem that's been around for a while. I hope you enjoy using it. ■■■

Info

- [1] ShellHub: <https://shellhub-io.github.io>
- [2] ShellHub GitHub Repository: <https://github.com/shellhub-io>
- [3] Docker: <https://www.docker.com>
- [4] Compatibility Matrix: <https://docs.docker.com/compose/compose-file/compose-versioning/>
- [5] Docker Compose GitHub Repository: <https://github.com/docker/compose/releases>
- [6] Connecting to a ShellHub Device: <https://shellhub-io.github.io/guides/connecting-device>

Author

Chris Binnie's latest book, *Linux Server Security: Hack and Defend*, shows how hackers launch sophisticated attacks to compromise servers, steal data, and crack complex passwords, so you can learn how to defend against such attacks. In the book, Binnie also shows you how to make your servers invisible, perform penetration testing, and mitigate unwelcome attacks. You can find out more about DevOps, DevSecOps, Containers, and Linux security on his website: <https://www.devsecops.cc>.

9 Years of ADMIN on One DVD

ADMIN

Network & Security

COMPLETE ARCHIVE DVD
ISSUES 00-49

ISSUE 50/2019

While this ADMIN magazine is not responsible for any damage to your system, we cannot be held responsible for any damage to your system or data and computer systems related to the use of this disc.

DVD

OVER
3,700
PAGES OF UP-CLOSE
SYSTEM
ADMINISTRATION

Smart and Practical Information
for Network Professionals



This searchable DVD gives you 50 issues of ADMIN, the #1 source for:

- network security
- system management
- troubleshooting
- performance tuning
- virtualization
- cloud computing

Clear off your bookshelf and complete your ADMIN library with this powerful DVD!

ORDER NOW!

shop.linuxnewmedia.com



MakerSpace

Science projects with the
Python MetPy library

Homework Helper

MetPy can help you answer your kids' science questions.

By Pete Metcalfe

Are you stumped by your kids' homework? Sometimes answering their science questions can be quite challenging. Luckily, some great resources and software packages can help parents out of a homework jam. A good example is the MetPy Python library [1], which has some awesome thermodynamic and weather functions.

In this article, I introduce the MetPy library and show how it can be used to solve questions like: Can you make snow when it's above freezing? How much thinner is the air with altitude? How do you calculate wind chill?

Getting Started

To install the MetPy Python library, enter:

```
pip install metpy
```

One of nice things about MetPy is that it manages scientific units, so a variable is defined with both its value and units. In the code

```
>>> from metpy.units import units
>>> tempToday = [40] * units.degF
>>> tempToday.to(units.degC)
<Quantity([4.44444444], 2
'degree_Celsius')>
```



Figure 1: A snow gun makes snow.

Lead Image © Pavel Romanchenko, 123rf.com

Listing 1: Wet-Bulb Temp

```
>>> import metpy.calc
>>>
>>> pressure = [101] * units.kPa
>>> temperature = [0.5] * units.degC
>>> dewpoint = [-2.5] * units.degC
>>>
>>> metpy.calc.wet_bulb_temperature(pressure, temperature, dewpoint)
<Quantity(-0.6491265444587286, 'degree_Celsius')>
```

the `units` module makes a simple temperature conversion. In this example, a temperature of 40°F is converted to 4.4°C with the `to()` method.

You can also do some interesting mixing of units in math calculations. For example, you can add 6 feet and 4 meters:

```
>>> print( [6] * units.feet +
           [4] * units.m)
[19.123359580052494] foot
```

MetPy has a very large selection of thermodynamic and weather functions, which I demonstrate in the next sections.

Can You Make Snow When It's Above Freezing?

Ski resorts can make snow by forcing water and pressurized air through a “snow gun” (Figure 1). Snowmaking [2] can be an expensive operation, but it allows ski resorts to extend their season.

Listing 2: When to Make Snow

```
01 #
02 # Find when you can make snow
03 #
04 import matplotlib.pyplot as plt
05 import metpy.calc
06 from metpy.units import units
07
08
09 print("Get temps vs. humidity")
10 print("-----")
11
12 # Wet bulb temp of -2.5C is needed for snow making
13 wet_temp = [-2.5] * units.degC
14 pres = [1] * units.atm # assume a pressure of 1 atm
15
16 plt_temp = []
17 plt_hum = []
18 for temp in range (-10,11): # Check dry temperatures
    between -10 - 10 C
19     dry_temp = [temp] * units.degC
20     # Get the relative humidity
21     rel_humid = metpy.calc.relative_humidity_wet_
    psychrometric(dry_temp, wet_temp,pres)
22     # Strip the humidity units for charting, and make a
    percent (0-100)
23     the_humid = rel_humid.to_tuple()[0] * 100
24     if (the_humid <= 100) and (the_humid > 0) : # Get
    valid points
25         plt_temp.append(temp) # append a valid temp
26         plt_hum.append(the_humid) # append a valid
    humidity
27     print (temp, the_humid)
28
29 fig, ax = plt.subplots()
30 ax.plot(plt_temp, plt_hum, color="blue" )
31 ax.set(xlabel='Temperature (C)', ylabel='Humidity (%)',
    title='When you can make Snow')
32 ax.grid()
33 fig.savefig("makesnow.png")
34 plt.show()
```

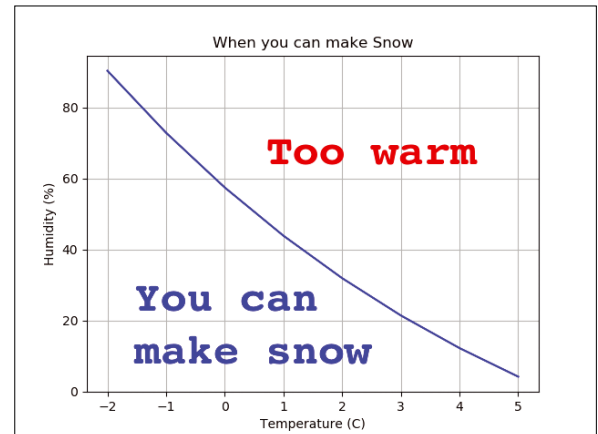


Figure 2: When you can make snow.

When you get a weather forecast, the temperature is given as the ambient, or dry-bulb,

temperature. The wet-bulb temperature takes the dry air temperature and relative humidity into account. The wet-bulb temperature is always less than the outside temperature, so snowmaking can take place if the wet-bulb temperature is below -2.5°C or 27.5°F.

MetPy has a number of functions that can find the humidity and wet-bulb temperature (`wet_bulb_temperature()`), which just needs the pressure, dry temperature, and dew point.

In Listing 1, the temperature is below freezing, but it's not possible to make snow, because the wet-bulb temperature is only -0.6°C (not the required -2.5°C).

Knowing that -2.5°C (27.5°F) is the wet-bulb temperature upper limit for snowmaking, the `relative_humidity_wet_psychrometric` function can be used to

create a curve of humidity and dry temperature points that shows where it is possible to make snow (Figure 2).

Listing 2 shows how to use the MetPy library to find when snow can be made. The code iterates between the temperatures of -10°C to 10°C (line 18), getting the relative humidity with the `relative_humidity_wet_psychrometric` call (line 21). The relative humidity is multiplied by 100 to get a percentage and is made unitless by the `to_tuple()` [0] method (line 23). The results are plotted by Matplotlib library functions (lines 29-34).

From the data, you can see that if the humidity is low, it is possible to make snow, even when the temperature is above freezing (+4°C is the typical cutoff temperature).

Listing 3: Air Pressure by Altitude

```
>>> import metpy.calc
>>>
>>> New_York_alt = [33]*units.ft
>>> metpy.calc.height_to_pressure_std(New_York_alt).to(units.atm)
<Quantity([0.99880745], 'standard_atmosphere')>
>>> Denver_alt = [5280] * units.ft
>>> metpy.calc.height_to_pressure_std(Denver_alt).to(units.atm)
<Quantity([0.82328412], 'standard_atmosphere')>
>>> Mexico_city_alt = [7350]*units.ft
>>> metpy.calc.height_to_pressure_std(Mexico_city_alt).to(units.atm)
<Quantity([0.76132418], 'standard_atmosphere')>
```

How Much Thinner is the Air with Altitude?

Everyone knows the air is thinner when you're in an airplane, but how much thinner is it in Denver or Mexico City compared with New York City?

Again, MetPy can help. The `height_to_pressure_std` function calculates a pressure value by altitude. The `to()` method converts the pressure to standard atmospheres (Listing 3).

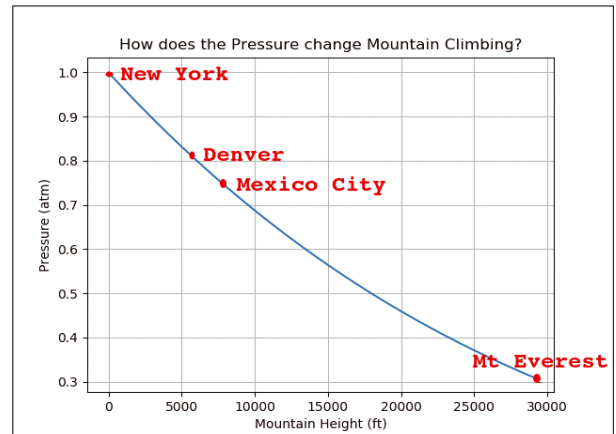


Figure 3: How the atmosphere changes when climbing a mountain.

The results show that the air pressure in Denver is 82 percent that of New York's, or 18 percent thinner, and Mexico City's atmosphere is 76 percent that of New York's, or 24 percent thinner.

Listing 4: Air Thinning with Altitude

```
01 #
02 # heights.py - How much does the air thin as you climb?
03 #
04 import matplotlib.pyplot as plt
05 import metpy.calc
06 from metpy.units import units
07
08
09 print("Get height vs. Atm Pressure")
10 print("-----")
11
12 # create some plot variables
13 plt_ht = []
14 plt_press = []
15
16 # Check Atmospheric Pressure from sea level to Mt. Everest (29,000 ft) heights
17 for temp in range(0,30000,1000): # Check dry temperatures between -10 - 10 C
18     height = [temp] * units.feet
19     pressure = metpy.calc.height_to_pressure_std(height)
20     atm = pressure.to(units.atm)
21     print (height, atm)
22     plt_ht.append (height.to_tuple()[0]) # put the value into plt list
23     plt_press.append (atm.to_tuple()[0]) # put the value into plt list
24
25 fig, ax = plt.subplots()
26 ax.plot(plt_ht, plt_press )
27 ax.set(xlabel='Mountain Height (ft)', ylabel='Pressure (atm)', title='How does the Pressure change Mountain Climbing?')
28 ax.grid()
29 fig.savefig("height_vs_press.png")
30 plt.show()
```


Listing 5: A -20°C Wind Chill Curve

```

01 #
02 # windchill.py - Create a -20C Wind Chill Curve
03 #
04 import matplotlib.pyplot as plt
05 import metpy.calc
06 from metpy.units import units
07
08 # Create some plotting variables
09 plt_temp = []
10 plt_speed = []
11
12 for temp in range (0,-46,-1): # Check dry temperatures
                                between -10 - 10 C
13     the_temp = [temp] * units.degC
14     for wind in range(1,61,1):
15         the_wind = [wind] * units.kph
16         windchill = metpy.calc.windchill(the_temp, the_
                                wind, face_level_winds=True)
17         # Select points with a wind chill around -20
18         if (windchill.to_tuple()[0]) <= -19.9 and
                                (windchill.to_tuple()[0] >= -20.1) :
19             plt_temp.append(temp)
20             plt_speed.append(wind)
21
22 fig, ax = plt.subplots()
23 ax.fill_between(plt_temp, plt_speed, label="-20 C - Wind
                                Chill", color="red" )
24
25 ax.set(xlabel='Temperature (C)', ylabel='Wind Speed
                                (kph)', title='When is the Wind Chill -20C ?')
26 ax.grid()
27 fig.savefig("windchill.png")
28 plt.show()

```

Listing 4 uses the `height_to_pressure_std` function to create a chart (Figure 3) of atmospheric pressure between sea level and the top of Mt. Everest (29,000ft). At the top of Mt. Everest, the air is 70 percent thinner than at sea level!

How Do You Calculate Wind Chill?

The MetPy `windchill` function returns a “feels like” temperature according to wind speed and ambient temperature. For example, an outside ambient temperature of 40°F with a wind of 20mph feels like 28°F:

```

>>> import metpy.calc
>>>
>>> temp = [40] * units.degF
>>> wind = [20] * units.mph
>>> metpy.calc.windchill(temp,
wind, face_level_winds=True)
<Quantity([28.42928573],
'degree_Fahrenheit')>

```

When the temperature starts going below -20°C, parents need to keep a close eye on their kids for frostbite.

Listing 5 calculates a wind chill curve (Figure 4) of -20°C by ambient temperature and wind. The code iterates between temperatures of 0°C and -45°C (line 12) and wind speeds of 1 to 60kph (line 14) and finds wind chills of -20°C (line 18).

From the results, you can see that even if you have a nice ski day of -8°C, a high wind of 40kph will make the temperature feel like -20°C, a 12° drop.

Summary

MetPy didn't help me answer all of my kids' science questions, but it really helped with water and weather problems. If you have a budding meteorologist or chemical engineer in your house, take a look at the Python MetPy library. ■■■

Info

- [1] MetPy: <https://unidata.github.io/MetPy/latest/index.html>
- [2] Snowmaking: <https://en.wikipedia.org/wiki/Snowmaking>

Author

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

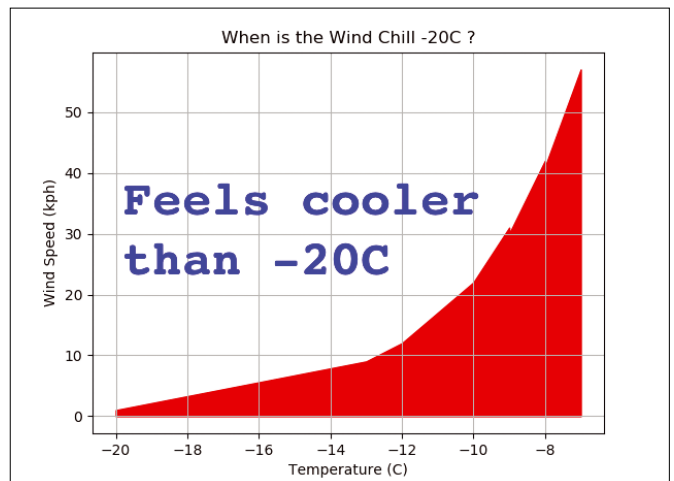


Figure 4: Wind chill at -20°C.



MakerSpace

BerryLan simplifies headless
Raspberry Pi install

No-Head Start

BerryLan installs a system on a Raspberry Pi that can be integrated into the wireless network with a smartphone app over Bluetooth. *By Christoph Langner*

The Raspbian operating system can get the Raspberry Pi up and running – with or without a desktop environment. Starting with the standard installation, the system can be adapted to your own needs with very little effort. However, this requires a connected keyboard, mouse, and monitor. Even SSH access is a little more complicated because Raspbian has not started the SSH server automatically for some time now [1].

Building a headless Raspberry Pi (i.e., a computer without peripherals or even a network cable) requires more effort and a certain amount of know-how. Before the Raspbian system starts for the first time, the `/etc/wpa_supplicant` configuration file has to be edited and the network access credentials entered.

The BerryLan project seeks to handle this work for you wirelessly over Bluetooth

with a smartphone app. With the customized Raspbian image and a smartphone, the Raspberry Pi system will be running and on the network within a few minutes.

Jump Start

For BerryLan [2], developers resorted to the proven Raspbian distribution but have limited themselves to the Lite version (see the “Raspbian Lite” box). In addition to the basic system configuration, only the Nymea network manager [3] is added, which lets you set up the network for the Raspberry Pi on a smartphone over a Bluetooth LE connection without manual interaction. The open source project offers the apps in the respective app stores.

Installing BerryLan is similar to installing a normal Raspbian system: You download the current version of BerryLan from

Raspbian Lite

The Raspberry Pi Foundation maintains a branch of its operating system that installs a lightweight system without a graphical user interface, thus conserving the limited resources of the Raspberry Pi. The Lite version is therefore recommended for users who want to run the Raspberry Pi in headless mode (i.e., without input and output devices). If necessary, however, the system can be upgraded to the familiar user interface by importing some application packages with the package manager:

```
$ sudo apt install raspberrypi-ui-mods rpi-chromium-mods lightdm
```

To load the graphical desktop environment, call `sudo raspi-config` and select the **3 Boot Options / B1 Desktop/CLI / B4 Desktop Autologin** option.



Figure 1: The Raspberry Pi booted with BerryLan shows up in the BerryLan app as *BT WLAN setup*.

the project homepage, extract the ZIP archive (`raspbian-stretch-berry-lan-lite-latest.zip`), and write the resulting image file to the desired memory card. As usual, you can do this with the familiar command-line tools or with cross-operating system graphical tools such as Etcher [4].

If you own a Raspberry Pi that already runs Raspbian, you don't necessarily have to set up the system again with BerryLan and adapt the system to your requirements. In the project FAQ [5] you will find instructions on how to extend an existing Raspbian system with BerryLan.

Configuration

After writing the SD memory card, slot it into the Raspberry Pi and boot the system. For configuration, install the BerryLan app suitable for your smartphone. The developers maintain variants for Android [6] and iOS [7]; for this article, I used the Android version of the app.



Figure 2: The Raspberry Pi scans the environment for available wireless networks and transfers the list to the BerryLan app on the phone.

The app immediately tries to detect the Raspberry Pi over Bluetooth, so Bluetooth must be enabled on the mobile phone. On the other side, you have to allow BerryLan on Android to retrieve the device location. Don't let this confuse you: This is not about GPS tracking, but about Bluetooth routines that can also be used for locating purposes [5].

