

FREE DVD

CentOS 7
(1811) 64-bit

REGULAR EXPRESSIONS

NOW FEATURING
LINUXVOICE

LINUX
MAGAZINE



LINUX

PRO

MAGAZINE

JUNE 2017

A NEW APPROACH TO

REGULAR EXPRESSIONS

Regex meets natural English with Simple Regex Language

Put a GUI on systemd

Discreete Linux

Keep away from intruders and data collectors

Alexa Tricks

Get geeky with your Amazon Echo

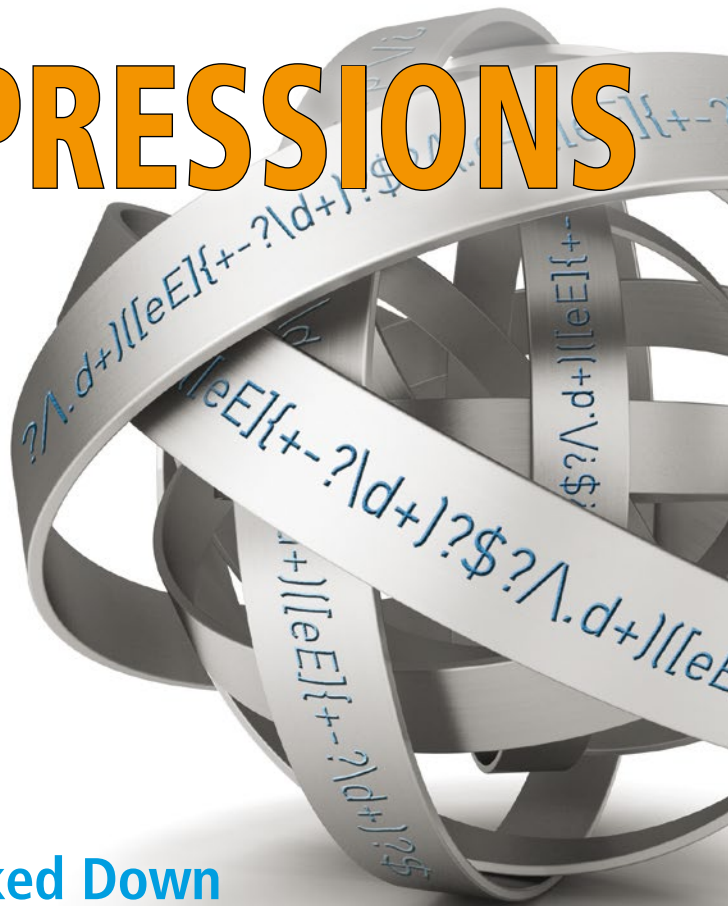


Chakra Linux

Elegant and simple Linux built for KDE

Locked Down

What you need to know about container security



LINUXVOICE

- The art of free software advocacy
- TensorFlow: Free library for your AI creations
- Working with network sniffers



FOSS Picks

- Sonic Visualiser 3
- Starfighter 1.7

Tutorials

- Server Security
- TkInter

Issue 199
June 2017
US\$ 15.99
CAN\$ 17.99



HETZNER
ONLINE

LIGHTNING QUICK

RAZOR SHARP



Dedicated Root Server EX51-SSD-GPU

Intel® Core™ i7-6700
Quad-Core Skylake Processor
64 GB DDR4 RAM
2 x 500 GB SATA 6 Gb/s SSD
GeForce® GTX 1080 graphics card
100 GB Backup Space
50 TB traffic inclusive*
No minimum contract
Setup Fee \$105



monthly \$ **105**

The ideal solution for resource-intensive calculations.

The new dedicated root server EX51-SSD-GPU houses an incredibly powerful graphics card, the top-of-the-line GeForce® GTX 1080, which accelerates all sorts of graphics and video processing, such as 3D rendering.

www.hetzner.de/us

* There are no charges for overage. We will permanently restrict the connection speed if more than 50 TB/month are used. Optionally, the limit can be permanently cancelled by committing to pay \$1.30 per additional TB used.

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. Intel, Intel Logo, Intel Core and Core Inside are brands of the Intel Corporation in the USA or other countries.

TIP TALK

Dear Reader,

The New York City Taxi and Limousine commission announced a proposal that would require ride-hailing services such as Uber to include a space in the app for riders to leave a tip for the driver. The proposal was the work of the Independent Drivers Guild, an organization representing drivers from Uber and other ride-hailing vendors. The drivers want what coffee shop baristas and drivers for other services such as Lyft already have: an easy way for credit card customers to leave a tip.

Uber drivers haven't been faring so well recently. The Uber website says as a driver you can "earn as much as you want," but that's not what the drivers say. Some recent studies show the net income for an Uber driver at around US \$11 per hour – not even a living wage in many cities. Like Walmart, McDonald's, and other infamous big corporations with low wages, Uber is quite content to accept the labor of full-time employees without giving them enough back to buy food and pay their bills. The New York City Uber drivers make a little more (more like \$20 per hour), but then, they have to pay for New York housing, which is around the most expensive in the US.

The absence of a tipping feature in the Uber app is quite curious. The company's website puts it this way:

"The Uber app does not include a tip when billing you for a trip fare. In most cities, Uber is a cashless experience. Tipping is voluntary. As a rider, you are not obligated to offer your driver a gratuity in cash. If you decide you would like to tip, your driver is welcome to accept."

If you wanted Uber to be a cashless experience, it seems like you would *want* to include a tipping feature in the cashless mobile phone app. Otherwise, you force users who want to tip to bring cash. And why go to the trouble of intentionally telling people they are "not obligated" to tip? If Uber drivers are so underpaid, why wouldn't Uber love an opportunity for their drivers to get more money without requiring additional direct compensation?