The Raspberry Pi you booted with BerryLan shows up in the app as *BT WLAN setup* (Figure 1). However, other pairing-capable Bluetooth devices also appear in the list. When you tap on the Raspberry Pi in the dialog, BerryLan transfers all wireless networks in the immediate vicinity detected by the Raspberry Pi to the app (Figure 2). In the next step, select the desired network and enter the appropriate access credentials with the smartphone's keyboard. Once the connection has been established, the app displays the IP address of the freshly installed Raspbian system (Figure 3).

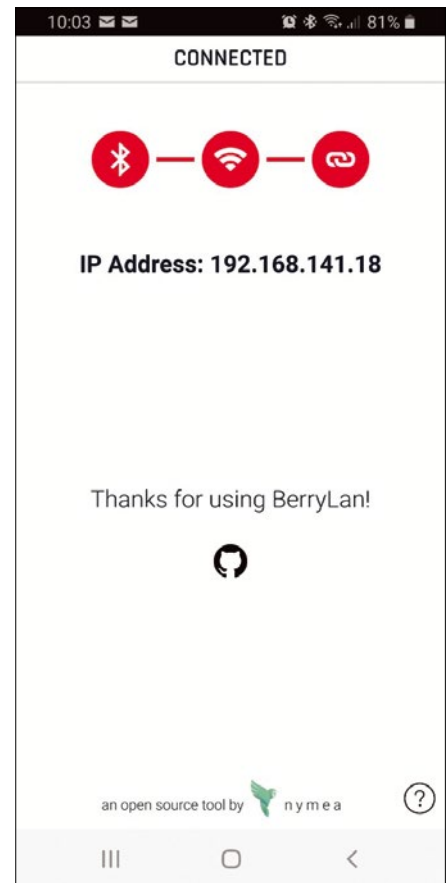


Figure 3: After establishing the connection, the app displays the Raspberry Pi's IP address on the smartphone.

Now that you have the IP address, you can log on to the system under Linux by typing

```
ssh pi@<RaspPi-IP>
```

(Figure 4). The password is the usual *raspberry*. Alternatively, use an SSH client for Android such as JuiceSSH [8] or the Termux [9] Linux environment adapted for Android.

Now you will want to change the password without delay. To do this, either call `sudo raspi-config` and select the *1 Change User Password* option or run the `passwd` command on the Raspbian system's terminal and follow the prompts.

Conclusions

BerryLan proves to be a useful helper after the install. If you want to use the Raspberry Pi on another wireless network, you simply boot the Pi in the usual way and launch the BerryLan app again. As in the initial installation, the smartphone should then detect the

```

pi@raspberrypi: ~
[16:24:07 toff ~]$ ssh pi@192.168.188.78
The authenticity of host '192.168.188.78 (192.168.188.78)' can't be established.
ECDSA key fingerprint is SHA256:NLMUHfKT+bq3kxg88wF5JmwfYAovFUUXCPbqp2ImHI.
Are you sure you want to continue connecting (yes/no/[fingerprint])? yes
Warning: Permanently added '192.168.188.78' (ECDSA) to the list of known hosts.
pi@192.168.188.78's password:
Linux raspberrypi 4.14.79+ #1159 Sun Nov 4 17:28:08 GMT 2018 armv6l

The programs included with the Debian GNU/Linux system are free software;
the exact distribution terms for each program are described in the
individual files in /usr/share/doc/*/copyright.

Debian GNU/Linux comes with ABSOLUTELY NO WARRANTY, to the extent
permitted by applicable law.

SSH is enabled and the default password for the 'pi' user has not been changed.
This is a security risk - please login as the 'pi' user and type 'passwd' to set
a new password.

pi@raspberrypi:~ $
    
```

Figure 4: In contrast to Raspbian, the SSH server is enabled out of the box on BerryLan. For security reasons, you will want to change the password immediately.

BerryLan Raspberry Pi over Bluetooth, and you can configure the wireless network in the app. The BerryLan developers have hardly changed the Raspbian system itself; they

have simply added their own package source, from which the Bluetooth-enabled add-on for the network manager is installed. The operating system can be updated in the usual way, because all

the updates and post-install packages come from the official Raspbian package sources. BerryLan is therefore ideally suited as a basis for users who want to use a Raspberry Pi on different wireless networks in a number of scenarios. ■■■

Info

- [1] "Enable SSH" by Christoph Langner, *Raspberry Pi Geek*, issue 23, 2017, <https://www.raspberrypi-geek.com/Archive/2017/23/Future-proofing-the-Raspbian-SSH-Server/>
- [2] BerryLan: <http://www.berrylan.org>
- [3] Nymea network manager: <https://github.com/nymea/nymea-networkmanager>
- [4] Etcher: <https://www.balena.io/etcher>
- [5] BerryLan FAQ: <http://www.berrylan.org/faq.html>
- [6] Android app: <https://play.google.com/store/apps/details?id=io.guh.berrylan>
- [7] iOS app: <https://apps.apple.com/us/app/berrylan/id1436156018>
- [8] JuiceSSH: <https://juicessh.com>
- [9] Termux: <https://termux.com>

Shop the Shop → shop.linuxnewmedia.com

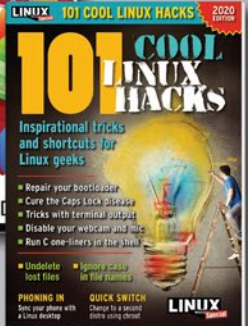
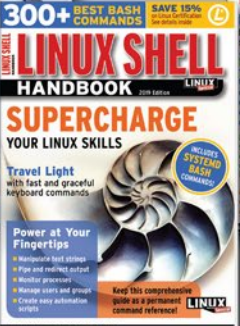
Discover the past and invest in a new year of IT solutions at Linux New Media's online store.

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

➤ shop.linuxnewmedia.com

DIGITAL & PRINT SUBSCRIPTIONS

SPECIAL EDITIONS



Millions of users around the world spend their workday online, and, if you're hiding out from the recent pandemic, you're probably depending on your personal computer even *more* than before. If you manage appointments and events with an online calendar, and you don't like giving up your privacy to Google or another commercial service, you can implement your own calendar service using the Free CalDAV standard. This month we show how to set up a CalDAV server and access it from a remote computer or personal device. Also, in this month's LinuxVoice, synchronize your notes with Simplenote, and add physics to a home-built computer game with the Lua-based LÖVE framework.



Image © Olexandr Moroz, 123RF.com

LINUXVOICE

Doghouse – United We Stand 70

Jon "maddog" Hall

It's time to innovate – in the midst of the disruptions caused by the coronavirus, let's find ways to keep everyone communicating, working, and learning.

Simplenote and sncli 72

Dmitri Popov

If you're using Simplenote, check out sncli, a Python-based tool for syncing and managing your notes.

Tutorial – CalDAV/CardDAV 75

Marco Fioretti

You can manage your calendars and address books with the CalDAV/CardDAV standards, Nextcloud, and a few open source tools.

FOSSPicks 84

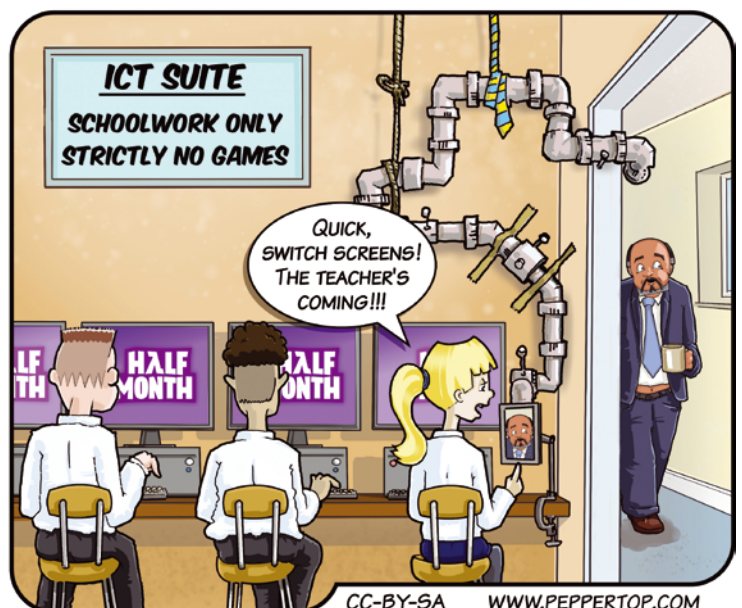
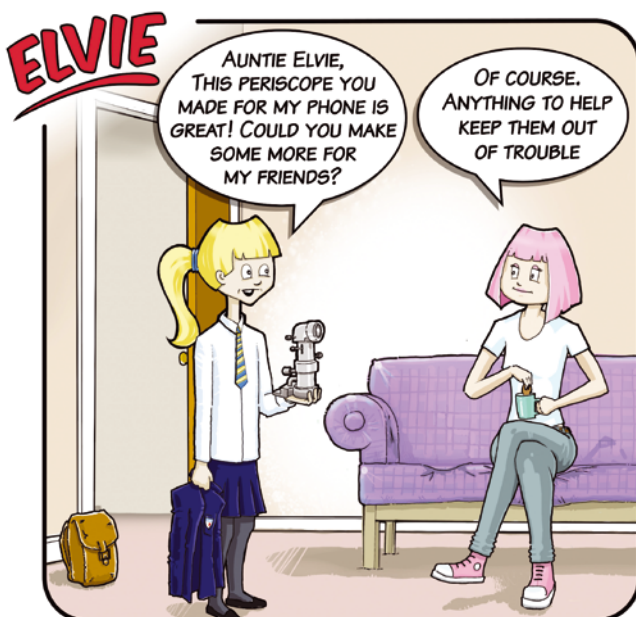
Graham Morrison

This month Graham looks at navi, SDRangel, Photoflare, Ikona, emulator-sun-2, Quantum Tetris, Selfless Heroes, and more!

Tutorial – LÖVE Physics 90

Paul Brown

Video game animation is not simply a matter of making your characters move – you also have to consider the physics.



MADDOG'S DOGHOUSE

It's time to innovate – in the midst of the disruptions caused by the coronavirus, let's find ways to help keep everyone communicating, working, and learning. BY JON "MADDOG" HALL



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

United We Stand

I write this on March 25, 2020, and reflect on what a month can mean.

A month ago, the people around me knew about the impact of the coronavirus on China, but were in denial about its effects on the United States.

I was scheduled to attend three conferences in Brazil, one of which was very large and would attract thousands of people for a week.

On February 27, I wrote a detailed email to the organizers of the largest conference and pointed out the issues of the coronavirus within their event.

There was one case of coronavirus in Brazil at that time.

On March 2, they answered and said they were "looking at it." There were two cases of coronavirus. On March 12, they did the right thing and postponed it, three days before I was supposed to fly to Brazil.

There were hundreds of cases of the coronavirus in Brazil.

I realized that by the time the other two conferences (being held at universities) were going to occur the universities would likely be closed and the conferences canceled. I joined the tens of thousands of other airline passengers and immediately started canceling my airline tickets and hotel reservations.

Right after that the other two conferences were postponed.

Since that time, stock markets all over the world are collapsing, and more than three million people registered for unemployment in one week in the USA alone. Some people estimate that 30 percent of the workforce in the USA could be out of work.

This is war.

We, the users of free and open source software and hardware, need to answer this call to war. We need to think about how to get people back to work in a way that keeps people healthy. We need to help our children get the education they need even when they can not meet in the traditional brick and mortar school. We need to innovate.

Companies that we work for may stay in business, and we can go back to work for them as the virus recedes or scientists find a vaccine for it. On the other hand, when we do go back to work perhaps some of us can tailor our work so it is better than the work we left behind.

What can we do now? Many people are unemployed or on reduced hours. Some of them have children at home, so they need to spend time with the children. Perhaps this is the time to innovate and learn new skills ourselves and to teach our children new skills.

What would you like to learn so when this crisis passes you can apply that new knowledge to your work and life?

There are many free sources of training and knowledge online. There are free textbooks and free training videos. Search them out and learn those skills.

There are free novels online to help entertain yourself and your children.

On the other hand, there are many of you that know some skill or information. Think about mentoring another person who wants to learn. This will help at least two people, the person who wants to learn and you, the teacher, since teaching is one way of learning the information really well.

I have often told people that I had a hard time passing my university courses in compiler design, and I hardly understood how compilers worked. I passed the course, but barely. But the first time I had to teach the course is when I learned it completely, because you are not allowed to think "the professor will never ask that on a test." You know your students will ask a question about the topic that you know the least. So you learn it ... cold. I have not taught compiler design for over 30 years, but I could walk up to a white board today and teach it.

Many people think "I do not know anything that other people want to know," but you may be surprised at what you know that others do not.

I would ask high school students interested in GNU/Linux what they know about computers, and they would say "nothing really." Then I would ask who fixes the computers in their household or in their neighborhood. Who fixes the bad disk? Who installs the new version of the operating system? Who sets up the wireless network? These are simple things to these students, but impossible for their parents and grandparents.

Finally, think about jobs that you can do solo or with a small group of people. This may be the time that you start your own business or get together and form a cooperative with other people who may be out of work.

There are many "freelancers" in the world or "consultants" who work alone or remotely. You might reinvent yourself to be one of them.

Carpe Diem. Carpe Diem! ■■■

Available Now

* 2018 EDITION *

Linux Magazine Archive DVD

Searchable
Linux Archive:

19,000

Pages of Practical
Know-How

214 issues of Linux Magazine
on one handy DVD!

ORDER NOW!

shop.linuxnewmedia.com



Manage Simplenote with sncli

Duly Noted

If you're using Simplenote, check out sncli, a Python-based tool for syncing and managing your notes.

BY DMITRI POPOV

Despite the proliferation of Markdown-based and regular note-taking tools and services, Simplenote remains a popular choice for note-keeping. There are many reasons for its enduring success. Although the service is continuously being improved and tweaked, it manages to maintain a delicate balance of functionality and simplicity. In addition to essential features like tagging, Markdown support, and search, Simplenote also makes it possible to collaborate on notes, as well as to publish them on the web. And creature comforts like word count and versioning make Simplenote an ideal writing tool. In short, you'll be hard pressed to find a service or software that offers a similar level of functionality free of charge.

Although Simplenote offers desktop applications, they are basically web apps disguised as desktop tools. Fortunately, the service provides an API, so there are several third-party solutions for integrating Simplenote into your preferred editor. For example, if you happen to use Emacs, you'll be pleased to learn that there is a plugin that lets you work with your Simplenote notes without leaving the convenience of your favorite text editor.

Figure 1: You can preview notes directly in the terminal or open them in your preferred text editor.

```
File Edit View Bookmarks Settings Help
Title: # digiKam ninja tricks [3ff9359f-b007-440a-942f-ace4302817f5] 1/161
Date: Sun, 08 Mar 2020 13:06:32 [presentations] [v4] [ m]
# digiKam ninja tricks

It seems that digiKam is a somewhat underappreciated application. My unscientific research shows that many Linux users consider it a decent tool for managing photo collections, but they are not aware of digiKam's more advanced features and the fact that this application offers functionality that covers the entire photographic workflow. Today, I'd like to do something about it.

## Deploy AppImage

Advantages:
- No installation required
- Everything in one file

Drawbacks:
- Slower launch
- No system integration
- File system can't be modified (not possible to replace the default splash screen)

## Workflow example

I shoot both raw and JPEG, and I usually do all adjustments on the JPEG files. Sometimes I process raw files manually for better results, but usually, I simply store raw files as source files.

I use digiKam's Import module to transfer files from a storage card to the production machine. The Import setup is configured to rename incoming files on the fly using date and time data from EXIF metadata using the *yyyyMMdd-hhmmss* format. While digiKam makes it easy to see the date and time of each photo, having these data in the file name helps to manage photos outside the photo management application.
```

But if you are looking for a standalone client that allows you to sync Simplenote notes as well as manage and edit them using any text editor, sncli [1] might be right up your alley. This handy Python-based tool allows you to work with Simplenote notes online and offline, providing synchronization functionality and a wide range of useful features (Figure 1).

Getting Started

Sncli is written in Python, and the easiest way to install it on a mainstream Linux distribution is to use the pip tool. Before you proceed, make sure that this tool is installed on your system. Run the `which pip3` command, and you should see the path to the `pip3` executable. If the output of the command is blank, you need to install the `pip3` package using the default package manager of your Linux distribution. On openSUSE, you can do this using the command:

```
sudo zypper in python3-pip
```

With pip installed on your system, run the command:

```
sudo pip3 install sncli
```

Before you launch sncli, you need to create a `.sncliirc` configuration file (Listing 1). Sncli supports a wide range of configurable options that you can specify in the configuration file, but as a minimum you need to specify your Simplenote credentials to give sncli access to your account. To do this create a `.sncliirc` file in your home directory and paste the lines from Listing 1 into it.

Listing 1: .sncliirc

```
[sncli]
cfg_sn_username = USER
cfg_sn_password = PASSWORD
```


Replace `USER` and `PASSWORD` in Listing 1 with your actual Simplenote username and password. Save the changes, open the terminal, and run the `sncli` command.

During the first start, `sncli` performs a full synchronization. And depending on the number of notes in Simplenote, it may take a moment. Once the sync is finished, you should see `sncli`'s main interface with a list of notes.

Features

Despite its simplicity, the text-based interface packs in a lot of useful information (Figure 2). To begin with, all notes are color-coded, which makes it easier to quickly view their status. For example, the green color marks the notes that have been updated in the last week, and brown is used for notes that have been updated in the last month. The notes that have not been updated in a year are light blue. The interface itself is split into four columns. The first column from the left displays the modification date of each note. The next column shows the note's current status and the note's flags. For example, you'll see the `*` flag next to a pinned note, and all Markdown-formatted notes are marked with the `m` flag. When you trash a note, the `T` flag is added to it until `sncli` synchronizes the changes. The not yet synced notes have the `X` flag next to them, while the published (or shared) notes are marked with the `S` flag. Finally, the middle column lists a list of notes, and the last column lists the tags assigned to notes.

Since `sncli` has a text-based interface, all operations are performed using the keyboard. The default key bindings in `sncli` mimic those in the `vi` editor. So you can use the `j` and `k` keys to go up and down the list of notes. To scroll one page up, use the `b` key, and to scroll one page down press `Ctrl +f`. The `g` lets you jump to the top, and pressing `G` jumps to the bottom. To quickly preview the currently selected note, press `Enter`, while hitting `Space` will open the note in the default editor (i.e., the editor defined by the `$EDITOR` environmental variable). Remember that `sncli` is easy to customize (Figure 3). If you prefer to use a different editor for editing notes, add the following line to the `.sncliirc` file (replace `emacs` with the desired editor installed on your system):

```
cfg_editor = emacs {fname} +{line}
```

```

File Edit View Bookmarks Settings Help
Simplenote (online) 11/33
[2020/03/08] * # Photo notes travel,photography
[2020/03/08] * # DocBook snippets work
[2020/03/08] *Sm # Git commands git
[2020/03/08] * # German phrases language
[2020/03/08] * # Danske ord og vendinger language
[2020/03/08] * # Reading list 2020 reading
[2020/03/07] * # Linux Commands linux
[2020/03/04] * # English phrases language
[2020/03/08] #0: Simplenote with sncli lxm,writing
[2020/03/08] # # Lilut: Hald CLUT web UI linux-photography
[2020/03/08] # # digiKam ninja tricks presentations
[2020/03/08] # # From idea to article presentations
[2020/03/08] # # openSUSE workstation configuration work
[2020/03/08] # # Use digiKam with an external storage device digiKam-recipes
[2020/03/08] # # Disaster-proof digiKam setup digiKam-recipes
[2020/03/08] # # Use G'MIC-Qt filters digiKam-recipes
[2020/03/08] # # Group RAW and JPEG files with a script digiKam-recipes
[2020/03/08] # # Introducing micro.sth linux-photography
[2020/02/03] # # Towards an efficient photographic workflow on Linux linux-photography
[2020/01/31] # # Tokyo Taxi Lights tokyo-taxi-lights
[2020/01/07] # # 128x64 OLED wiring little-backup-box
[2020/01/03] # # Reading list 2019 reading
[2019/12/04] # # Shutter count shell script
[2019/09/22] # # Sony ILCE-6X00 notes photography
[2019/09/22] # # Photography notes photography
[2019/09/22] Tokyo 2016-05 tokyo,travel
[2019/09/22] Pimsleur Conversational Japanese language
[2019/09/22] # # Nakagin Capsule Tower architecture
[2019/09/22] # # La Grande Motte architecture,lagrandemotte
[2018/06/03] # # Osaka, Japan 2018-06-11 travel

```

Figure 2: `Sncli`'s main screen provides an informative overview of all notes. In the color scheme used above, the newest notes are in red.

There are a few other keys that are worth memorizing right from the start: `C` to create a new note, `t` to add and edit tags, `p` to pin the current note, and `m` to enable Markdown formatting for the current note. Finally, `h` displays a quick overview of all key bindings in `sncli` (Figure 4).

If the default key scheme is not your cup of tea, you can easily reconfigure it by specifying the appropriate key parameter in the `.sncliirc` file. For example, if you prefer to use the `e` key instead of `Space` to open a note for editing, add the line below to the configuration file:

```
kb_edit_note = e
```

The help screen is evoked with `h` and shows parameters for all supported key bindings (e.g., `kb_down` and `kb_up` for up and down keys), and you can use them to reconfigure the default key scheme.

Figure 3: You can customize `sncli`'s configuration and behavior by specifying parameters in the `.sncliirc` file.

```

File Edit View Projects Bookmarks Sessions Tools Settings Help
Documents .sncliirc
1 [sncli]
2 cfg_sn_username = dmpop@tuta.io
3 cfg_sn_password = cq37ndhdv
4
5 # as an alternate to cfg_sn_password you could use the following config item
6 # any shell command can be used; its stdout is used for the password
7 # trailing newlines are stripped for ease of use
8 # note: if both password config are given, cfg_sn_password will be used
9 cfg_sn_password_eval = gpg --quiet --for-your-eyes-only --no-tty --decrypt ~/sncli-pass.gpg
10
11 # see http://urwid.org/manual/userinput.html for examples of more key combinations
12 kb_edit_note = space
13 kb_page_down = ctrl f
14
15 # note that values must not be quoted
16 clr_note_focus_bg = light blue|
17
18 # if this editor config value is not provided, the $EDITOR env var will be used instead
19 # warning: if neither $EDITOR or cfg_editor is set, it will be impossible to edit notes
20 # cfg_editor = emacs
21
22 # alternatively, {fname} and/or {line} are substituted with the filename and
23 # current line number in sncli's pager.
24 # If {fname} isn't supplied, the filename is simply appended.
25 # examples:
26 # cfg_editor = nvim {fname} +{line}
27 cfg_editor = emacs {fname} +{line}
28
29 # this is also supported for the pager:
30 cfg_pager = less -c +{line} -N {fname}

```

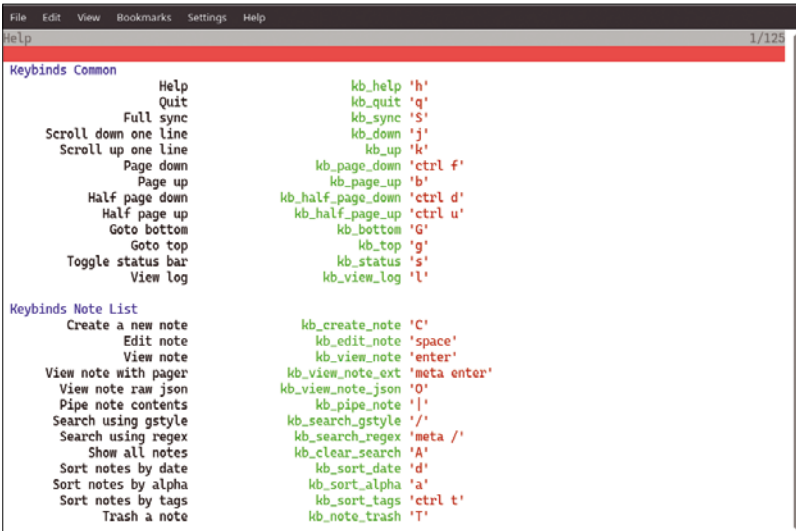


Figure 4: The help screen in sncli is only one key press away.

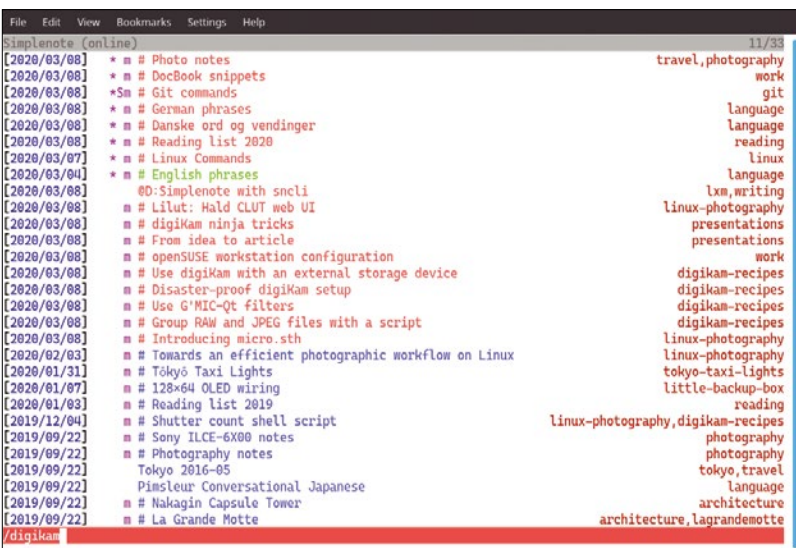
Speaking of configuration, there is one more option you might want to specify while you are at it. When you open and edit a note, it's not updated in sncli until you close the editor. Until then, the editor uses a temporary file in the default temporary directory (e.g., `tmp`). This means that if your editor or the entire system crashes, all unsaved changes will be gone. To prevent this from happening, use the `cfg_tmpdir` option to specify a persistent directory for temporary notes (replace `USER` with your actual username in the example below):

```
cfg_tmpdir = /home/USER/.sncli/tmp/
```

If the directory doesn't exist, you have to create it manually.

Sncli saves synchronized notes as files in the JSON format in the `~/sncli` directory, and you can use the `sncli dump` command to save them all in a single Markdown file:

Figure 5: Sncli features powerful search functionality.



```
sncli dump >> allnotes.md
```

If you have a large number of notes in Simplenote, you'll be pleased to learn that sncli provides search functionality (Figure 5). Hit `/`, enter the search string, and press Enter to see the list of matching notes. You can also enter multiple words in the search field, and sncli will search for all the specified words. If you need to find a specific phrase, wrap it in quotes (e.g., "Japanese history"). By default, sncli searches titles, the content of each note, and tags. However, you can limit the search to tags only using the `tag:` prefix, for example: `tag:travel`. Keep in mind that regular searches are not case-sensitive. Sncli also supports searches using regular expressions.

Being a command-line tool, sncli can be used for automating Simplenote-related actions. For example, using the `sncli create` command, you can create notes directly from the command line. The following command pipes the output of the `echo` command to sncli, which creates a note with the received output as its title:

```
echo 'New note' | sncli create -
```

It's also possible to pipe notes in the JSON format to the `sncli import` command:

```
echo '{"tags":["travel","tokyo"],"content":"Tokyo travel notes"}' | sncli import -
```

Conclusions

Sncli provides a clever solution to the problem of missing synchronization functionality in Simplenote. This clever command-line tool is not only easy to deploy and use, it also offers a wide range of useful features: from the main screen that offers an overview of the notes and their statuses to flexible search capabilities and extensive customization. If you use Simplenote as your preferred note-taking tool, sncli is a must-have. ■■■

Info

[1] sncli: <https://github.com/insanum/sncli>

The Author

Dmitri Popov has been writing exclusively about Linux and open source software for many years, and his articles have appeared in Danish, British, US, German, and Russian magazines and websites. You can find more about his work on his www.tokyo-made.photography website.

Using CalDAV/CardDAV to manage calendars and address books

Personal Data Manager

You can manage your calendars and address books with the CalDAV/CardDAV standards, Nextcloud, and a few open source tools. **BY MARCO FIORETTI**

If you keep a digital calendar or address book, you want your data to be stored in one central location and accessible from any device, wherever and whenever you need it. You also want ownership of your data and metadata. By using free and open source software (FOSS), you can create calendars and address books in a private cloud that allows you to synchronize and share that data, without being locked into some corporate, data-harvesting walled garden.

In this tutorial, I will explain the open standards, CalDAV and CardDAV, that make independent storing and sharing of calendar and address book data possible. Then, I will show you how to automatically import or export calendars and contacts, from any source, to a Nextcloud instance, process that data, and migrate it to another server. Finally, I will outline how to set up your own standalone calendar and address book.

CalDAV/CardDAV

CalDAV and CardDAV, the open standards that allow centralized storage and management of personal data, are both supersets of the Web Distributed Authoring and Versioning (WebDAV) system. WebDAV's specification describes how software programs can edit remote content over the Internet, using the same protocol (HTTP) that browsers use to load pages from websites. CalDAV is the WebDAV extension for calendars, and CardDAV is the extension for personal contacts.

Both CalDAV and CardDAV are client-server protocols: They let many users, each with their own interfaces, simultaneously access the same set of events or contacts stored on a common server.

At the lower level, single events or whole calendars are stored inside files with the iCalendar extension `.ics` (also `.ical` or `.icalendar`) [1]. Files that contain address books, instead, have the Virtual Contact File extension `.vcf` [2].

Both `.ics` (Listing 1) and `.vcf` (Listing 2) files use plain text formats with a relatively simple syntax that could be written manually with any text editor. The calendar event in Listing 1 shows an annual all staff meeting that must occur from February 12 to 13

(2008 in this example), as described in the `RRULE` (recurrence rule) field. The `DESCRIPTION` variable describes the meeting's purpose (a project status checkup) and the `GEO` variable gives the meeting's location. The whole event is enclosed by `BEGIN:VEVENT` and `END:VEVENT` tags, inside a `VCALENDAR` that may contain other events both before or after this event.

Besides `VEVENTS`, an `.ics` calendar may also store to-do items (`VTODDO`), journal entries (`VJOURNAL`), and time zone information (`VTIMEZONE`). Rich text in HTML format can be specified by the parameter `X-ALT-DESC`.

Both `.ics` and `.vcf` files also contain unique identifiers (UID) that allow the servers to index them. On the host computers, the files are stored in folders with a defined structure called `vdir`. Each `vdir` corresponds to a different calendar or address book. While it is possible to create, delete, or edit a `vdir`'s contents, I recommend using a proper CalDAV or CardDAV client to prevent corruption of `vdir`'s indexing.

Nextcloud

In this tutorial, I use Nextcloud [3], which is currently the most promising solution for open source, self-hosted cloud services. To install Nextcloud and understand its importance, please see my blog [4].

Under the hood, both the calendars and contacts of all of a Nextcloud instance's users are managed by an embedded CalDAV/CardDAV server.

Listing 1: An `.ics` Event

```
BEGIN:VCALENDAR
BEGIN:VEVENT
SUMMARY:All Staff Meeting
STATUS:CONFIRMED
RRULE:FREQ=YEARLY;INTERVAL=1;BYMONTH=2;BYMONTHDAY=12
DTSTART:20080212
DTEND:20080213
CATEGORIES:teamwork, project management
GEO:87.5739497;-45.7399606
DESCRIPTION:project status check-up
END:VEVENT
END:VCALENDAR
```

Listing 2: A `.vcf` Contact

```
BEGIN:VCARD
VERSION:4.0
FN:Bruce Wayne
ADR;TYPE=WORK;;;Gotham City;USA;;Planet Earth
EMAIL;TYPE=HOME:batman@example.com
TEL;TYPE="HOME,VOICE":555-5555
URL:https://batman.example.com
ORG:Billionaire
TITLE:Dark Knight
END:VCARD
```

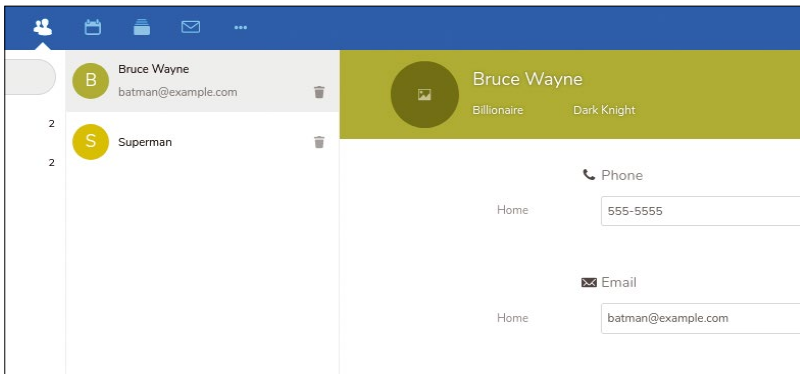
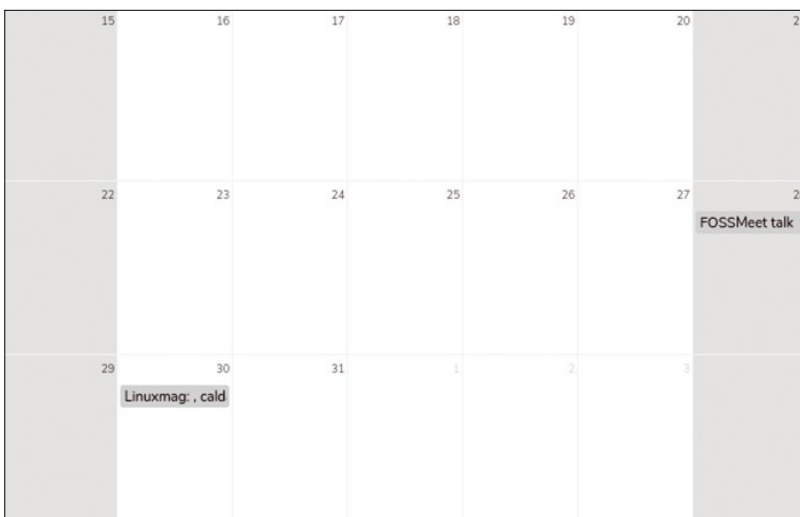


Figure 1: Nextcloud's Contacts app is the interface to its own CardDAV server.

Figures 1 and 2 show details of the address book and calendar interfaces that I created for a test Nextcloud account. In Figure 1, you can see the data that Nextcloud saved in Listing 2.

I describe the Nextcloud GUIs shown in Figures 1 and 2 in more detail in my blog [4]. However, there are at least two instances when there is a better way to process this personal data: The first is mass migration to or from another server or bulk export to some other database; the second is whenever you want to simultaneously modify contacts or calendars. Especially in the second instance, it can be much faster to make local copies of all the records, modify them with a script, and then re-upload everything to the server. The tools described in this tutorial can be used for all these tasks.

Figure 2: Part of a Nextcloud calendar in monthly mode shows events entered from its own graphical interface.



Listing 3: Installing vdirsyncer on Ubuntu

```
01 curl -s https://packagecloud.io/install/repositories/pimutils/
   vdirsyncer/script.deb.sh | sudo bash
02 sudo apt-get update
03 sudo apt-get install vdirsyncer-latest
04 export PATH="$PATH:/opt/venvs/vdirsyncer-latest/bin"
```

alternate approach that seems easier to implement with standard tools and even more flexible: Copy the server data on your computer, change as needed those copies, and then sync the `vdirs` on the server with those on your local archive. This approach also has the advantage of keeping contacts and events accessible even when the server is unreachable. I use `vdirsyncer` [5] to perform the synchronization.

In addition to installing manually, you can find `vdirsyncer` binary packages for several distributions, but not in the usual places. On my Ubuntu 19.10 desktop, for instance, I had to install from a custom repository with the sequence of commands shown in Listing 3, which I found in `vdirsyncer`'s very complete manual [6].