That question gets down deep into Uber's business plan. According to Uber CEO Travis Kalanick, "Uber doesn't grow if car ownership is cheaper than taking Uber." In other words, although it looks like a high-tech taxi service, Uber is not really even trying to compete with the cost of a cab but is actually competing with the cost of *car ownership*. Their goal is to cost less than the cost of owning a car, which

forces Uber to keep its cost down to way, way less than the price of riding around all year in taxis.

Encouraging tips adds to the total cost of using Uber, which, ultimately makes Uber less competitive with car ownership, even though they might still be well under the cost of a taxi.

How are they doing with that strategy? Not well, at least by old-fashioned standards of profitability. According to *Wired*, Uber lost a total of around US \$3 billion in 2016. Uber fares reportedly cover only 41 percent of the total cost of the ride, with the rest of the cost subsidized directly by the company.

How long can they continue to lose money at this rate? Uber has been raising a lot of cash recently, and they reportedly had around \$15 billion on hand last year, so they could keep losing \$3 billion a year for another 4 to 5 years in order to wait for something to happen that hasn't happened yet. For instance, they could perfect their effort to employ self-driving cars, so they don't have to pay their drivers (although they would have the overhead of maintaining and insuring the cars). Or, they could out-last the competition and raise their prices, although higher prices would take them farther from their stated goal of competing with the cost of car ownership.

If Uber really could replace the role of car ownership in the lives of people around the world, as Kalanick suggested, maybe they wouldn't need to raise their prices. A huge increase in market share, such as a mass exodus of millions of people abandoning car ownership and depending on Uber, would spread the overhead around through more rides, thus potentially putting the company in the black.

Uber is thus a gigantic bet on the transformative power of new technology. Its success depends on the world changing in either (or both) of the following ways:

- cars without drivers
- people giving up their cars and living by ride sharing

Note that people giving up their cars and living by mass transit won't do the trick. Uber needs you, and millions like you, to continue to ride in cars – just Uber's cars, not your cars.

Will this work for them? It sure seems like a lot of moving parts will have to line up for Uber to break even. Some high-tech companies have succeeded with this sort of bold strategy in the past, but for now, just tip the driver if you can.

INFO

- [1] Uber Tipping: First NYC, then the Country?
<https://www.usatoday.com/story/tech/news/2017/04/17/new-york-city-uber-tipping-in-app-tip-ride-hailing/100564850/>
- [2] Uber's 2016 Losses to top \$3Billion According to Leaked Financials": <http://www.wired.co.uk/article/uber-finances-losses-driverless-cars>

Joe

Joe Casad,
Editor in Chief



LINUX MAGAZINE

WHAT'S INSIDE

Tired of composing and deciphering those weird, inscrutable strings of characters known as regular expressions? This month we show you Simple Regex Language (SRL), an innovative tool that lets you create regular expressions using simple English commands.

Other highlights within:

- **Systemd Graphic Tools** – You don't need to be a command-line junkie to keep an eye on the systemd service manager. We show you some graphic front ends that offer a friendlier view (page 26).
- **Amazon Echo's Alexa** – This month's Programming Snapshot shows how you can teach the Alexa assistant that lives in your Amazon Echo some new tricks (page 46).

You'll find more great articles in our Linux Voice section, including a practical lesson on network sniffers and a look at the free TensorFlow library for artificial intelligence applications.

SERVICE

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

NEWS

08 News

- Microsoft is shutting down CodePlex
- Fedora 26 Alpha released
- Old Linux kernel bug discovered
- OpenSSH 7.5 released
- Ubuntu Touch not dead
- FBI refuses to release the tool used to hack terrorist's iPhone

12 Kernel News

- Peer-to-peer memory devices
- Adding a `printk()` kernel thread
- EOL for AVR32 architecture
- Making system calls reject unknown flags

REVIEWS

20 Exploring the Min Web Browser

A simple design, efficient performance, and a built-in ad blocker warrant a closer look at this web browser built with the Electron framework.



COVER STORY

16 Simple Regex Language

Regular expressions are a powerful tool, but they can also be very hard to digest. Simple Regex Language lets you write regular expressions in natural language.

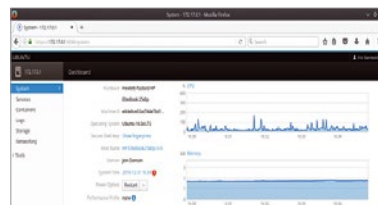


24 Discrete Linux

Fly under the radar of hackers and data collectors with Discrete Linux.

26 Systemd Graphical Tools

Systemd has several tools that offer services just a mouse click away. We look at six graphic apps for systemd.



IN-DEPTH

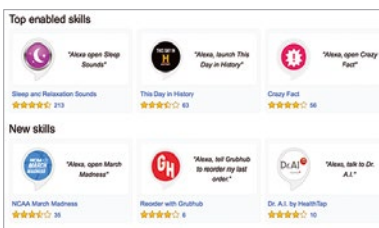
- 34 Professor Knopper's Lab**
An experiment in removing systemd.
- 38 Chakra Linux**
KDE lovers can rejoice at Chakra Linux's beautiful and functional operating system.



- 42 Container Security**
A recent flurry of activity in the container space raises a number of interesting questions about security in the enterprise environment.



- 46 Programming Snapshot – Alexa**
Asking Alexa only for built-in functions like the weather report gets old quickly, but with a few lines of code, you can teach this digital pet some new tricks.



- 52 Charly – XMLStarlet**
Charly and XML have never been best friends; however, it was vital that he have an excellent indoor climate, so he plucked up his courage and considered XMLStarlet.
- 54 Command Line – WordGrinder**
WordGrinder offers distraction-free writing; we look at how realistic that concept is in everyday use.

- 58 Gnome Flashback**
Lamented by many as dead and gone, Gnome 3 fallback is still alive and kicking in Gnome Flashback.
- 62 Core Infrastructure Initiative**
How does the Core Infrastructure Initiative fare three years in?