The command in line 1 of Listing 3 downloads and executes a shell script that fetches data for the custom repository and adds them to the Ubuntu package database. Line 3 is what actually installs `vdirsyncer` on your system. Line 4 tells Ubuntu to add the `vdirsyncer` executable's folder (`/opt/. . .`) to the list of folders (`$PATH`) where programs are stored.

Once `vdirsyncer` is installed, it needs to know which `vdirs` it should sync and their location. You do this with configuration files like `vdirsyncer-nextcloud.conf` (Listing 4).

Lines 1 and 2 of Listing 4 tell `vdirsyncer` where to store the synchronization status of all the `vdirs` it manages.

Lines 3 to 15 define a pair of address books to sync. The address book on your local computer, `my_contacts_local` (aliased as `a`), is stored in normal folders located inside the `~/ .contacts` directory and contains `.vcf` files. The Nextcloud remote address book (lines 11 to 15) is accessible with the CardDAV protocol at the URL shown on line 13, with the credentials shown in lines 14 and 15. Line 6 means that `vdirsyncer` must sync all the collections in each pair of `vdirs`.

The only difference in lines 17 to 29, visible in the variable names and file extensions, is that this section tells `vdirsyncer` to sync two sets of calendars instead of address books.

After the initial configuration, but before the actual synchronization, you need to make `vdirsyncer` "discover" all the data on the remote server (unless you configured to only sync one calendar, as explained in the documentation). Listing 5 shows how to do this plus includes an excerpt of the output.

The output means that everything went well, and `vdirsyncer` discovered everything it was supposed to find. Passing the configuration file with the `-c` option is necessary when you want to use `vdirsyncer` with different servers. The actual synchronization is done with the following single command (which you could make a regular cron job):

```
#> vdirsyncer -c vdirsyncer-nextcloud.conf sync
```

Listing 4: vdirsyncer-nextcloud.conf

```

01 [general]
02 status_path = "~/vdirsyncer/status/"
03 [pair contacts_nextcloud_to_local]
04 a = "my_contacts_local"
05 b = "my_contacts_nextcloud"
06 collections = ["from a", "from b"]
07 [storage my_contacts_local]
08 type = "filesystem"
09 path = "~/.contacts/"
10 fileext = ".vcf"
11 [storage my_contacts_nextcloud]
12 type = "carddav"
13 url = https://
    full-url-of-my-private-nextcloud-instance/"
14 username = "nextcloud_test_user"
15 password = "nextcloud_test_password"
16
17 [pair cal_nextcloud_to_local]
18 a = "my_cal_local"
19 b = "my_cal_nextcloud"
20 collections = ["from a", "from b"]
21 [storage my_cal_local]
22 type = "filesystem"
23 path = "~/.calendars/"
24 fileext = ".ics"
25 [storage my_cal_nextcloud]
26 type = "caldav"
27 url = https://
    full-url-of-my-private-nextcloud-instance/"
28 username = "nextcloud_test_user"
29 password = "nextcloud_test_password"

```

Vdirsyncer has many other options and uses. Due to space constraints, I will only discuss how to handle conflicts. If two calendars contain two different records for the same event, you must decide which calendar is the master that should win such conflicts and explicitly tell vdirsyncer. (Listing 5), if the Nextcloud calendar is the master, you should add a line like this:

```
conflict_resolution = "b wins"
```

right after the lines 6 and 20 of Listing 5. In addition to the vdirsyncer manual [6], see [7] for more conflict resolution examples and vdirsyncer configuration tips.

khal and khard

In addition to vdirsyncer, two other tools, khard [8] for address books and khal [9] for calendars, offer a simple solution for this tutorial. Khard is the most essential. Khal can be used interactively: Figure 3 displays the same calendar shown in Figure 2 read from the local copy previously created by vdirsyncer. Both khard and khal can be installed from the standard repositories of major Linux distributions.

Khal's configuration can be as simple as Listing 6: The `calendars` section lists all the desired

Listing 5: Discovering Data on the Remote Server

```

01 #> vdirsyncer -c vdirsyncer-nextcloud.conf discover
02 Discovering collections for pair contacts_nextcloud_to_local
03 my_contacts_local:
04   - "contacts"
05 my_contacts_nextcloud:
06   - "contacts" ("Contacts")
07 Saved for contacts_nextcloud_to_local: collections = ["contacts"]
08 Discovering collections for pair cal_nextcloud_to_local
09 my_cal_local:
10 my_cal_nextcloud:
11   - "work" ("Work")
12 .....
13 warning: No collection "work" found for storage my_cal_local.
14 Should vdirsyncer attempt to create it? [y/N]: y
15 Saved for cal_nextcloud_to_local: collections = ["work"]

```

calendars, while the `default` section sets the options, most with self-explanatory names, that are common to all calendars: default calendar, number of days to display (`timedelta`), highlighting, and date formatting.

Khard's configuration (Listing 7) can be almost as simple, if you only use it for importing bulk contacts or as an address grabber for console email clients like Mutt (more on this later).

The `editor` and `merge_editor` options need some additional explanation. If these options do not point to existing programs, khard will not start. In Listing 7, the `merge_editor` used to handle conflicts is the one included in the khard package.

If you launch khard without options, it will just list all the contacts it found. In my case, the contacts shown in Listing 8 are the same ones that I had created in Nextcloud.

You can add new contacts on the command line. For the default address book, just type

```
#> khard new
```

```

select an event
Mar  Mo Tu We Th Fr Sa Su  Saturday, 2020/03/21 (3 weeks ago)
    24 25 26 27 28 29  1  Sunday, 2020/03/22 (~2 weeks ago)
    2  3  4  5  6  7  8  Monday, 2020/03/23 (~2 weeks ago)
    9 10 11 12 13 14 15  Tuesday, 2020/03/24 (~2 weeks ago)
   16 17 18 19 20 21 22  Wednesday, 2020/03/25 (~2 weeks ago)
   23 24 25 26 27 28 29  Thursday, 2020/03/26 (~2 weeks ago)
Apr  30 31  1  2  3  4  5  Friday, 2020/03/27 (~2 weeks ago)
    6  7  8  9 10 11 12  Saturday, 2020/03/28 (2 weeks ago)
    13 14 15 16 17 18 19  FOSSMeet talk
    20 21 22 23 24 25 26
May  27 28 29 30  1  2  3  Sunday, 2020/03/29 (~1 week ago)
    4  5  6  7  8  9 10  Monday, 2020/03/30 (~1 week ago)
   11 12 13 14 15 16 17  Linuxmag: , caldav servers
   18 19 20 21 22 23 24
   25 26 27 28 29 30 31  Tuesday, 2020/03/31 (~1 week ago)
Jun  1  2  3  4  5  6  7  Wednesday, 2020/04/01 (~1 week ago)
    8  9 10 11 12 13 14  Thursday, 2020/04/02 (~1 week ago)
   15 16 17 18 19 20 21  Friday, 2020/04/03 (~1 week ago)
   22 23 24 25 26 27 28  Saturday, 2020/04/04 (1 week ago)
Jul  29 30  1  2  3  4  5  Sunday, 2020/04/05 (6 days ago)
khal v0.9.10 | q: quit, ?: help

```

Figure 3: The command-line khal calendar, showing a local copy of the same calendar shown in Figure 2.

Listing 6: Configuring khal

```
[calendars]
  highlight_event_days = true
  [[work]]
    path = ~/.pim/work
    type = calendar
    color = dark green
  [[personal]]
    path = ~/.pim/personal
    type = calendar
    color = yellow
[default]
default_calendar = work
highlight_event_days = true
timedelta = 15d
#Dates formatting
[locale]
local_timezone= Europe/Rome
firstweekday = 0
timeformat = %H:%M
dateformat = %Y/%m/%d
longdateformat = %Y/%m/%d
datetimetypeformat = %Y/%m/%d %H:%M
longdatetimetypeformat = %Y/%m/%d %H:%M
```

Listing 7: Configuring khard

```
[addressbooks]
  [[All]]
    path = ~/.pim/contacts
[general]
debug = no
default_action = list
editor = gedit
merge_editor = /usr/share/doc/khard/examples/
sdiff/sdiff_khard_wrapper.sh
[contact table]
display = first_name
group_by_addressbook = no
sort = last_name
```

and answer khard’s questions. If you need to work on a non-default address book, specify it with the `-a` switch. It is also possible to configure Mutt to add or fetch addresses from khard, with the following two instructions that are explained in detail in the khard documentation [10]:

```
macro pager,index a
"<pipe-message>khard add-email<return>"
"add the sender address to khard"
set query_command= "khard email --parsable %s"
```

Khard and khal are very simple, very quick ways to read or write personal data from a Linux terminal. Their real power, however, is their support for automatic import and export of data to or from Nextcloud or any other CalDAV/CardDAV server. To load two new events into my Nextcloud work calendar (Figure 4) I used the following commands, and then ran `vdirsyncer` again:

```
khal -c khal-config-local.conf
new -a work '2020/03/24 09:00'
'2020/03/24 11:30' 'Engineers meeting'
khal -c khal-config-local.conf
new -a work '2020/03/25 09:00'
'2020/03/25 18:00' 'Project meeting'
```

Listing 8: List of Contacts

```
#> khard
Address book: All
Index Name Phone E-Mail UID
1 Bruce Wayne HOME,VOICE: 555-5555 HOME: batman@example.com b
2 Superman HOME,VOICE: HOME: 3
```

With khal, you can add hundreds of events to multiple calendars with a shell script that reads the events to be added from a plain text file. The plain text file may be a spreadsheet in CSV format, a database dump, or anything else, as long as it stores one event per line in a format similar to:

```
Engineers meeting|work|2020/03/24
09:00|2020/03/24 11:30
```

A script would need to read that file one line at a time, assigning each pipe-separated field to a variable (e.g., in the example above, `$EVENT`, `$CALENDAR`, `$START`, and `$END`) and then launch khal as follows, once per line:

```
khal -c $CONFIG_FILE new
-a $CALENDAR $START $END $EVENT
```

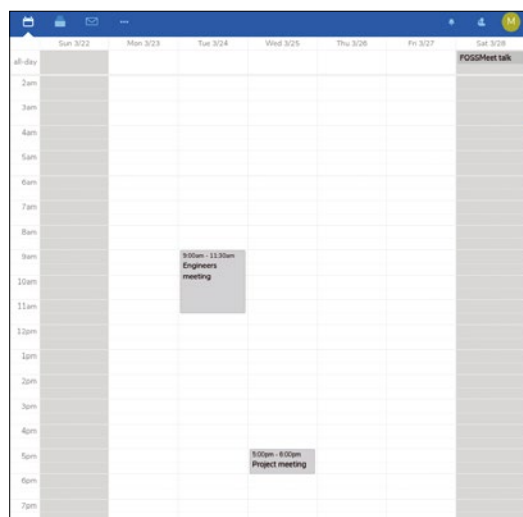
In a real script, the above command should be modified to handle spaces inside each variable. But once the input file has been read, one call to `vdirsyncer` will copy all those events to the Nextcloud calendar.

Importing addresses with khard works in the same way, with one important difference: With khard, you must write all the contact data in a template generated by khard with the following command:

```
khard -c khard.conf show --format=yaml >
template.yaml
```

Once you have the template, you can replace the value of all parameters with the right ones for your contact inside the file. Listing 9 shows the content of the `template.yaml` file, when filled with Wonder Woman’s contact information; Figure 5 shows the corresponding contact in the Nextcloud address book, after running the following khard command:

Figure 4: After synchronization, events created with khal on the desktop also display in the Nextcloud interface.



```
khard -c khard.conf new -i template.yaml
```

Just like with `khal`, you can use the `khard` command above inside a loop, which fills the template with new values read from a plain text file at every iteration and calls `vdirsyncer` when it has finished.

Backups and Post-Processing

As a side benefit of `vdirsyncer`, you can maintain a full backup of all your CalDAV/CardDAV data, regardless of what you do with it. However, that is not the only way to export or back up that data, and sometimes it is not even the best one. If you want to back up all of the personal data of all your Nextcloud installation's users, for example, you can run the Bash script, `calcardbackup` [11], on the server as follows:

```
sudo -u www-data /PATH/TO/calcardbackup ➤
$NEXTCLOUD_ROOT_FOLDER
```

The result will be a tar archive containing several `.ics` or `.vcf` files with names like `USERNAME-CALENDAR-NAME`, with each file containing one whole calendar or address book. The `calcardbackup` website also explains how to compress or encrypt backups or just back up the data of selected users.

Personally, I use `calcardbackup` any time I need to export all my calendar events, in a simpler CSV format, which is much better suited for further processing. To do that, I wrote a very quick and dirty script called `ics2csv.pl` (Listing 10). Listing 11 shows how to run the script, and its resulting output.

Listing 10 shows the power of storing your personal data in any CalDAV/CardDAV server, be it Nextcloud or anything else. Once you have done that, not only can you access the data anywhere on any device, the data also becomes very easy to reuse. It took me less than 15 minutes, testing included, to put together the whole thing (Listing 10), including copying and pasting.

Lines 1 to 10 of Listing 10 call the Perl libraries that perform several variable checks or provide time formatting functions. They then define all the variables I need and print the header line. Lines 11 and 12 scan the `.ics` file passed as an argument, skipping all lines except those that start with any of the strings in line 12.

Line 17 spots the variable containing the calendar name and saves its lowercase version into `$CALENDAR`. Line 19 does the same job for the event summary.

Lines 21 to 31 recognize and reformat the current event's start date, which is `DTSTART` in `.ics` files. Lines 33 to 43 do the same for the end date.

Whenever it finds the string `END:VEVENT` (line 45), the script knows that the description of

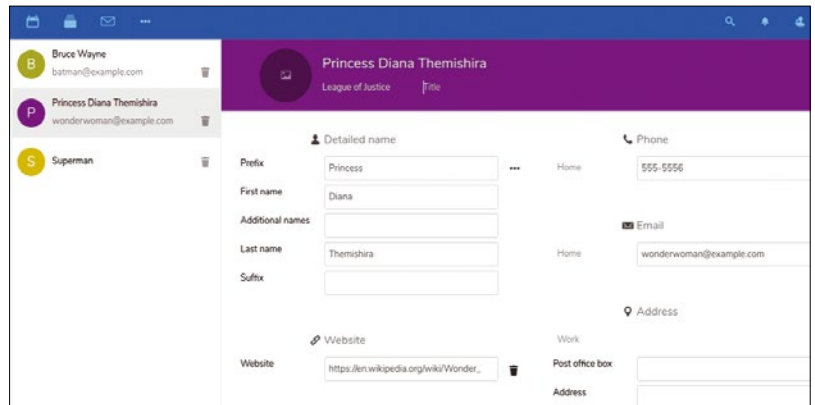


Figure 5: Wonder Woman's contact information displayed in the Nextcloud Contacts app.

an event has finished, and it appends a string with the calendar name, start time, event description, and end time (separated by pipes) to an array called `@APPOINTMENTS`.

Listing 9: template.yaml

```
Prefix      : Princess
First name  : Diana
Additional  :
Last name   : Themishira
Suffix      :

Nickname    :

Anniversary :
Birthday    :

Organisation : League of Justice

Title       :
Role        :

Phone       :
  HOME,VOICE : 555-5556

Email       :
  HOME       : wonderwoman@example.com

Address     :
  WORK:
    Box      :
    Extended :
    Street   :
    Code     :
    City     :
    Region   : Mount Olympus
    Country  : Planet Earth

Categories  :

Webpage     : https://en.wikipedia.org/wiki/Wonder_Woman

Private     :
  Jabber    :
  Skype    :
  Twitter   : wonderwoman_official

Note       :
```

Listing 10: ics2csv.pl

```

01 #! /usr/bin/perl
02 use strict;
03 use POSIX 'strftime';
04
05 my $TODAY = strftime '%Y%m%d', localtime;
06 my ($CALENDAR, $SUMMARY, $START, $STARTDAY, $END, $ENDDAY,
    $STARTHOUR, $ENDHOUR);
07 my @APPOINTMENTS;
08
09 print "## All events starting from $TODAY\n\n";
10
11 while (<>) {
12 next unless ($_ =~ m/^(X-WR-CALNAME|SUMMARY|DTSTART|DTEND|
    END:VEVENT)/);
13
14 s/;VALUE=DATE//;
15 s/;TZID=Europe/Rome//;
16
17 if ($_ =~ m/^(X-WR-CALNAME:(.*)\r\n/) { $CALENDAR = $1;
    $CALENDAR = lc($CALENDAR)}
18
19 if ($_ =~ m/^(SUMMARY:(.*)\r\n/){ $SUMMARY = $1; }
20
21 if ($_ =~ m/^(DTSTART:(\w*)/){
22 $START = $1;
23 $STARTDAY = substr($START, 0,8);
24 $STARTHOUR = substr($START, 9);
25 substr($STARTDAY,4,0) = '-';
26 substr($STARTDAY,7,0) = '-';
27 if (length($STARTHOUR) > 0) {
28 substr($STARTHOUR,2,0) = ':';
29 substr($STARTHOUR,-2) = '';
30 }
31 }
32
33 if ($_ =~ m/^(DTEND:(\w*)/){
34 $END = $1;
35 $ENDDAY = substr($END, 0,8);
36 $ENDHOUR = substr($END, 9);
37 substr($ENDDAY,4,0) = '-';
38 substr($ENDDAY,7,0) = '-';
39 if (length($ENDHOUR) > 0) {
40 substr($ENDHOUR,2,0) = ':';
41 substr($ENDHOUR,-2) = '';
42 }
43 }
44
45 if ($_ =~ m/^(END:VEVENT/){
46
47 push (@APPOINTMENTS,sprintf("%10.10s %5.5s | %-10.10s |
    %-35.35s | %10.10s %5.5s\n", $STARTDAY,$STARTHOUR,
    $CALENDAR, $SUMMARY,$ENDDAY, $ENDHOUR));
48 }
49 }
50
51 for my $APP (sort @APPOINTMENTS) {
52 my $CURRENTDAY = substr($APP,0,10);
53 $CURRENTDAY =~ s/-//g;
54 print "$APP\n" if ($CURRENTDAY >= $TODAY);
55 }

```

After the whole .ics file has been scanned, all events are written to standard output, in chronological order (lines 51 to 55).