LINUXVOICE

- 65 Welcome**
This month in Linux Voice.
- 66 News with Simon Phipps**
OSI approval guarantees the freedom to innovate.
- 67 Computing's Golden Age**
Everything is awesome – the past is a foreign country.
- 68 Doghouse – Legacies**
"maddog" ponders how our actions affect others in ways we cannot predict.
- 70 FOSS Advocacy**
Learn the tricks, tips, and techniques for converting friends, family, and colleagues to free and open source software.
- 74 FAQ – TensorFlow**
Welcome our new artificial intelligence overlords by tinkering with their gray matter.
- 76 Core Tech – Network Sniffers**
Learn what's going on in your network using Linux and its arsenal of packet capture tools.
- 82 FOSS Picks**
Sonic Visualiser 3, Latte Dock 0.5.91, Tizonia 0.7.0, HFS+ Rescue 3.3, Project: Starfighter 1.7, and more!
- 88 Tutorial – Server Security**
Fear not the barbarians of cyberspace, and follow our guide to shoring up your digital defenses.
- 92 Tutorial – TkInter**
Expand your Python knowledge and write GUI apps with a smattering of code, thanks to the TkInter toolkit.



On the DVD



CentOS 7
(1611) 64-bit



Linux Mint 18.1
"Cinnamon" 32-bit

TWO TERRIFIC DISTROS
DOUBLE-SIDED DVD!

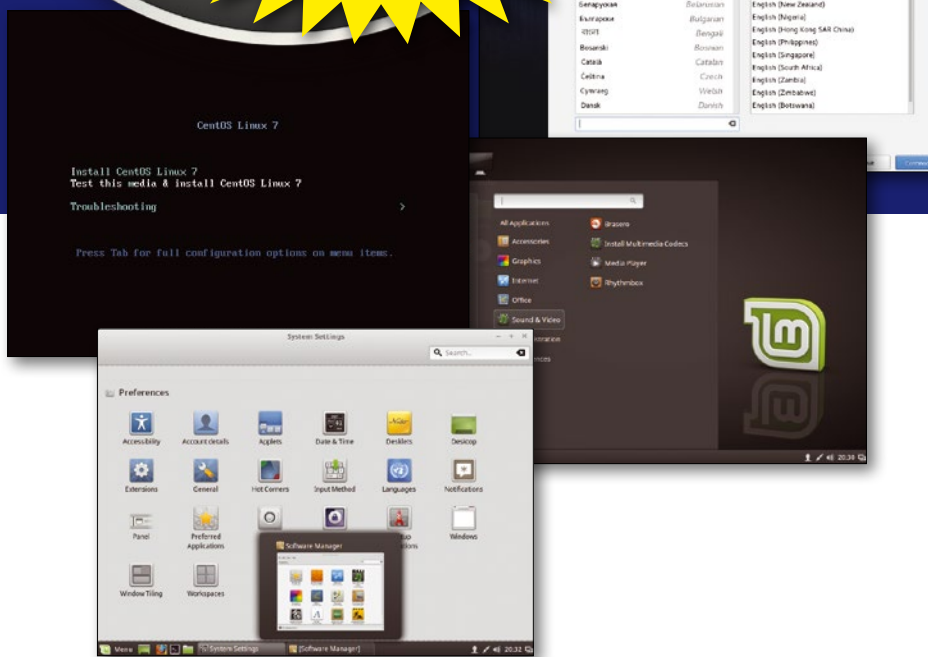
CentOS 7 Release 1611 (64-bit Install)

CentOS derives from Red Hat Enterprise Linux (RHEL) and features a two-year release cycle, a six-month update cycle, and a 10-year security maintenance cycle, resulting in a "secure, low-maintenance, reliable, predictable, and reproducible Linux environment" [1]. Release 1611 now supports seventh-generation Core i3, i5, and i7 Intel processors and the I2C on sixth-generation Core processors. Other major changes include SHA2 support in OpenLDAP, Bluetooth LE, and updated storage, network, and graphics drivers. On booting the DVD, you have the choice of installing a minimal operating system; an environment with the Gnome or KDE Plasma desktop; a development and creative workstation; an infrastructure, web, or file and print server; a virtualization host; or a server with a GUI [2] [3].

Linux Mint 18.1 Cinnamon (32-bit Live)

Linux Mint 18.1 "Serena" Cinnamon long-term support (LTS) release will be maintained until 2021. The updated software incorporates refinements and adds new features [4] [5], including:

- Cinnamon 3.2
- Qt 5.7+ support
- Redesigned, faster screensaver
- Battery power indicator and media controls on lock screen
- Synaptics, libinput touchpad, and iio-sensor-proxy accelerometer support
- Vertical panels on desktop



ADDITIONAL RESOURCES

- [1] CentOS 7: <https://wiki.centos.org/FrontPage>
- [2] CentOS 7 release notes: <https://wiki.centos.org/Manuals/ReleaseNotes/CentOS7>
- [3] CentOS 7 tips and tricks: <https://wiki.centos.org/TipsAndTricks>
- [4] Linux Mint: <https://www.linuxmint.com>
- [5] New features in Linux Mint 18.1 Cinnamon: https://www.linuxmint.com/rel_serena_cinnamon_whatsnew.php

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

COMPLETE YOUR LIBRARY

Order a digital archive bundle and save at least **50% off** the digisub rate!



ORDER YOURS TODAY!
shop.linuxnewmedia.com



You get an **entire year of your favorite magazines** in PDF format that you can access at any time from any device!

NEWS

Updates on technologies, trends, and tools

THIS MONTH'S NEWS

- 08 Farewell to CodePlex**
- Microsoft is Shutting Down CodePlex
 - Fedora 26 Alpha Released

- 09 The Future of Ubuntu Touch**
- Old Linux Kernel Bug Discovered
 - OpenSSH 7.5 Released
 - Ubuntu Touch Not Dead
 - More Online

- 10 iPhone 5c Flaw Withheld**
- FBI Refuses to Release the Tool Used to Hack Terrorist's iPhone

Microsoft is Shutting Down CodePlex

Microsoft has announced that it is shutting down its open source code hosting platform CodePlex, which allowed developers to host and share the source code of open source software. Microsoft created the site in 2006.