Radicale

If Nextcloud doesn't meet your needs, another option is the lightweight CalDAV/CardDAV server called Radicale [12]. A Python 3 package, Radicale hosts multiple calendars and contacts for multiple users, with several options to restrict and log accesses. It does not support all the CalDav/CardDAV specifications' features, but it is adequate for small scale usage. Above all, Radicale is very simple to install and configure. Besides Python 3, its only dependencies are the `htpasswd` utility to encrypt user passwords and the Python 3 libraries to decrypt them.

Radicale's website explains step by step how to install Radicale with the pip package manager and run it for one single user. However, I have found that on Ubuntu and other distributions, a Radicale multi-user installation is as simple as typing the following:

```

#> sudo apt-get install radicale apache2-utils
python3-bcrypt python3-passlib
#> systemctl start radicale

```

This makes it much easier to keep current afterwards.

The `apache2-utils` package contains the `htpasswd` program, and the other two packages con-

Listing 12: Syncing Radicale to Nextcloud with vdirsyncer

```

[general]
status_path = "~/vdirsyncer/status/"

[pair cal_local_to_radicale]
a = "my_cal_local"
b = "my_cal_radicale"
collections = ["from a", "from b"]

[storage my_cal_local]
type = "filesystem"
path = "~/pim/"
fileext = ".ics"

[storage my_cal_radicale]
type = "caldav"
url = "http://localhost:5232"
username = "my_radicale_user_name"
password = "my_radicale_password"

```

Listing 11: Running ics2csv.pl

```

#> ics2csv.pl calcardbackup-2030-03-01/marco-work.ics > calendar.csv
#> cat calendar.csv
## All events starting from 20200301

2020-03-28 | work | FOSSMeet talk | 2020-03-29
2020-03-30 | work | Linuxmag: caldav servers | 2020-03-31

```


tain the required Python libraries. Radicale's default configuration is enough to run and access Radicale only on your local machine. To make it accessible from other computers or from the Internet, you need to "bind" Radicale to other addresses. To learn how to do this simple process, see Radicale's documentation [12]. I strongly suggest, however, that you play with a local-only installation before going live. For that, the only critical settings in `/etc/radicale/config` define the location and encryption method of usernames and passwords as follows:

```
[auth]
type = htpasswd
htpasswd_filename = /etc/radicale/users
htpasswd_encryption = bcrypt
```

To create a Radicale user and store the user's encrypted passwords in `htpasswd_filename`, run the following command for each user and follow its instructions:

```
sudo htpasswd -B /etc/radicale/users marco
```

Now, you can restart Radicale, point your browser to `http://localhost:5232/`, log in, and start creating calendars and address books in the interface shown in Figure 6. Of course, since Radicale is only a server, you will need clients like khal, khard, or Thunderbird to actually read and write those databases. To import personal data from other servers, just use vdirsyncer, and it will create automatically all the desired calendars and address books. When I configured vdirsyncer as

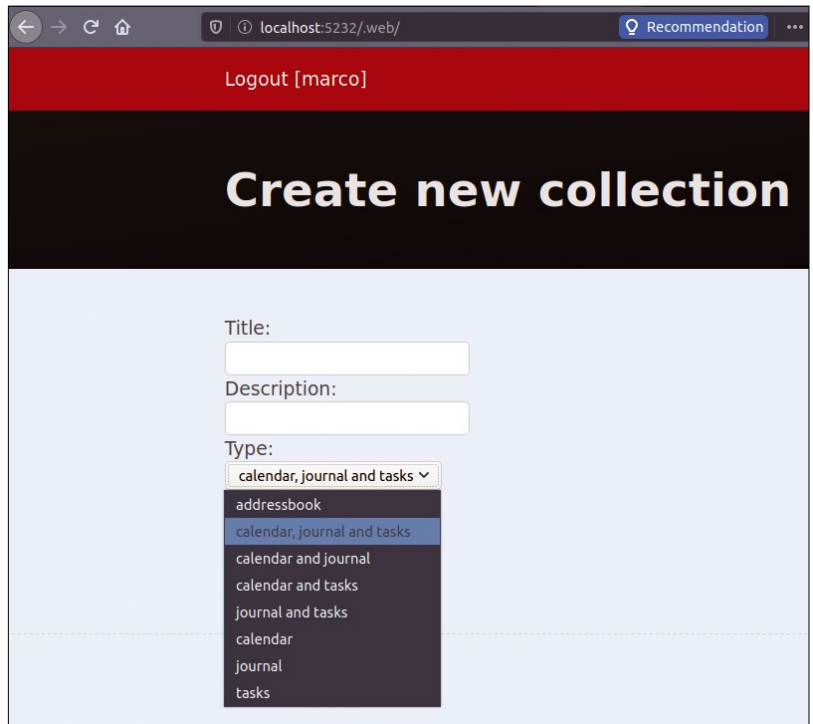


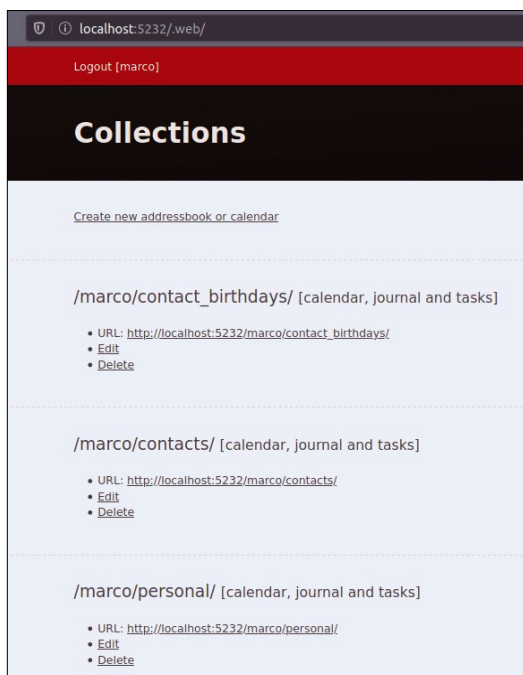
Figure 6: Radicale offers a very simple web interface to create calendars or address books for multiple users, which can then be filled with data from other clients.

shown in Listing 12, it uploaded all the data previously copied on my computer from the Nextcloud server to my Radicale server (see Figure 7).

Conclusions

Personal contacts and calendars are essential data for every computer user. With the utilities and methods described here, you have the basic tools to efficiently create, manage, process, store, and share your personal data. ■■■

Figure 7: Calendars automatically imported from Nextcloud show up under Collections in Radicale.

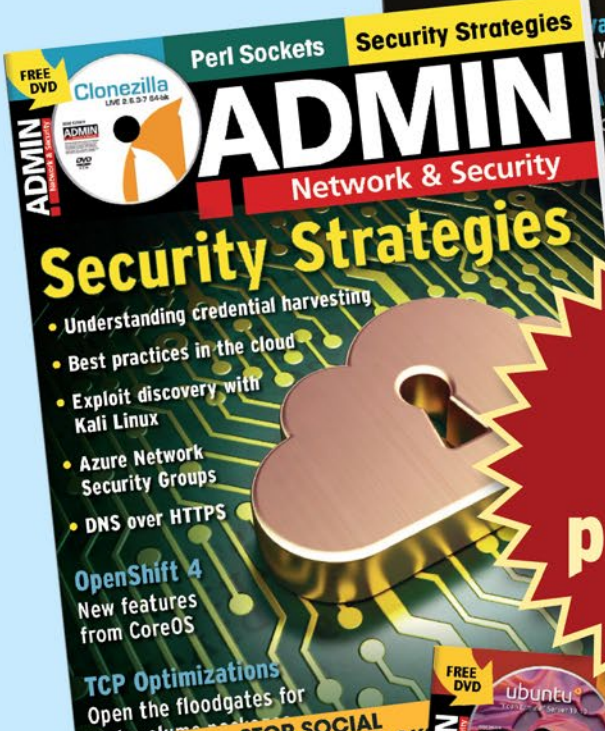
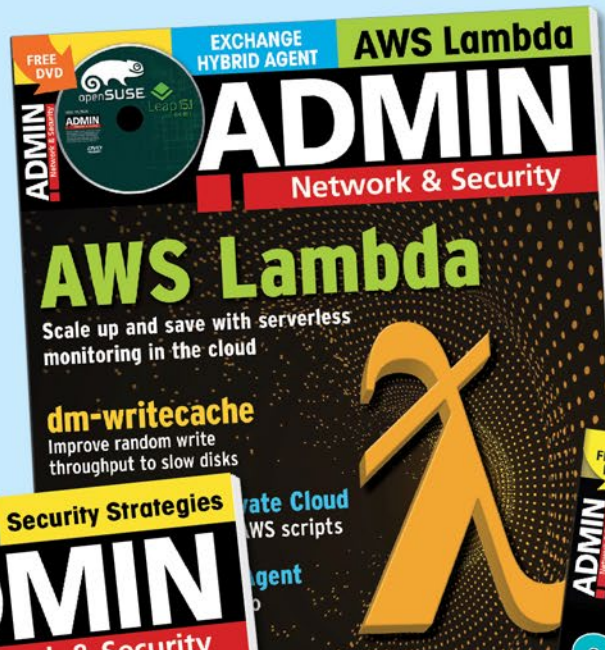


The Author

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

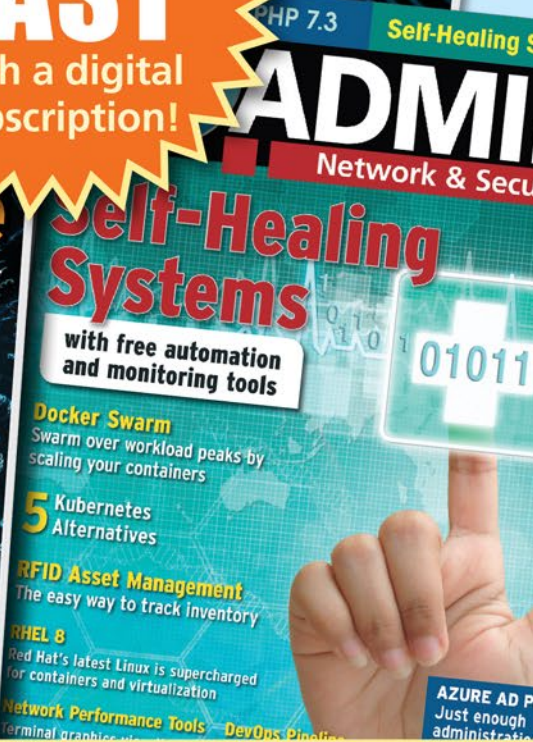
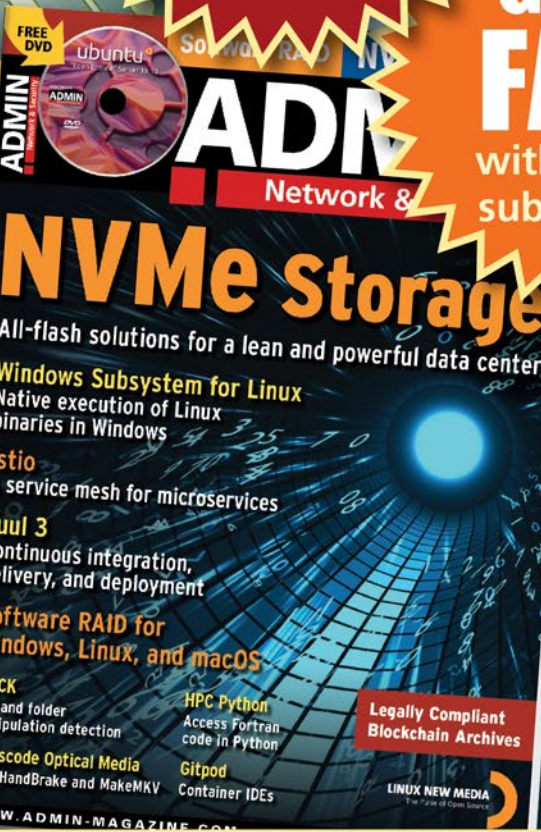
Info

- [1] iCalendar: <https://icalendar.org/>
- [2] Virtual Contact File: <https://wiki.fileformat.com/email/vcf/>
- [3] Nextcloud: www.nextcloud.com
- [4] Nextcloud tutorial: <http://freesoftware.zona-m.net/nextcloud-16-review/>
- [5] vdirsyncer: <http://vdirsyncer.pimutils.org/>
- [6] vdirsyncer manual: <http://readthedocs.org/projects/vdirsyncer/downloads/pdf/latest/>
- [7] vdirsyncer configuration examples: <https://www.dj-bauer.de/terminal-calendar-with-khal-en.html>
- [8] khard: <https://github.com/scheibler/khard>
- [9] khal: <https://github.com/pimutils/khal>
- [10] Using khard with Mutt: <https://khard.readthedocs.io/en/v0.14.0/>
- [11] calcardbackup: <https://codeberg.org/BernieO/calcardbackup>
- [12] Radicale: <https://radicale.org/2.1.html>



6 issues per year!

GET IT FAST with a digital subscription!



shop.linuxnewmedia.com

REAL SOLUTIONS *for* REAL NETWORKS

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

It isn't all Windows anymore - and it isn't all Linux.

A router is more than a router. A storage device is more than a disk. And the potential intruder who is looking for a way around your security system might have some tricks that even you don't know.

Keep your network tuned and ready for the challenges with the one magazine that is all for admins.

Improve your admin skills with practical articles on:

- Security
- Cloud computing
- DevOps
- HPC
- Storage and more!

SUBSCRIBE NOW
SAVE 30%

ADMIN
Network & Security

shop.linuxnewmedia.com

FOSSPicks

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



Like tardy London buses, Graham has waited months for a decent open source instant messenger client to arrive, and then in this month's FOSSPicks, he found two. Perfect for staying in touch with friends and family from the comfort of your own sofa. **BY GRAHAM MORRISON**

Radio scanning

SDRangel

Software-defined radio (SDR) is a subject that may sound too nerdy even for the average Linux user, perhaps the equivalent of Citizens Band (CB) radio in the 1970s and 1980s. But like CB radio, the useful and technically geeky aspects of this technology can also make it very appealing. For one thing, SDR needs very little hardware to get started. A simple, low-cost, DVB USB receiver is all you need initially, and many of us have spare devices simply because digital radio and

television has moved from broadcast to streaming. SDR replaces the plethora of hardware needed to investigate the radio signals in the ether with clever algorithms and processing, allowing you to decode everything from a remote thermometer to airline communication using software alone.

On Linux, there are a huge number of tools and utilities dedicated to working with SDR. GNU Radio, for example, is technically complex, but Gqrx SDR is easier to use and understand. SDRangel sits

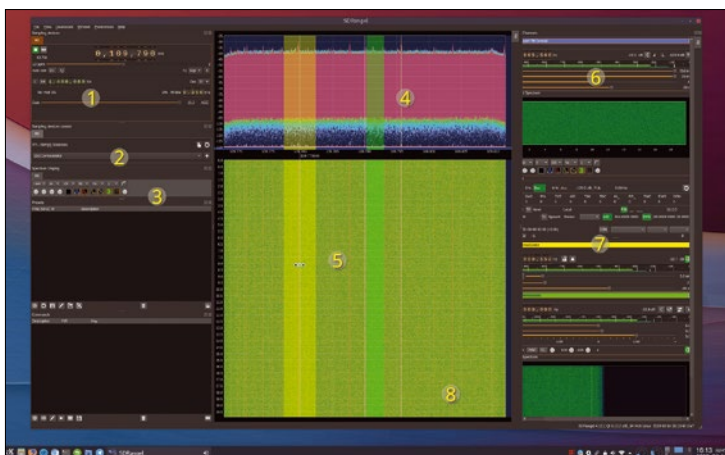
somewhere between GNU Radio and Gqrx in complexity. It is still capable of advanced analysis, but thanks to the UI, everything is also easily discoverable without having to read reams of documentation. Like Gqrx, its primary function is to let you scan and probe specific frequency ranges for signals that can potentially be decoded/demodulated into either a digital or an analog signal. A digital signal could be to a remote device with a proprietary controller you want to decode and control from your Raspberry Pi, whereas an analog signal would be something you could listen to, such as the chatter from a pilot to an airport control tower.

There are many steps between selecting your sampling device, finding the right frequency, and decoding a signal, but the heart of SDRangel is common to many SDR applications: the spectrogram waterfall. This is a scrolling chart where each horizontal line is a slice of a frequency range, with pixels colored according to the amplitude or intensity of a signal at that point. It's a waterfall, because, as each slice is measured, the previous slice moves down, creating a scrolling waterfall of color. This kind of chart makes it easy to spot signals, either as a continuous

line for audio, or as intermittent blobs for bursts of digital data. Scanning the de-regulated frequencies between 433.050 MHz to 434.790 MHz, for instance, can reveal all types of signals from your environment, including automatic garage doors, weather monitors, and even remote car key fobs. Some of these will be momentary, such as when you unlock your car, whereas others will be periodic, perhaps repeating every 30 seconds or every hour. To capture all of these, you need to play around with both the frequency ranges you're scanning, and the scanning period or speed of the waterfall. This is because you'll miss a signal with an hourly period if the previous transmission has scrolled off the screen, for instance. While this overview may make the process sound arduous, the reality is quite the opposite. Discovering signals, zooming in to them, working out their repeating periods, and finally trying to decode the pulses in a repeating signal is fascinating and can provide a great deal of insight into how the technology around you works and how it communicates.