Microsoft is not the only vendor that has shut down an open source code hosting platform. In 2015, Google shut down Google Code.

Linus Torvald's Git version control system is the reason behind the demise of Google Code and CodePlex. In a blog post, Microsoft engineer Brian Harry wrote, "Over the years, we've seen a lot of amazing options come and go but at this point, GitHub is the de facto place for open source sharing and most open source projects have migrated there."

Even Google and Microsoft are now using the Git-based GitHub to host their open source code. "As many of you know, Microsoft has invested in Visual Studio Team Services as our 'One Engineering System' for proprietary projects, and we've exposed many of our key open source projects on GitHub (Visual Studio Code, TypeScript, .NET, the Cognitive Toolkit, and more). In fact, our GitHub organization now has more than 16,000 open source contributors – more than any other organization – and we're proud to partner closely with GitHub to promote open source."

Microsoft has disabled the ability to create new CodePlex projects. In October, it will be set to read-only, and by December 2017, plugs will be pulled on the service, bringing an end to an era.



Fedora 26 Alpha Released

The Fedora community has announced the alpha release of Fedora 26. According to the release schedule, Fedora 26 is scheduled to be released on June 27, 2017, but unlike Ubuntu, Fedora is not firm with release dates, and they are known for delaying releases if things are not ready.

The alpha is not meant for production usage; it's meant for testing and filing bug reports.

Ryan Lerch wrote in *Fedora Magazine*, "Fedora Alpha releases are provided for Fedora users to try out the upcoming release. More importantly, Fedora engineers want you to file bugs against the upcoming release. The Fedora 26 Changeset page on the Fedora wiki provides a list of new features provided in Fedora 26."

Fedora 26 comes with newer versions of packages, including Gnome 3.24 and DNF 2.0. You can see the list of packages included in Fedora 26 online (<https://fedoraproject.org/wiki/Releases/26/ChangeSet>).

You can download and test Fedora 26 Alpha from the official download page.

Old Linux Kernel Bug Discovered

Alexander Popov, one of the winners of the 2016 Linux Foundation Training (LiFT) scholarship, has discovered a very old bug in the Linux kernel that can affect modern systems.

Popov wrote on a mailing list, "This is an announcement of CVE-2017-2636, which is a race condition in the `n_hdlc` Linux kernel driver (`drivers/tty/n_hdlc.c`). It can be exploited to gain a local privilege escalation. This driver provides HDLC serial line discipline and comes as a kernel module in many Linux distributions, which have `CONFIG_N_HDLC=m` in the kernel config."

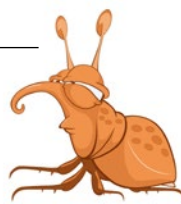
Popov fixed the issue by using a "standard kernel linked list protected by a spinlock and got rid of `n_hdlc.tbuf`. In the case of transmission error, the current data buffer is put after the head of `tx_buf_list`," he wrote on the mailing list.

The issue is affecting major distributions, including Red Hat Enterprise Linux (RHEL). According to a Red Hat Bugzilla submission, although RHEL 5 is unaffected, the bug does affect the Linux kernel packages shipped with RHEL 6 and 7 and Red Hat Enterprise MRG 2. Because this issue is rated important, it has been scheduled to be fixed in future updates for the respective releases. Canonical has already released a patch; SUSE is working on it.

The bug is old, and the module is used in really old hardware; even if the module is shipped with modern Linux distributions, it's never loaded by default. However, the module is automatically loaded "if an unprivileged user opens a pseudoterminal and calls `TIOCSETD ioctl` for it setting `N_HDLC` line discipline," explained Popov.

One might wonder why users should worry about it. Steven J. Vaughan-Nichols explained, because "it's easy to do, which means it's easy for a local user to exploit. Before poo-pooing this as a non-issue, keep in mind that with hosted and cloud computing, many people have 'local' access to Linux servers."

As always, check your distribution and run updates to patch the flaws.



MORE ONLINE

Linux Magazine

www.linux-magazine.com

Off the Beat • Bruce Byfield

Unity Enters its Twilight

Unity, Ubuntu's default desktop environment, has always been ironically named. Begun after conflicts between Ubuntu and GNOME, Canonical Software and Ubuntu developed it as a solo effort after other projects ignored Mark Shuttleworth's famous challenge to develop an interface to rival Apple's.

New Guard and Old Guard Clash at Free Software Foundation

Does the Free Software Foundation (FSF) have internal conflicts? An answer is hard to find, because people prefer not to talk about the possibility. However, there are increasing indications that the FSF is having trouble adjusting to modern activism – and that part of the problem might be growing differences between FSF supporters and its founder Richard Stallman (RMS).

Free Software Makes Computers Knowable

A high school class mate of mine posted a comic piece about the problems with new software systems. The unspoken assumptions were that computers always caused problems, but that nothing much could be done about the situation. I smiled, but I realized that I no longer shared this attitude – and that free software was the reason why.

Why Universal Packages Aren't Universal Solutions

The initial announcements of Flatpak and Snap presented them as the solution to all of Linux's packaging problems. These claims soon proved to be ahead of actual development, but they linger in the minds of many users.

ADMIN HPC

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

Simple HDF5 in Python and Fortran • Jeff Layton

Using the popular HDF5 I/O library with Python and Fortran.

ADMIN Online

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

Segmenting Networks with VLANs Mathias Hein

Network virtualization takes very different approaches at the software and hardware levels to divide or group network resources into logical units independent of the physical layer. It is typically a matter of implementing secure strategies. We show the technical underpinnings of VLANs.

Bugs © Ludmila_Pantelejenkova, 123RF.com



OpenSSH 7.5 Released

The OpenSSH project has announced the release of version 7.5 that comes with added security features.