Project Website

<https://github.com/f4exb/sdrangel>



1. Tuning: SDRangel works with a huge variety of hardware and can tune in to any frequency that any device is capable of. **2. Demodulation:** More than one demodulation scheme can be used on a single device, such as AM decoding for both audio and digital signals. **3. Display controls:** Almost everything about the spectrogram display can be changed, from its color to its measuring algorithm. **4. Spectrogram:** See the incoming signal's amplitude across a frequency window in real time. **5. Waterfall:** Each slice of a spectrogram scrolls down making momentary signals easy to spot. **6. Demodulators:** Tune and view each demodulator separately. **7. Audio output:** Listen to signals you tune in to, such as AM and FM radio broadcasts. **8. Tuning characteristics:** Colored bands are shown in the waterfall to represent the demodulator tuning frequencies.

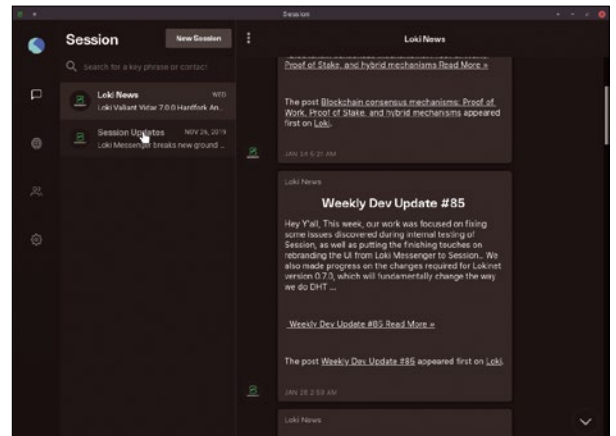
Instant messenger

Session

Signal is widely recognized as the instant messenger most likely to be trusted by security experts. This is because both its server and client are open source, and it defaults to using end-to-end encryption. In theory, only you and the people in your group can decode messages, because the decryption keys are local to individual devices and not stored on any remote server. However, Signal's user interface and functionality are a little austere for users of WhatsApp and Telegram, making it a difficult proposition for those of us wanting to convince our friends and family to switch. The Session messaging app is a fork of Signal that also happens to look good. In theory, it could be just as secure as Signal. But

it's important to emphasize "in theory," because while the provenance of Signal is clear, thanks to its founder Moxie Marlinspike being a prominent security and privacy advocate, it's too early to say whether the non-profit Loki Project behind Session can be equally trusted. Its FAQ does state the company is in the process of arranging a full third-party code audit, so we'll have to wait and see.

However, Session is still definitely worth experimenting with, and not just because it looks lovely. First, you don't need a mobile number to be able to create an account. Instead, when you launch the app, clicking *Create Account* will generate a unique session ID that you use for your login. Creating a password is optional. This is



If Session passes an independent security audit, it could become a brilliant secure messaging platform.

a huge advantage over Telegram. Session's client-to-client communication is managed with a server, rather than peer-to-peer, although the messages themselves are sent via a decentralized onion routing network similar to Tor. This allows for public and private groups and asynchronous messaging, as well as session syncing across multiple devices simply by using the same session ID. It works brilliantly and we sincerely hope it's able to prove its privacy credentials.

Project Website

<https://getsession.org>

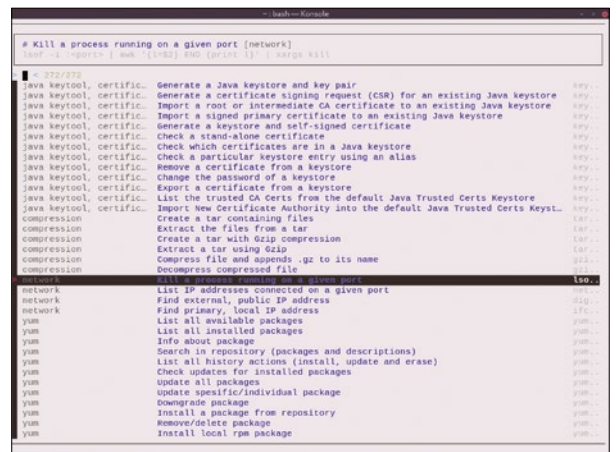
Terminal cheat sheet

navi

It's awesome that so many people are now using the command line. This is because rather than being viewed as an anachronism, it's quite rightly now being seen as simply the best way to get many kinds of jobs done. But it can also still be quite intimidating for users more familiar with the desktop, and it's surprising how few tools there are to help them, other than the old tried and tested method of reading man pages. This is where navi can help. It's a simple command-line tool that describes itself as an "interactive cheat sheet" packed full of commands to help you accomplish a variety of common command-line tasks.

When first launched, you're presented with a huge list of

entries sorted by categories that mostly reflect a command. There's `git`, `grep`, `docker`, and `yum`, for example. But there are also more general categories in the list, like `network`, `compression` and, `android`. The list can be filtered in real time using `fzf` by typing whatever you're searching for. Type `file`, for example, and the list is limited to entries that include only that word. Select any entry and a small top pane explains how to do something useful from the command line. It could be listing IP addresses, compressing a folder, updating a `git` branch, or even viewing the weather. Press `Return`, and the commands from the selected recipe are pasted into the terminal for you to easily edit and execute. It can also



Navi contains over 250 recipes to help you get the most out of the command line.

be used as a shell widget rather than as a full-screen terminal app, which means you can summon its functionality to appear directly beneath the terminal prompt.

Project Website

<https://github.com/denisidoro/navi>

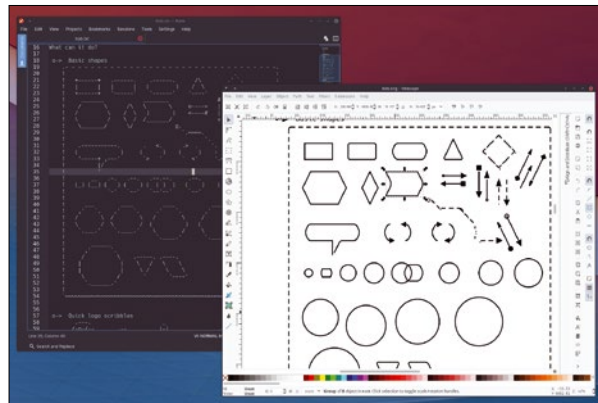
Text to diagrams

Svgbob

If you've ever looked into writing documentation or managing the documentation for a project, you'll know that simple ASCII text-based writing and publishing formats have taken over from application-specific formats. This is partly thanks to a movement that handles text just like code, with a similar publishing process that involves `git` branches, commits, CI-systems, and reviews. This process works because the text is flat ASCII, usually written using Markdown, and not a shape-shifting XML LibreOffice document or binary blob. Simple text like this produces diffs and patches just like code, which are easier to review, to version, and to maintain. The only question in this documentation nirvana is what do you do

with illustrations? Binary blobs don't work well with `git` because they can't easily be reviewed or versioned, and text-based illustration formats like SVG change too much between modifications to be useful as text.

Helping to fill this gap is Svgbob, a small utility that will convert illustrations drawn in simple ASCII into perfectly rendered SVGs. It does this by interpreting the special characters in most character sets into graphical primitives like polygons, ellipses, and lines. Draw a line using multiple "-" characters, for example, pipe the file through Svgbob, and you'll get SVG output containing a perfect line. But you can also draw remarkably complicated diagrams. Arrows are rendered directly from their characters, and there are even special



Transform simple ASCII text into SVG perfection with Svgbob.

sequences to indicate components in a circuit, nodes on a chart, and colors and grids of all shapes and sizes. Even Unicode characters can be used to draw direct representations in SVG. But the best thing is that everything can be stored within a simple text file and edited and managed just like normal text.

Project Website
<https://github.com/ivanceras/svgbob>

Workstation emulator

emulator-sun-2

As most emulators are targeted at recreating old gaming systems, or home computers where gaming was a major component of their popularity, we often feature emulators in the gaming section of these pages. But emulation, of course, isn't just useful for playing old games, it's also an essential way of keeping old software alive long after the original hardware has melted under its own battery acid. Emulation also allows you to play with hardware that may have been completely inaccessible on its original release, which is the case for most of us and the mighty Sun-2, a brand of workstation built by Sun Microsystems in the early 1980s. The bottom-of-the-range Sun-2/120 could be purchased for a measly \$30,000 at the time, which is about \$70,000 in today's

money. That would buy you 4MB of RAM, 100MB of storage and a Motorola 68010.

Which coincidentally is the exact default configuration you get from running `emulator-sun-2`, a software emulator of the venerable Sun-2/120. You do need to have access to the firmware, a SunOS disk image with NetBSD, and something for the emulated SCSI tape, but these are relatively easy to track down. You can then simply boot up the emulator with a command, and you'll soon be presented with the limited glory of SunOS. The easiest thing from this point is to boot into NetBSD (type `b vmunix`), and you can operate a computer like you're in WarGames, complete with Times Roman font rendering, monochrome monitor, and an emulation of the unique keyboard that



Emulating old proprietary hardware on a modern PC is a great way to appreciate how far we've come.

came with the hardware. If you've not used the real hardware before, you need to read some contemporary instructions about how the system operates and works, but that itself is interesting and a reminder of what things were like before Bash, mice, and GUIs became ubiquitous.

Project Website
<https://github.com/lisper/emulator-sun-2>

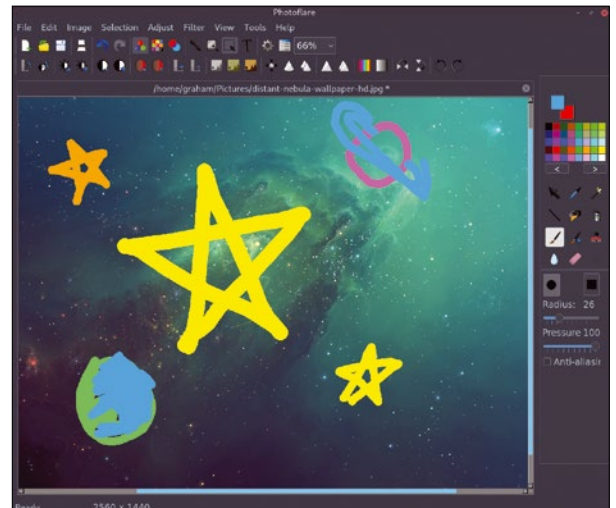
Image editor

Photoflare

Despite having Gimp, Glimpse, and the wonderful Krita, there aren't many image editors or drawing tools if you only need to make a quick edit or sketch. Pinta and MyPaint are good options, but there's definitely room for something different. Photoflare is something different. Like Krita, it's cross platform, which will hopefully help to give the project lots of momentum. Unusual for a Linux application, its development was inspired by a Windows application, PhotoFiltre, a proprietary image retouching tool that's been available for many years. What's great about Photoflare is that it's very much a WYSIWYG kind of application, with every editing function easily accessible and easy to understand. If you've

used Microsoft Paint, you'll know what almost every function does.

Whether you're starting a fresh image or editing something that already exists, Photoflare quickly lets you make the changes you need to make. Rotation, crops, transparent backgrounds, selection fill, brightness, and contrast are all immediately selectable, and the application itself responds quickly even with large images. There are some more advanced features, such as drop-shadow generation, but there are no layers, filters, or complex palette interactions to worry about. However, these features are coming with the currently in-development version 2, and along with them a very Windows-like split between the open source "community" edition and a paid-for edition with proprietary plugins. This is initially being



Sometimes, all you need is a fast, easy-to-use image editor to get the job done. Photoflare is that editor.

done to help fund development of those advanced features, including layers, but we hope that if it's successful, those proprietary plugins will eventually make their way into the community editions, or at least as open source code on the project's GitHub repository.

Project Website
<https://photoflare.io/>

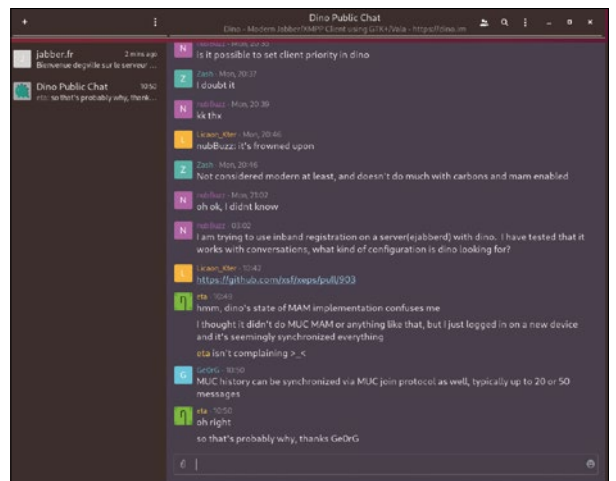
Instant messenger

Dino

It feels like we're at an interesting juncture for instant messaging. On the one hand, we have the huge popularity of proprietary protocols run by corporates, like Facebook's WhatsApp, Apple's FaceTime, and Google's Meet and Hangouts, and Microsoft Teams. On the other hand, more people are wanting to control their own data, and are aware of the privacy implications of trusting a third-party to host and control some of their most private conversations. This may account for Telegram's growing popularity. The client is famously open source, but the server is infamously not. Unlike WhatsApp, it doesn't enable end-to-end encryption by default. Signal is another option and is truly open source and encrypted end-to-end by default. But this is also

its disadvantage, because it can't handle group membership well, or sync messages across devices, although all this is being actively worked on. We even looked at Session, a fork of Signal, earlier. Matrix is another great open source option that's recently been adopted by Mozilla to replace its ancient IRC channels. It's good, but complex for general users not familiar with its federation.

Dino doesn't quite solve any of these problems, but it does remind us that the old familiar open Jabber/XMPP protocol is still active. There was a time when many desktop and mobile clients worked with this protocol, even with federation across services and corporations, and it still works well. Dino uses the Jabber/XMPP protocol to chat with individuals or add a group from a remote server



Even Google used the Jabber protocol for its messenger, but when many other services went proprietary, many of our contacts went with it.

to your client and chat. It also incorporates end-to-end encryption, via either OMEMO or OpenPGP, which once enabled, means messages that leave your system can only be decrypted by the receiver, regardless. Despite its alpha state, it works well and is only missing more users. Which means it's time to brush off that old Jabber ID.

Project Website
<https://github.com/dino/dino>

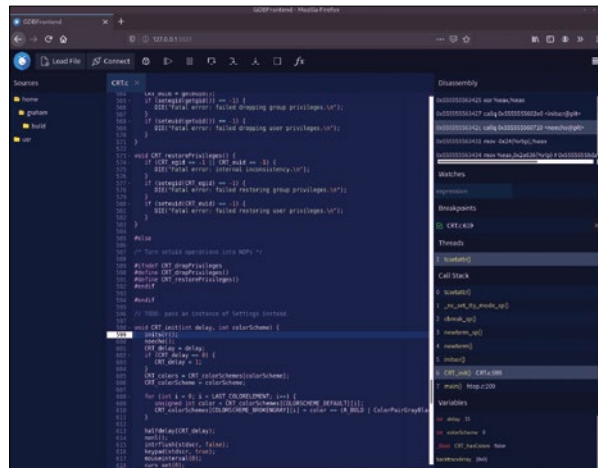
Debugging GUI

GDBFrontend

Even if you're not hugely into code, the GNU Debugger (GDB) is a fascinating tool. It allows programmers to step through each line of compiled code in the binaries their projects produce. Learning how to use GDB is a subject that fills books. In essence, it helps developers fix bugs by allowing them to jump into their code when a crash occurs or to set a point in their code when they want to break out of its running state and into the debugging state where you can view the values held by variables and pointers, and perhaps step through each problem a line at a time. By default, all of this is accomplished from GDB's command-line interface. But there are graphical options,

including desktop GUIs, GDB integration with many of the most popular IDEs, and this, GDBFrontend.

Unlike other graphical helpers for GDB, which typically run on a desktop, GDBFrontend's GUI is accessed using a web browser. This is extremely useful if you only have access to a command line (and not a desktop), or if you want to debug an executable over a network. On one port is an interactive GDB session, which is exactly like running GDB on the command line, while another port hosts the web UI. You can use the web UI much like you would a native desktop application. You can load and execute a binary; set breakpoints; step through, step into, and jump over code;



If you have difficulty remembering the myriad GDB commands, GDBFrontend spawns a simple, powerful GUI in a local web session.

view variables, and a file browser to study the source. There's even a disassembler for binaries without the debugging symbols, and you quickly forget you're using a debugger in a web browser.

Project Website
<https://github.com/rohanrhu/gdb-frontent>

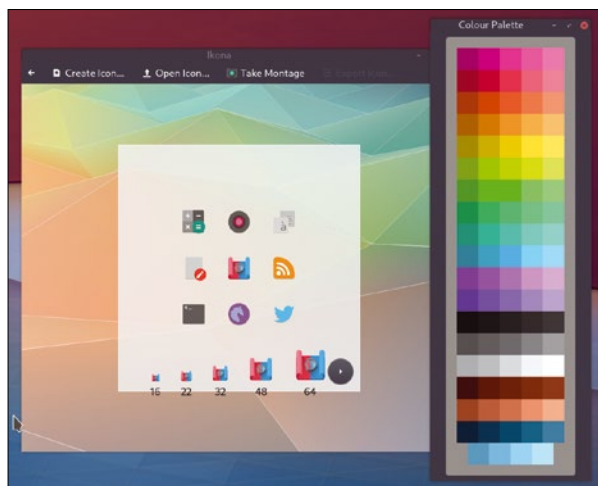
Icon preview

Ikona

Our desktops and applications wouldn't be the same without the icons used to manage and launch them or represent the files created by them. These icons don't just signify the branding of a specific distribution or desktop, they're often more like home furnishings, chosen to suit a user's personality or sense of aesthetics. This is why there's such a huge range, and so much effort is expended by designers to create them. And as with font creation, the process of designing and creating icons is horribly complicated. They may be simple graphics files, but to create one, you not only need an artist's sensibilities and skill, you need to apply those skills using a set of

curves, colors, and styles you can use across all the icons you create, often in collaboration with other people. Any help you can get to better integrate your designs into the diverse world of desktops will be hugely valuable.

This is why the KDE Plasma team have developed Ikona, an icon preview utility to help designers see how their work looks in context, without always having to install and test manually. It's designed to work alongside the drawing tool, easily lets you check an icon's palette, see how it looks in both light and dark themes, and view it alongside an assortment of other icons, all rendered in pixel perfection in sizes 16, 22, 32, 48, and 64 pixels wide. This is thanks to its new SVG-based format that



Ikona is one of the first KDE applications to be written in Rust, a language and platform that's becoming increasingly popular.

allows it to embed multiple sizes in a single file. Imported SVGs won't have this feature, but saved icons will. Saving an icon lets you choose which sizes you want to include, and you can also easily create a montage of icon options to share with other designers for feedback.

Project Website
<https://invent.kde.org/kde/ikona>

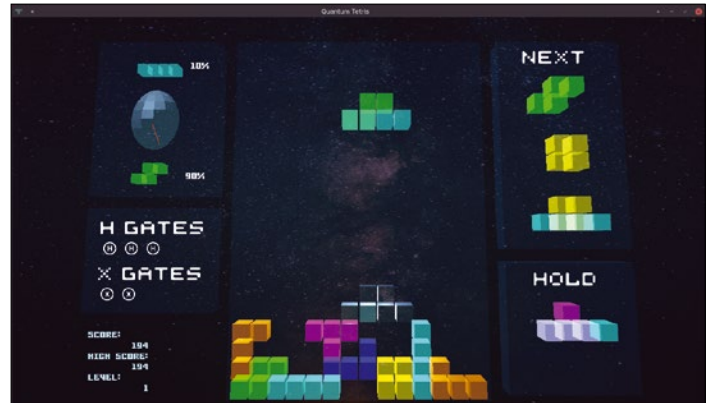
Superposition puzzle

Quantum Tetris

Tetris is a timeless classic. Blocks made from a different arrangement of four smaller blocks fall into a 2D well, and it's the player's job to fit these together into seamless vertical rows of four to evaporate them to stop blocks filling past the top, after which it's game over. But what if Tetris could also teach you something about the nature of matter and time itself? That's a job for Quantum Tetris, a unique take on an old classic that follows the same old rules and adds a few more borrowed directly from quantum mechanics. The first of these is the use of quantum superposition pieces. The special new "superposition" piece is two shapes at the same time. A probability percentage, shown in the top right, is used to decide which

shape the block becomes when it lands on the Tetris stack. When playing this piece, you find yourself gambling with the outcome, and placing it somewhere non-essential unless there's a dire need.

The second new rule is called "quantum entanglement." This rule binds two superposition pieces together across an imaginary vertical mirror plane through the middle of the play area. The second piece moves opposite to the first piece and will settle into the opposite state when the first piece lands. If you've played Tetris before, both these new rules break the muscle memory of your play style and are initially difficult to get your head



This game is amazingly polished and shows off the Godot game platform's capabilities.

around. But they both also fit perfectly into the original mechanic, adding a new level of complexity and challenge to a game that many of us have played hundreds of times before. The other great advantage is that you'll learn about quantum superposition and entanglement while playing.

Project Website
<http://quantumtetris.com/>

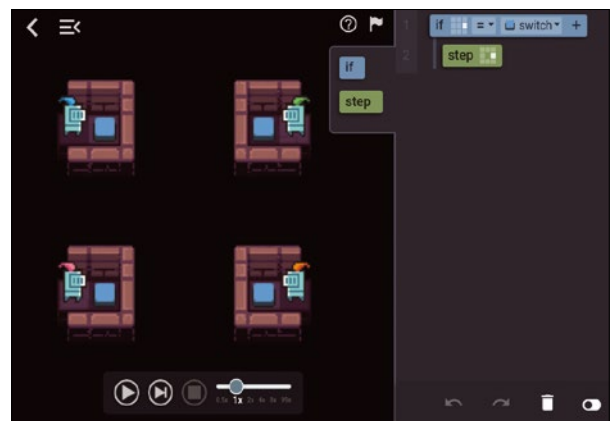
Programming game

Selfless Heroes

Despite many games trying to help people learn to program, designing such a game is a difficult and challenging task. Some, like Minecraft, make the coding and logic elements an advanced topic for those wanting to get the most out of a game, whereas others, like Robocode, make the code an integral part of your success. This is what Selfless Heroes does, except rather than asking you to code a battle robot's AI, it asks the player to use code blocks to help its characters navigate through a 2D puzzle. This is why the game looks so ordinary, in a cute 2D top-down way, which may help it attract players who might otherwise be put off by the idea of writing

code rather than twiddling a joystick.

To beat each level, you have to navigate your team of knights to a specific target in a set number of moves by writing a program using a set number of lines. Things start very simply, and each new command is introduced gently. The first level demands you only move your knights three steps in three lines, for example. You do this from a block-like coding environment, but you can impressively switch this to a text-based code editor to accomplish the same task. When you've constructed your algorithm, you need to hit the play button to execute your code in a single pass. If you succeed, there's even a testing phase to make sure it works



Teach people how to code, to test, and to iterate on their designs without them even knowing.

every time – just like real code. If it fails, you try again. You can even save your solution and work on a new one or revisit old challenges when you learn new techniques. It's so much fun that most players won't even realize they're learning how to code.

Project Website
<https://selflessheroes.fr/>

Implementing physics in a LÖVE game

Gravity

Video game animation is not simply a matter of making your characters move – you also have to consider the physics of the world in which they move.

BY PAUL BROWN

In issue 234 of *Linux Magazine* [1], I introduced LÖVE [2], the Lua-based framework used for creating 2D games, by drawing a character, Cubey McCubeFace, who could walk across the screen. Now I'm going to explore another aspect of LÖVE by going back into an animated world and causing an object to fall out of the sky.

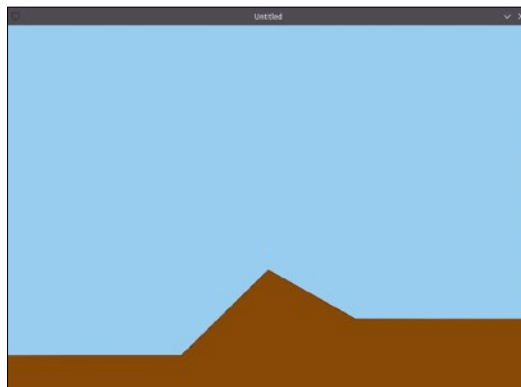
As long as your game characters are moving from side to side, things are more or less easy. The moment you need them to jump or fall, things get more complicated – that is, if you have to program a physics engine yourself. Luckily, LÖVE provides a way to simulate 2D rigid bodies in a realistic manner through its physics module. In this tutorial, I'll explore how that works [3].

Landscaping

First of all you need a playing field in which things can move around and collide with each other. I'll set up a "landscape" like the one you can see in Figure 1 and by drawing the outline of the terrain first (see Listing 1).

As you will remember from the article in the last issue of *Linux Magazine* [1], a LÖVE game is usually divided into three parts: the *load*, the *update*, and the *draw* (see the box "Anatomy of LÖVE" for more on this). You can use LÖVE's `polygon ()` function in the *draw* section to create the ground in your game (line 35). The `polygon ()` function takes a `mode` argument, `'line'` for an outline or `'fill'` for a filled polygon

Figure 1: The scenery for your game.



and then a bunch of coordinates for the polygon's vertices. If you then define two variables, say `w_width` and `w_height` (lines 2 and 3), for the size of the playing field, you can then use them to correctly place the ground. As the `(0, 0)` position in a LÖVE screen is in the upper left-hand corner, the first vertex for the ground, in the lower, left-hand corner, will be at `(0, w_height)`. The second vertex will be somewhere above that, so at `(0, w_height - some random number)`, and the third will be one-third across at the same height at `(w_width / 3, w_height - some random number)`; and so on. Coded into LÖVE, that would look like lines 9 to 26 in Listing 1.

Lua's `math.randomseed ()` module (line 1) makes sure that the `math.random ()` random functions (lines 9, 10, 14, and 16) will return a different set of numbers each time the game is run. Lines 12 to 17 make sure that the central mountain sticks out above the two sides of the playing field (i.e., that it is actually a mountain and not a valley),

Anatomy of LÖVE

A LÖVE game is usually split into three distinct parts, each defined by its own function:

- The `love.load ()` function is where you set things up. You load images, set the background, calculate the frames in each animation, set the initial values of variables, create objects, and so on.
- The `love.update ()` function is the main loop of the game. Here is where things change as the game progresses. You calculate the new coordinates for sprites; read in keystrokes, mouse movements, or other player-generated input; modify the playing field; and so on. The special variable `dt` is usually associated with `love.update ()` so you can calculate *game time*. `dt` contains the time that has passed since the last time `love.update ()` was called.
- The `love.draw ()` function is where you draw what will be seen on the screen after each iteration in `love.update ()`.

and lines 19 to 26 put all the vertices into a table that you can then use as an argument with `love.graphics.polygon ()` on line 35. When you run this program, it will show what you can see in Figure 2.

You may think filling in the “ground” shape would be as simple as adding

```
love.graphics.polygon ("fill", ground)
```

to the `love.draw ()` function, but that is not the case.

You see, LÖVE uses a very fast filling algorithm. It needs to if it has to redraw and fill several polygons many times a second. But the trade-off is that it is not very good at filling in concave shapes (i.e., shapes with dents and holes in them). When you try to ‘fill’ in the ground polygon, you get what you can see in Figure 3.

As you can see, the left side of the playing field is wrong: The fill algorithm has filled in a triangle that goes from the lower left-hand corner of the window to the top of the hill, cutting over the flat area in the left side of the field.

The way you solve this is with *triangulation*. LÖVE incorporates its own `math` module that includes a function, `triangulate ()` that breaks complex polygons into triangles.

Add the line

```
groundT = love.math.triangulate (ground)
```

Figure 2: The outline of the ground.

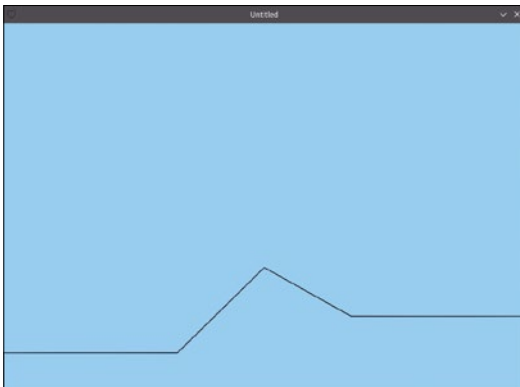
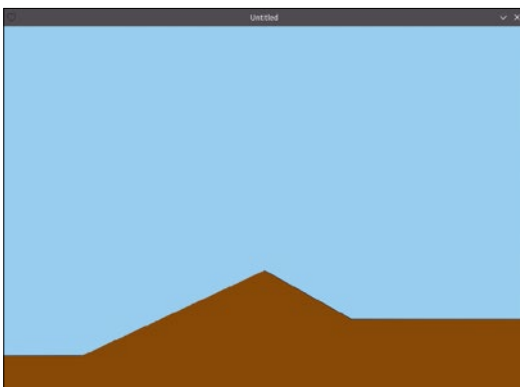


Figure 3: LÖVE’s fill algorithm struggles with concave shapes.



after you define the `ground` table, and `groundT` will fill up with the vertices from a bunch of triangles that, when put together, make up your polygon ground.

As all triangles are convex, you can then loop over each of them and fill each to draw the ground, as shown in lines 9 to 11 in Listing 2.

Listing 1: main.lua (Original)

```
01 math.randomseed(os.time())
02 w_width = 900
03 w_height = 600
04
05 function love.load ()
06     love.window.setMode (w_width, w_height, {resizable = false})
07     love.graphics.setBackgroundColor (0.5, 0.8, 1, 1)
08
09     level01 = w_height - (math.random (10, (w_height / 3)))
10     level02 = w_height - (math.random (10, (w_height / 3)))
11
12     if level01 < level02
13     then
14         mountain = level01 - (math.random (10, (w_height / 3)))
15     else
16         mountain = level02 - (math.random (10, (w_height / 3)))
17     end
18
19     ground = { 0, w_height,
20               0, level01,
21               w_width / 3, level01,
22               w_width / 2, mountain,
23               w_width * (2 / 3), level02,
24               w_width, level02,
25               w_width, w_height
26             }
27 end
28
29 function love.update ()
30
31 end
32
33 function love.draw ()
34     love.graphics.setColor (0, 0, 0, 1)
35     love.graphics.polygon ("line", ground)
36 end
```

Figure 4: A polygon built up from triangles.

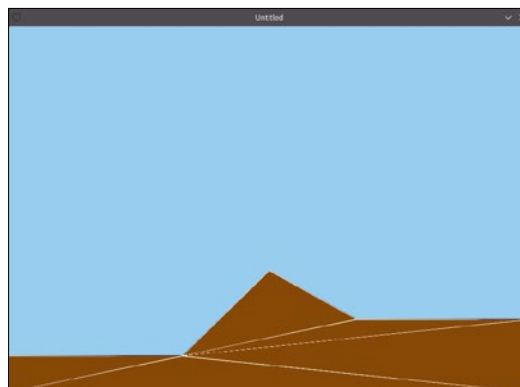


Figure 4 shows what the triangles look like when made visible.

Physical Ground

So far, the ground is just a graphical element and nothing will interact with it. In fact, no graphical element ever physically interacts with any other graphical element in LÖVE.

When graphical elements seem to fall, collide, and bounce, what they are really doing is taking the data for their position and rotation from invisible *bodies* defined by the physics module. These bodies are the ones doing the falling, colliding, and bouncing.

Before you start to make bodies tumble, you need some rules for your world. That

is the purpose of the `love.physics.setMeter ()` and `love.physics.newWorld ()` functions.

The `love.physics.setMeter ()` function determines how many pixels make a meter in your world. It is best to run this function before doing anything else with physics, since if you change it halfway through, things will get weird, as objects drawn in one scale before the change will remain in that scale, while objects drawn in the new scale will use the new scale. The default value for pixels-to-meters is 30 pixels for one meter, but you can change that to, say, 10 pixels to a meter with

```
love.physics.setMeter (10)
```

The `love.physics.newWorld ()` takes three parameters: the strength of the horizontal component of gravity (yes, you can have things falling sideways), the strength of the vertical component of gravity, and whether objects in this world can sleep.

In this example, gravity is going to behave as usual and drag things down towards the bottom of the world. It will do that at its regular rate of 9.81m/s², too. To do that, you can use `love.physics.newWorld ()` as shown on line 2 of Listing 3.

Listing 2: The Ground, draw ()

```
01 .
02 .
03 .
04 function love.draw ()
05     love.graphics.setColor (0, 0, 0, 1)
06     love.graphics.polygon ('line', ground)
07
08     love.graphics.setColor (0.5, 0.3, 0, 1)
09     for i=1, #groundT do
10         love.graphics.polygon ('fill', groundT [i])
11     end
12 end
13 .
14 .
15 .
```

Listing 3: pworld.lua

```
01 love.physics.setMeter (30)
02 world = love.physics.newWorld (0, 9.81 * 30, true)
03
04 Earth = {}
05
06 function Earth:init (terrain)
07     self.ground = {}
08     for i=1, #terrain.groundT do
09         self.ground[i] = {}
10         self.ground[i].body = love.physics.newBody (world, 0, 0, 'static')
11         self.ground[i].shape = love.physics.newPolygonShape (terrain.groundT [i])
12         self.ground[i].fixture = love.physics.newFixture (self.ground [i].body, self.ground [i].shape)
13     end
14 end
```

Listing 4: main.lua (Final)

```
01 require "scenery"
02 require "pworld"
03 require "pobject"
04
05 w_width = 900
06 w_height = 600
07
08 function love.load ()
09     love.window.setMode (w_width, w_height, {resizable =
10         false})
11     love.graphics.setBackgroundColor (0.5, 0.3, 1, 1)
12     terrainG = Scenery
13     terrainG:init ()
14
15     terrainP = Earth
16     terrainP:init (terrainG)
17
18     object = Box
19     object:init (460, 50, 50, 0.5, 0.2)
20 end
21
22 function love.update (dt)
23     world:update(dt)
24 end
25
26 function love.draw ()
27     terrainG:draw ()
28     object:draw()
29 end
```

By multiplying gravity's rate of acceleration by the `setMeter` value, you will achieve a natural-looking fall for your objects.

The third argument, is the `sleep` argument. If it is set to `true`, it means that objects that are not moving or being interacted with are allowed to `sleep`. The interpreter will not waste cycles on them until something collides with them and they start moving again.

In Listing 3, I have also separated the world and ground configuration from the rest of the code, just to keep stuff tidy.

Listing 4 shows `main.lua`, from which you call all the rest of the files and their components.

As you can see on line 1 of Listing 4, I have also separated the graphical component of the terrain into its own file (Listing 5) and now access its attributes and modules using Lua's object-like calls.

On line 12 of `main.lua` (Listing 4), you create a `Scenery` object called `terrainG`, and you call its initiation function on line 13. This does all the calculating of random levels, defining the polygon's shape, and triangulating (Listing 5, lines 5 to 26) I talked about in my first, standalone example.

Back in Listing 4, on line 15 you create another object, `terrainP` ("P" for "Physical"), which will be the physical representation of the terrain. On line 16, you pass the graphical terrain object `terrainG` to the `terrainP`'s `init ()` function.

To see what `init()` does, turn to Listing 3 (`world.lua`). As with the fill algorithm I mentioned above, LÖVE's physics engine has problems with concave bodies, so what you take from `terrain` is its `groundT` attribute, as this contains all the triangular shapes you calculated on line 25 of Listing 5.

As you can extract the number of items in a Lua table using the `#` operator, it is simply a matter of iterating the triangles (lines 8 to 13 in Listing 3) and creating a corresponding physical body for each.

To make a LÖVE physics body (and the terrain is a body), you need to register it in `world` (Listing 3, line 10). The second two arguments are its relative initial placement. As you are "drawing" the physical triangles that make up the physical ground relative to the upper left-hand corner of the playing field, use `0, 0`. The `'static'` argument means that the objects will not move and are as if stuck to the world.