"This release deprecates the `sshd_config UsePrivilegeSeparation` option, thereby making privilege separation mandatory. Privilege separation has been on by default for almost 15 years and sandboxing has been on by default for almost the last five," according to the release notes.

Additionally, portable OpenSSH has removed support for building against OpenSSL versions before 1.0.1, because OpenSSL stopped supporting those versions more than 12 months ago (i.e., they no longer receive fixes for security bugs).

According to Softpedia, "Among other changes, OpenSSH 7.5 updates the format of various log messages emitted by the packet code to include details about the user's authentication state, etc." You also can find the entire changelog onsite.

Ubuntu Touch Not Dead

It would seem that, at least in the short and mid-term, there are no plans on behalf of Canonical of producing a new version of Ubuntu Touch, the operating system for the Ubuntu phones put out by companies like bq and Meizu. It is also true that Canonical's hardware associates seem to have no plans to put out any new devices running the system.

One would be excused for thinking that the Ubuntu Touch/Phone project is dead or, at least, suspended, until such a time when the mobile market is more favorable.

However, this would be wrong on two counts. Firstly, it is now clear Ubuntu Touch, a system running on top of an Android base, was not meant to be the final thing for Canonical. Unity 8 running on Mir and a full Ubuntu Linux stack was the end game. Canonical's vision was one system for all devices, and a Ubuntu Touch was a temporary solution. (We review Unity 8 in the current issue of Ubuntu User, and, spoiler alert, it is not all there yet, but it does seem to be on its way).



Secondly, a free software project (and Ubuntu Touch is free software¹) is not dead until there is nobody willing to support it. UBports is a group of independent developers who want to port Ubuntu Touch to as many models of smartphones as possible. Spearheaded by Smoose, a free software company from the Netherlands, and with the blessing from Canonical, the project has made its first target porting Ubuntu Touch to the Fairphone 2. The initiative has proven popular so far and the UBports community grew from 0 to nearly 100 members in its first two months.

The Fairphone is a logical choice for Smoose and UBports since it is consistent with the company's and community's principles. The Fairphone is produced with sustainability and fair trade in mind, and is also fair to the user, since all its components are easily replaced and customizable. Also, the rapid growth in the number of contributing members proves there is a real interest in developing an alternative, community controlled operating system to what is already out there.

Although Fairphone is the main immediate aim, UBports is also working on and improving the ports for the OnePlus One and Two, several Nexus devices, Optimus L90, and several others. This also gives hope to early adopting device owners who may think they were left out in the cold when they discovered no more updates for their handsets were forthcoming from Canonical.

The people of Smoose were at the MWC at the Canonical booth last week showcasing their project. They have also been kind enough to loan us a device for review. We will be talking about this project in depth and trying out the Fairphone with Ubuntu Touch in the next issue of Ubuntu User, on sale in May/June, so you can look forward to that.

Meanwhile, if you are interested in seeing if there is a port of Ubuntu Touch for your phone, or want to contribute, head over to the project's site, join the discussion, and try out some code.

1. Ubuntu Touch's code is mostly distributed under a GPL, with some components distributed under MIT and Apache licenses.

FBI Refuses to Release the Tool Used to Hack Terrorist's iPhone

The FBI has refused to disclose information about the tool it used to hack into the iPhone of San Bernardino shooter Syed Farook.

Initially the FBI asked Apple to create a backdoor so they could access the content of Farook's iPhone. Apple refused to create the backdoor, stating that once there, it can be used over and over.

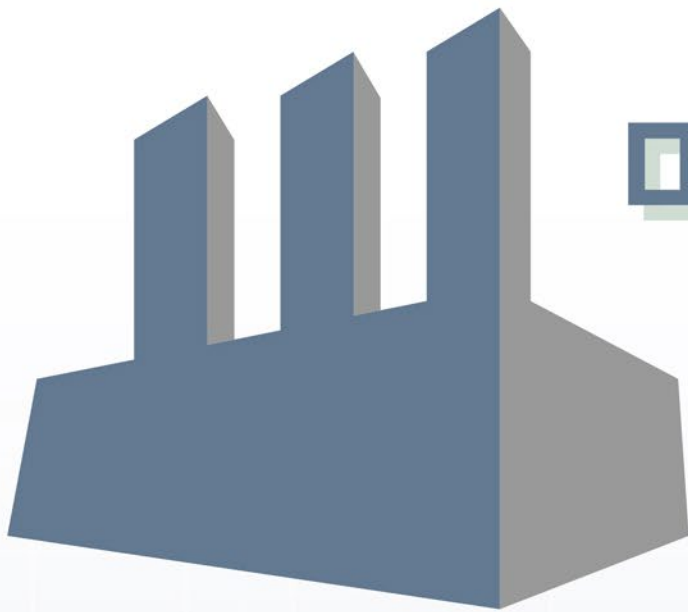
Now it's clear that Apple's concerns were correct. According to ZDNet, "Justice Dept. officials say that details of a hacking tool used to access a terrorist's iPhone should not be released because it may still be 'useful' to federal investigators."

That contradicts the statement by FBI director James Comey where he tried to downplay the scope of the tool. Last year Comey said that the tool affects only the iPhone 5c running iOS 9. Despite initial considerations to share the vulnerability it exploited to unlock the iPhone with Apple, FBI later refused to disclose any such information with the company.

The FBI reportedly wasted more than \$1 million to crack the iPhone in question, even though they did not extract any valuable information from the device. Last year the FBI was sued by three news organizations to disclose more information about the hack.

On March 13, 2017, David Hardy, section chief of the FBI's records management division, said in a court filing, "Disclosure of this information could reasonably be expected to cause serious damage to national security as it would allow hostile entities to discover the current intelligence gathering methods used, as well as the capabilities and limitations of these methods."