The next step is to define the shape of the object. You do that with the `newPolygonShape ()` function (Listing 3, line 11). This function takes a list of vertices that, in this case, you can get from the graphical ground element you define on line 25 of Listing 5.

A `fixture` (Listing 3, line 12) is what actually attaches the shape to the body and can also be used to define more qualities of a body, such as its density, bounciness, or friction. You will be playing around with those later, when I talk about moving bodies. For the ground, you only need the body's shape.

Listing 5: scenery.lua

```
01 math.randomseed(os.time())
02
03 Scenery = {}
04
05 function Scenery:init ()
06     level101 = w_height - 60 -- (math.random (10, (w_height / 3)))
07     level102 = w_height - 120 -- (math.random (10, (w_height / 3)))
08
09     if level101 < level102
10     then
11         mountain = level101 - 80 --(math.random (10, (w_height / 3)))
12     else
13         mountain = level102 - 80 --(math.random (10, (w_height / 3)))
14     end
15
16     self.ground = { 0, w_height,
17                   0, level101,
18                   w_width / 3, level101,
19                   w_width / 2, mountain,
20                   w_width * (2 / 3), level102,
21                   w_width, level102,
22                   w_width, w_height
23                 }
24
25     self.groundT = love.math.triangulate (self.ground)
26 end
27
28 function Scenery:draw ()
29     love.graphics.setColor (0, 0, 0, 1)
30     love.graphics.polygon ('line', self.ground)
31
32     love.graphics.setColor (0.5, 0.3, 0, 1)
33     for i=1, #self.groundT do
34         love.graphics.polygon ('fill', self.groundT[i])
35     end
36 end
```

And that's it: Once the loop runs through all the triangles in `groundT`, you will have an equivalent set of invisible, but physical triangles overlaying the ones you can see on the screen.

Now let's make a body that will interact with the ground.

Drop Box

Listing 6 defines a square box that falls onto the mountain and then bounces and slides until it stops (or slips off the edge of the world).

The `init ()` function is very similar to that of the physical ground shown in Listing 3: You register the body into the world (line 4, Listing 6), except that, in this case, the body is `'dynamic'` because it moves around; then you define its shape (a square) on line 5; and attach the shape to the body with the `newFixture ()` function (line 6).

What is new is that you define two new qualities of the body: `setRestitution ()` establishes how

Listing 6: pobject.lua

```

01 Box = {}
02
03 function Box:init (posx, posy, size, bounciness, friction)
04     self.body = love.physics.newBody (world, posx, posy, 'dynamic')
05     self.shape = love.physics.newRectangleShape (size, size)
06     self.fixture = love.physics.newFixture (self.body, self.shape, 1)
07     self.fixture:setRestitution (bounciness)
08     self.fixture:setFriction (friction)
09 end
10
11 function Box:draw ()
12     love.graphics.setColor (0.76, 0.18, 0.05)
13     love.graphics.polygon ('fill', self.body:getWorldPoints (self.
14         shape:getPoints ()))
15     love.graphics.print ('Friction: ' .. string.format ("%2f", self.
16         fixture:getFriction ()), 10, 10, 0, 2)
17     love.graphics.print ('Bouncy: ' .. string.format ("%2f", self.
18         fixture:getRestitution ()), 10, 40, 0, 2)
19 end

```

bouncy an object is. It takes a number between 0 (no bounce) and 1 (very bouncy indeed, so much so that if you drop an object with a restitution of 1 onto a flat surface, it will bounce up to its original height again and never stop bouncing).

`setFriction ()` is equally straightforward: 1 is total friction, no slipperiness at all (an object will stop in its tracks immediately); and 0 is no friction, so total slipperiness and the object will slip and slide into infinity.

The `Box` table/class comes with another module, `draw ()` (lines 11 to 17), which draws the object to the playing field. As drawn, rectangles have perfectly horizontal and vertical sides, you have to draw a polygon (line 13) if you want your object to spin realistically when hitting the ground.

To find out the location of the vertices of the polygon as it spins, you need the physical object's

`getPoints ()` function. This function returns the position of a body's vertices relative to its parent (a body can be part of a multi-body object). To transform those coordinates into *world* coordinates (the coordinates that will establish the body's vertices in the playing field), you need to use the `getWorldPoints ()` function. That is what happens on line 13.

Finally, on lines 15 and 16, I print out the values of the body's friction and bounciness for informational purposes. The `string.format ()` is part of Lua's standard arsenal of modules and formats the values so they show only up to two decimal places; Lua uses `..` to concatenate strings. After the string you want to print, you tell LÖVE the coordinates of where you want to print it, the rotation (in radians), and the size multiplier.

Running and Falling

Getting back to `main.lua` (Listing 4), I will use the special variable `dt` in `world:update ()` (line 23) to calculate the position of all moving parts depending on the time that has passed since the beginning of the execution.

As for drawing the results (lines 26 to 29), it is simply a matter of calling the `draw ()` functions of `terrain` (line 27) and the `box` (line 28).

Conclusion

If you're using ZeroBrane Studio [4] to edit your LÖVE code (and if you aren't, you should), you will be able to run the simulation directly from the editor by pressing the *Start* button in the toolbar.

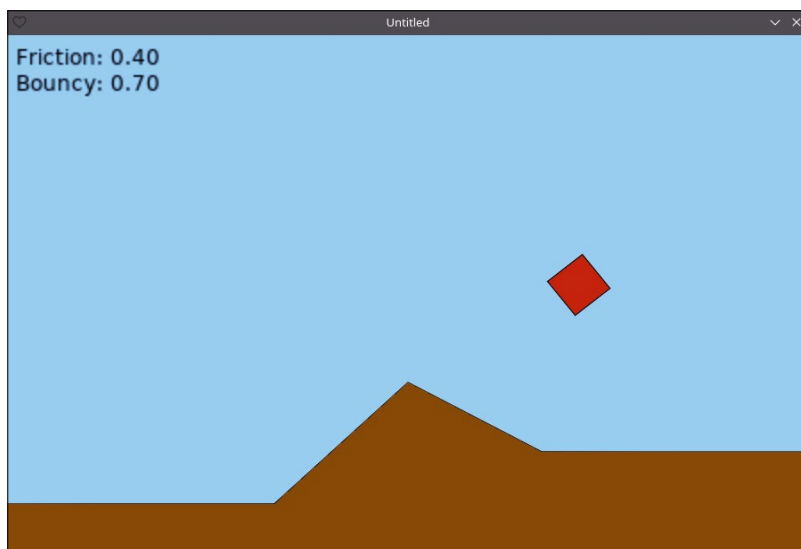
You can also run the program from the command line with:

```
love /path/to/project/directory
```

Either way, you'll see something like Figure 5: a box falling from the sky and bouncing against the ground.

Change line 19 in `main.lua` (Listing 4) to modify the initial position, bounciness, and friction of the box and add features to your body in `pobject.lua` (Listing 6). Add more objects, create particles, and change the components of the forces. In summary, enjoy your new physics sandbox! ■■■

Figure 5: A box falls from the sky and bounces against the ground – physics!



Info

- [1] "Tutorial – LÖVE Animation" by Paul Brown, *Linux Magazine*, issue 234, May 2020, pp. 88-93, <https://www.linux-magazine.com/Issues/2020/234>
- [2] LÖVE: <https://love2d.org>
- [3] The Bouncy Square project: <https://gitlab.com/linux-magazine/235>
- [4] ZeroBrane Studio: <https://studio.zerobrane.com/>

LINUX NEWSSTAND

Order online:
<https://bit.ly/Linux-Newsstand>

Linux Magazine is your guide to the world of Linux. Monthly issues are packed with advanced technical articles and tutorials you won't find anywhere else. Explore our full catalog of back issues for specific topics or to complete your collection.



#234/May 2020

Edge Computing

The Edge is a popular buzz word in high-tech news, but what does it mean really? We introduce you to an exciting new technology that could be changing the way we think about the cloud.

On the DVD: Manjaro 19.02 Gnome Edition and SystemRescueCd 6.1



#233/April 2020

Stream to Your TV

The line between computers and television blurred long ago, but the new tools and new ideas keep coming. This month we highlight some innovative apps for multimedia in Linux, including Gnome Cast for TV, and the easy-to-use Serviio media server.

On the DVD: The Complete Raspberry Pi Geek Archive



#232/March 2020

Stop Ads

Browser-based ad-blockers are useful for controlling many types of pop-ups and banners, but they are less effective with ads built into applications. We look at a couple of alternative tools for blocking ads at the network level: Pi-hole and Privoxy.

On the DVD: GParted 1.0.0 and Kali Linux 2019.4



#231/February 2020

Tiling

Are tiling window managers still relevant for today's desktop? We explore the possibilities of the tiling paradigm with a modern Linux built for tiling.

On the DVD: MX Linux MX-19 and Zorin OS 15 Core



#230/January 2020

Automation Tricks

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

On the DVD: Fedora Workstation 31 and Ubuntu "Eoan Ermine" Desktop 19.10



#229/December 2019

The Future of Vector Graphics

Vector graphics applications like Inkscape are a popular option for creating scalable graphics. Could these tools be even better? As is often the case, the science is out in front of the mainstream. We show you some innovations that could revolutionize tomorrow's graphics apps.

On the DVD: Arch Linux 2019.10.01 and CentOS 8.0.1905

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.



NOTICE

Be sure to check the event website before booking any travel, as many events are being canceled or converted to virtual events due to the effects of COVID-19.

SUSECON Digital 2020

Date: May 20, 2020

Location: From the Web or Your Mobile Device

Website: <https://www.susecon.com/>

Lead your company into a digital transformation by learning how to apply reliable, secure, open source solutions from technical experts, ecosystem partners, and your peers from around the world. At SUSECON Digital 2020, you will learn the latest developments in Enterprise-class Linux, Ceph storage, Kubernetes, Cloud Foundry, and other open source projects.

DeveloperWeek Global 2020

Date: June 16–17, 2020

Location: Global

Website: <https://www.developerweek.com/global/>

DeveloperWeek Global 2020 is the world's largest virtual developer & engineering conference, where thousands of participants from across the globe converge online to discover, learn, and support each other. DeveloperWeek Global is hosted by DevNetwork, the leading developer event organization and global developer community.

Events

Open Source 101 At Home	May 12	Everywhere	https://opensource101.com/events/at-home/
Linux Presentation Day 2020	May 16	Cities across Europe	https://l-p-d.org/en/start
DrupalCon Minneapolis 2020	May 18	Minneapolis, Minnesota	https://events.drupal.org/minneapolis2020
SUSECON Digital 2020	May 20	Everywhere	https://www.susecon.com/
DeveloperWeek Global	June 16-17	Everywhere	https://www.developerweek.com/global/
stackconf Online	June 17-18	Everywhere	https://stackconf.eu/
Embedded Linux Conference North America	June 22-24	Austin, Texas	https://events.linuxfoundation.org/embedded-linux-conference-north-america/
ISC High Performance 2020	June 21-25	Frankfurt, Germany	http://ct.isc-events.com/click.php?id=138
Linux Security Summit North America	June 24-26	Austin, Texas	https://events.linuxfoundation.org/linux-security-summit-north-america/
Cloud Foundry Summit	June 25	Austin, Texas	https://www.cloudfoundry.org/events/summit/austin-2020/
KubeCon + CloudNativeCon Europe	August 13-18	Amsterdam, Netherlands	https://events.linuxfoundation.org/kubecon-cloudnativecon-europe/
DevOpsCon	Aug 31 - Sep 3	London, UK	https://devopscon.io/london/
Open Source Summit Japan	September 15-16	Tokyo, Japan	https://events.linuxfoundation.org/open-source-summit-japan/
Storage Developer Conference	September 21-24	Santa Clara, California	https://www.snia.org/events/storage-developer
Open Source Summit Japan	September 15-16	Tokyo, Japan	https://events.linuxfoundation.org/open-source-summit-japan/

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.

Contact Info

Editor in Chief

Joe Casad, jcasad@linux-magazine.com

Copy Editors

Amy Pettle, 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, Lori White

Cover Image

©donnasterns, 123RF.com

Photo of Linus Torvalds by Krd / CC BY-SA

(<https://creativecommons.org/licenses/by-sa/3.0>)

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, Zieblandstr. 1, 80799 Munich, Germany.

Authors

Erik Bärwaldt	28
Chris Binnie	56
Paul Brown	90
Zack Brown	11
Bruce Byfield	32, 50
Joe Casad	3
Mark Crutch	69
Marco Fioretti	36, 75
Jon maddog Hall	70
Charly Kühnast	49
Christoph Langner	52, 66
Vincent Mealing	69
Pete Metcalfe	62
Graham Morrison	84
Dmitri Popov	72
Mike Schilli	44
Mayank Sharma	14, 22
Jack Wallen	8, 22

Issue 236 / July 2020

Approximate

UK / Europe Jun 06

USA / Canada Jul 03

Australia Aug 03

On Sale Date

Software-Defined Radio

Radio used to be one of the most universally recognized electronic hardware products, but now even the iconic radio has gone virtual. Next month, we explore the new world of software-defined radio – open source style.

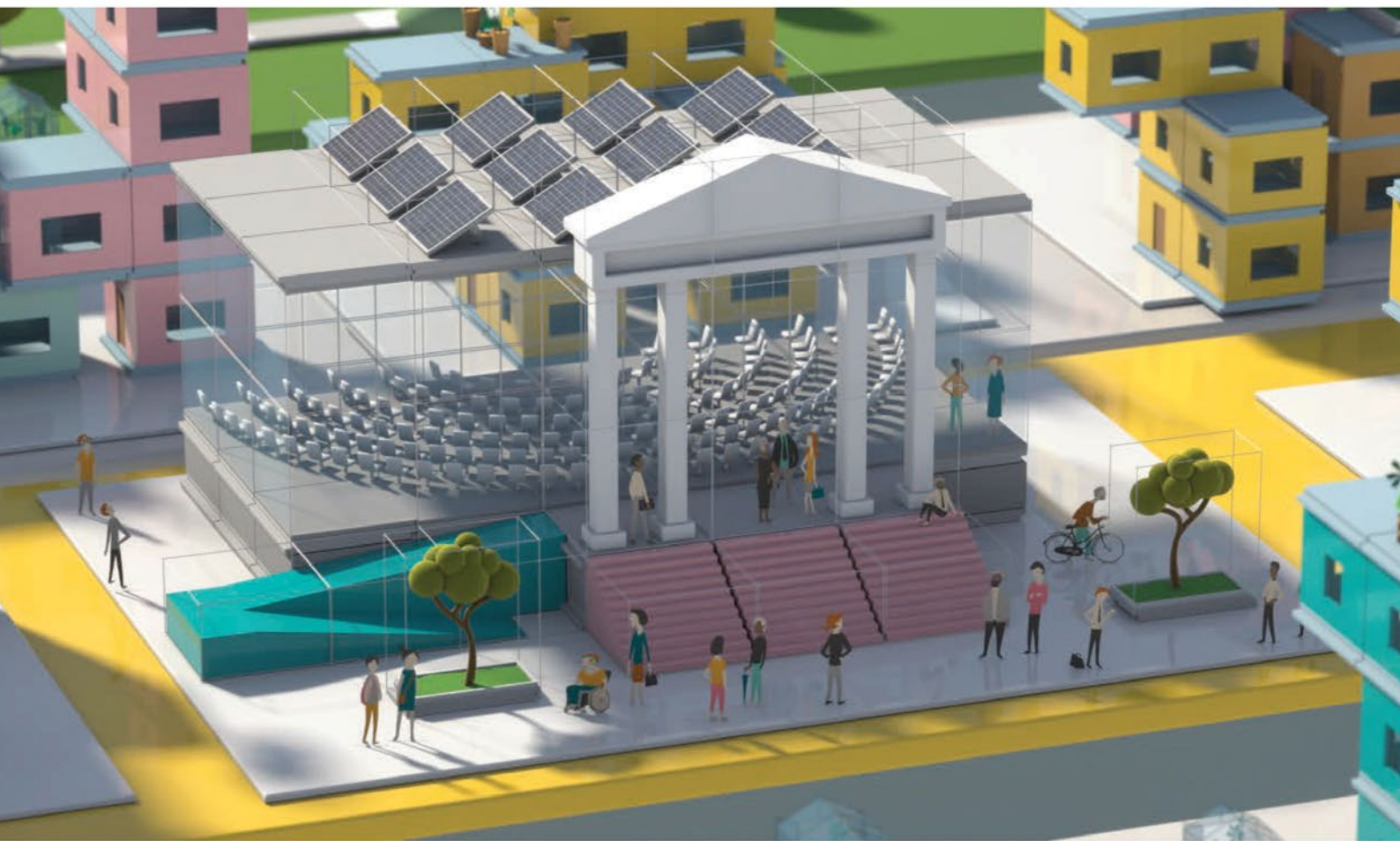
Preview Newsletter

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>

Public Money

Public Code



Modernising Public Infrastructure
with Free Software

HETZNER

NEW: DEDICATED ROOT SERVER DX181



EPYC PERFORMANCE FOR DEMANDING WORKLOADS

IDEAL FOR VIRTUALIZATION AND
HIGH-PERFORMANCE COMPUTING

Configure it with up to 8 x SATA SSD
or 10 x NVMe SSD and 512 GB DDR4 ECC RAM

Dedicated Root Server DX181

- ✓ AMD EPYC 7502P 32 Core "Rome"
Simultaneous Multithreading
- ✓ 128 GB DDR4 ECC RAM
- ✓ Individual storage capacity (at additional cost)
- ✓ 100 GB Backup Space
- ✓ SATA RAID Controller
- ✓ Traffic unlimited
- ✓ Location Germany
- ✓ No minimum contract
- ✓ Setup Fee \$207

monthly from **\$207**

All prices exclude VAT and are subject to the terms and conditions of
Hetzner Online GmbH. Prices are subject to change. All rights reserved by
the respective manufacturers.

www.hetzner.com