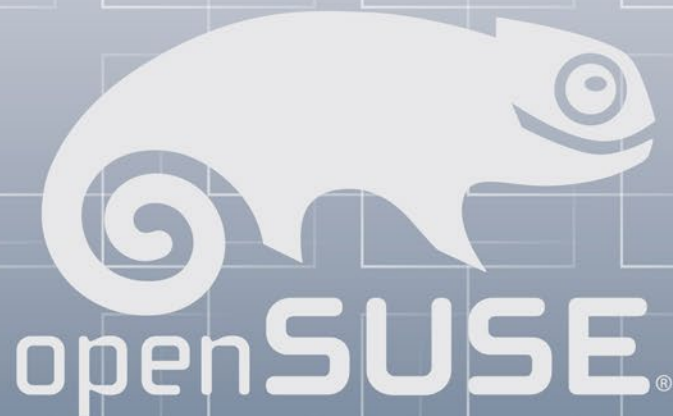
The FBI's refusal to share the flaw with Apple and the public is a double-edged sword. It's not just government agencies exploiting such flaws; there are security organizations whose primary business it is to find such flaws and sell them to criminals and repressive governments. By not disclosing information about the tool, the FBI is apparently putting every iPhone 5c user out there at risk of being hacked.



**open
build
service**

**A generic system to build
and distribute packages
from sources in an automatic,
consistent and reproducible way**

openbuildservice.org



Zack's Kernel News



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

By Zack Brown

ZACK BROWN

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**.

Supporting Peer-to-Peer Memory Devices

Logan Gunthorpe recently posted a patch to support peer-to-peer PCI memory devices. He explained that this is “a PCI card with a BAR space that points to regular memory. This may be an independent PCI card or part of another completely unrelated device (like an IB card or a NVMe card).” A base access register (BAR) is used for addressing memory on the device. Using PCI memory devices as regular system memory may increase the latency between memory transfers as compared with the primary RAM on the system motherboard, but it could come in very handy in cases where the primary RAM starts to be used up.

Logan, however, acknowledged there were caveats to this approach, particularly when dealing with older PCI devices. As a result, he explained, “The code is designed to only utilize the p2pmem device if all the devices involved in a transfer are behind the same PCI switch. Other cases may still work or be desirable for some end users but it was decided this would be the best course of action to prevent users enabling it and wondering why their performance dropped.”

Sinan Kaya had some concerns about the portability of Logan’s patch. He understood the desire to avoid unexplained performance hits, but he felt that requiring all devices to be behind the same switch was essentially arbitrary and should be left as a policy decision to be made by the user. He said, “I’d rather see the feature enabled by default without any assumptions. Using it with a switch is just a use case that you happened to test.”

Logan said that in principle this would be fine, except for the fact that there was a huge pile of devices to consider, some of which would work as peer-to-peer PCI memory devices and some wouldn’t. Trying to identify a complete blacklist would be a big effort. On the other hand, blacklisting everything by default, and white listing only

PCI switches would be easy and simple, since PCI switches were known to work. So, it wasn’t so much a policy decision as a shortcut that made sure the patch worked at all. Logan remarked that the load sharing facility (LSF) folks “were primarily concerned with not having users enable the feature and see breakage or terrible performance.”

Logan said that originally he was in favor of letting users configure which specific PCI devices would be used by his patch, and then they’d be responsible for testing and benchmarking it. But not enough people liked that idea, so he went with the behind-the-same-switch approach.

Sinan saw Logan’s point, but said he was “trying to generalize what you are doing to a little bigger context so that I can use it on another architecture like arm64 where I may or may not have a switch.” So he still wanted to stick with a blacklist instead of a white list; however, instead of painstakingly sifting through the wreckage of human progress, Sinan suggested simply blacklisting any device created before the year 2016. Presto!

Logan asked, “How do you get a manufacturing date for a given device within the kernel? Is this actually something generically available?” Sinan replied that such information was used all over the place in the kernel, “for introducing new functionality while maintaining backwards compatibility.” He suggested checking out the `drivers/pci` and `drivers/acpi` directories in the source tree for examples of SMBIOS calls.

Logan investigated but was only able to confirm that these calls provided the date the firmware was written for a piece of hardware – not the date of manufacture of the hardware itself. He said, “Saying that the system’s firmware has to be written after 2016 seems like an arbitrary restriction that isn’t likely to correlate to any working systems.”

Logan stuck to his original idea: “Allow all switches and then add a white list of root ports that are known to work well. If we care about preventing broken systems

in a comprehensive way then that's the only thing that is going to work."

Sinan agreed that relying only on the firmware date would not be enough. But he explained, "I recommended a combination of blacklist + P2P capability + BIOS date. Not just BIOS date." He went on to propose:

"PCI defines capability registers for discovering features. Unfortunately, there is no direct P2P capability register.

However, Access Control Services (ACS) capability register has flags indicating P2P functionality. P2P feature needs to be discovered from ACS: https://pdos.csail.mit.edu/~sbw/links/ECN_access_control_061011.pdf. This is just one of the many P2P capability flags.

'ACS P2P Request Redirect: must be implemented by Root Ports that support peer-to-peer traffic with other Root Ports5; must be implemented by Switch Downstream Ports.'

If the root port or a switch does not have ACS capability, P2P is not allowed. If these P2P flags are not set, don't allow P2P feature.

The normal expectation from any system (root port/switch) is not to set these bits unless P2P feature is present/working. However, there could be systems in the field with ACS capability but broken HW or broken FW. This is when the BIOS date helps so that you don't break existing systems.

The right thing in my opinion is: 1. Blacklist by pci vendor/device id like any other pci quirk in quirks.c. 2. Require this feature for recent HW/BIOS by checking the BIOS date. 3. Check the P2P capability from ACS."

Sinan added that Logan's idea of only supporting devices behind the same PCI switch couldn't be relied on either. He pointed out that in general, when it came to portability it was important for device drivers to avoid making architecture-specific decisions within their own code. He said, "There are hundreds of device drivers in the kernel. None of them are guaranteed to work in any architecture, but they don't prohibit use either. System integrators like me test these drivers against their own systems, find bugs to remove arch specific assumptions, and post patches."

In the case of Logan's code, Sinan said, the peer-to-peer memory feature was just one of the many users of peer-

to-peer support within Linux. It had to make use of that support in some kind of generic way that would work for other architectures as well.

Logan replied, "I'd agree that the final code for determining P2P capability should belong in the pci code. Or more likely an even more generic interface with struct device that is bus agnostic.

Though, I'd hope that a lot of this could happen later when there are more kernel users actually wanting to use this code.

It's hard to design a generic interface when you only have one user at present."

Logan also pointed out that his patch was not enabled by default, specifically because of the risk of lowering memory performance. The user had to explicitly make the choice, or else we'd see cases where users upgraded their systems, only to discover big unexplained slowdowns.

Sinan replied that this actually made the whole problem much simpler. He hadn't realized the code was disabled by default. In that case, he said, "Push the decision all the way to the user. Let them decide whether they want this feature to work on a root port connected port or under the switch. [...] If you are just worried about performance, the switch recommendation belongs to your particular product tuning guide or a how-to document not into the actual code itself. I think you should get rid of all pci searching business in your code." Logan liked this idea a lot and said it was what he'd originally preferred but had gotten too much opposition from the LSF folks.

At this point the discussion ended, although presumably the LSF folks will have their own response. Personally, I find this type of debate fascinating, because you can never tell when something that seems like a trivial detail will turn out to be the linchpin of some other element of the kernel, with its own set of requirements. Often these days, security seems to be the sudden wrench in the works of a hot new feature, but in the peer-to-peer memory case it seems to be portability. When the LSF folks have their say, another element may turn out to be crucially important.

Adding a Kernel Thread for printk()

Sergey Senozhatsky posted a patch to move `pr_intk()` to its own separate kernel

thread. This would allow the kernel to print messages to the console, while avoiding certain race conditions that might lead to system lockup. He'd been using this patch in production for nearly a year, with only a few issues. The biggest of these was that having a separate `printk()` thread changed the behavior of `printk()` in a couple particular corner cases. In one case, if one CPU on the system is much faster than the CPU running the `printk()` thread, it could result in some `printk()` messages never reaching the console. In the other corner case, if one CPU called a ton of `printk()`s, it could interfere with the other CPUs being able to call `printk()` successfully.

Petr Mladek liked Sergey's patch in general and offered some minor naming suggestions. Peter Zijlstra also offered his ideas, although he said he was not yet convinced that a whole new thread for `printk()` was truly justified. After all, each new thread on the system is something that the scheduler has to cycle through, which slows things down, even if it's just by a small amount.

Sergey agreed in general, but he said he saw no other way to cleanup the lockups and other problems currently associated with `printk()`. Sergey and Petr continued to discuss message passing issues, but it wasn't clear that Peter's objections were satisfied. Eventually the discussion petered out (ouch) without resolution.

Ultimately, there is likely to be resistance from many kernel folks, who don't want to add new kernel threads without very compelling justification. In the case of `printk()`, that justification may not exist. Yes, it's currently necessary for `printk()` users to be aware of when and how they call `printk()`, or else risk triggering a lockup or other problem. But, it is actually possible for those users to be aware of those things and avoid the problem. So, Sergey's patch may represent just a simple convenience and not a real improvement for the kernel. And, if simple convenience were all that was needed to justify implementing new kernel threads, many other kernel developers with their own pet projects would be submitting them as well. So, unless Sergey's code turns out to make things a lot better than they are now, he's likely to encounter trouble getting it into the main source tree.

AVR32 Architecture End Of Life

Hans-Christian Noren Egtvedt pointed out that the AVR32 architecture has not been well-maintained lately in the kernel. And because it shared so many drivers with Atmel ARM system on a chip (SoC) – which was actively maintained – it was starting to be harder to keep those drivers up to date for that architecture. Hans proposed ditching the AVR32 architecture entirely and just taking it out of the kernel.

Both of these architectures come out of the Atmel corporation and are SoC solutions. But as Hans put it, “all AVR32 AP7 SoC processors are end of life from Atmel” and would no longer be supported. Also, he said, the GCC tool chain was stuck at an older version, as it had not received any patches from Atmel in recent times. He suspected there were very few AVR32 users left in the world, if any at all. Better to put it out of its misery than keep it on life support.

Andy Shevchenko also pointed out that the Buildroot distribution had already removed support for AVR32 back in 2015. He approved Hans's patch wholeheartedly. Boris Brezillon also approved the patch and offered up his grateful thanks. Nicolas Ferre volunteered to help clean up any code fallout that resulted from removing the AVR32 architecture. And, Håvard Skinnemoen, who had worked alongside Hans on all of this, offered some additional suggestions of remaining pieces to expunge.

Ultimately, no one objected to getting rid of AVR32, so it does seem destined for the chopping block in the immediate future.

Making System Calls Reject Unknown Flags

Christoph Hellwig was unhappy with the fact that anyone could pass random garbage as input to system calls, without breaking anything. As a result, user code can't test system calls to see if they accept this or that input flag – they accept them all, even if they do nothing with them. He wanted to write a patch that would make system calls reject input flags that weren't actually supported.

Christoph acknowledged that this would be an application binary interface (ABI) change, which was nearly always a no-no for a few important reasons. For

one thing, it made it impossible to “bisect” the kernel development tree in order to hunt down bugs that crossed the version that implemented the ABI-breaking change. More importantly, however, it would break existing user code running out in the world on everybody's system and require that code to be recompiled. This is actually a big no-no because in many cases, the people running the binaries don't have the source code anymore!

So, ABI changes are generally hit with hammers by various folks, including Linus Torvalds. In this case, Linus didn't bother addressing the ABI issue, but objected to Christoph's main point, saying:

*“Probing for flags is why we *could* add things like `O_NOATIME` etc. – exactly because it ‘just worked’ with old kernels, and people could just use the new flags knowing that it was a no-op on old kernels.*

The whole concept of ‘probing for supported features’ is very suspect. It's a bad, bad idea. Don't do it.

What kind of new flag did you even have in mind that would have such broken semantics that it would completely change the other flags? Because now I'm starting to think that the whole series has an even deeper bug: stupid new features that were badly thought out and not even described.”

Christoph pointed to the atomic input/output issue as described in <https://lwn.net/Articles/573092/>. Linus was unimpressed, saying that in the typical case, any code that wanted atomic input/output would probably still want to perform input/output without the atomicity guarantee, in which case there were straightforward ways to handle it. He added, “My guess is that there is going to be very few `O_ATOMIC` users anyway, and they'll very carefully set it once and test it (or not even test it – just make it be a configuration flag and tell people ‘don't ask for `O_ATOMIC` if your system doesn't support it’).”

Meanwhile, Florian Weimer said that probing for flags was “vastly preferable to hard-coding kernel version dependencies in source or object code. Particularly if you want to move applications between different kernel versions (which seems to be all the rage right now).”

But the conversation trailed off at that point. Regardless of whether there's a decent justification for probing, it's still an ABI change, which kills it at the starting gate. ■■■



OSDC

OPEN SOURCE DATA
CENTER CONFERENCE

MAY 16 – 18, 2017
BERLIN

SIMPLIFYING COMPLEX
IT INFRASTRUCTURES
WITH OPEN SOURCE

REGISTER NOW!



www.osdc.de

NETWAYS®

Creating more readable regular expressions with Simple Regex Language

Clear-Sighted

Regular expressions are a powerful tool, but they can also be very hard to digest. The Simple Regex Language lets you write regular expressions in natural language. *By Tim Schürmann*

Regular expressions are a fundamental feature of Linux – and many other modern operating systems. A regular expression is a search term with special placeholders representing several possible characters at the same time. The concept of a regular expression is an extension of the idea behind the “wildcard” character used in many GUI search tools, but the power and subtlety of regular expressions far exceeds what you can do with a simple wildcard.

For example, suppose you want to search the `system.log` file for errors, but you don't know whether the term `Error` will appear with initial cap or all lowercase (`Error` or `error`). You could use a regular expression as part of the `Grep` command:

```
grep -e '[eE]rror' system.log
```

The expression `[eE]` means: There is either a lowercase `e` or uppercase `E`.

A quick check for capitalization is easy to read and interpret, but some regular expressions are much more exotic. Who is able to say right away what text the following expression describes:

```
/^(?:\w+[\.\-\+])+(?:@) (?:[a-z][0-9][\.\-])+(?:\.)[a-z]{2,}$/i
```

Once you derive an expression like this, it can be a powerful tool for a script or a string search tool like `Grep`, but for the human who created this expression, and the other humans who come along later and want to read it, decoding a regular expression can be a time-consuming endeavor. What is more, a small error that creeps into the expression could be difficult to spot, although it could have a significant effect on the value of the search result. An error in a complex regular expression could even form the basis for malicious code and an Internet attack.

The fledgling Simple Regex Language (SRL, [1]) from the developer Karim Geiger aims to address the problem of incomprehensibility in regular expressions. Geiger started SRL as a bit of fun in Fall 2016, and since then, other developers have helped to implement SRL in various coding languages.

The SRL allows you to write regular expressions in natural English. In the previous example of the logfile, the two words `Error` and `error` start with either `E` or `e`. In SRL, you could say:

```
one of "eE"
```

and follow it with the character string `rror`:

```
one of "eE" literally "rror"
```




This line forms a complete expression in the SRL. SRL does not consider uppercase and lowercase for keywords, so `LITERALLY` is thus the same as `literally`. However, for literal strings, uppercase and lowercase are very important: `literally "Error"` therefore means something completely different from `literally "error"`.

In SRL, the developer can frame strings – in the example `rRor` – with single or double quotes. You have the option of separating the individual components of the complete expression with a comma or a line break. Adding a break does not change the logic but instead simply improves the legibility:

```
one of "eE",
literally "rRor"
```

The example expression matches all text passages where the character strings `error` or `Error` appear. Hence the word `Terrorism` would be a valid reference.

Empty Words

Spaces (whitespaces) correctly separate the words:

```
whitespace one of "eE" literally "rRor" whitespace
```

The word `error` is usually at the beginning of a line in logfiles. Anyone who is only interested in these lines, just needs to write:

```
begin with one of "eE" literally "rRor"
```

The test text now needs to start with `Error` or `error`. However, the expression only works if the program considers each line of the file as text to be retested (similarly to `grep`).

Some logfiles mark errors with the abbreviation `EE`, which you could include in the expression with:

```
begin with any of (literally "EE", (one of "eE" literally "rRor"))
```

As with traditional regular expressions, brackets group matching subexpressions. The term `any of` serves as a logical Or. In the example, the text looks for lines beginning with either with the character string `EE`, or with `Error` or `error`. The comma is cosmetic.

When the Post Rings

Sometimes characters should be repeated several times. For example, with the abbreviation `EE`, there are exactly two `E`s in succession. Or in SRL, you could say: `literally "E" exactly 2 times`. Instead of `exactly 2 times`, you could also write `twice`.

In the following expression:

```
begin with any of (any character, one of ".-+") once or more
```

the expression `any character` stands for any letters between `A` and `Z` or for a digit between `0` and `9` or an underscore `_`. Uppercase and lowercase are of no importance. The permitted characters can be repeated as often as desired; however, there must be at least one character. The entry `once or more` ensures a minimum of one character.

If the string you are looking for is an email address, you'll also need to ensure the presence of the `@` character: `literally "@"`. The domain name behind it may, in turn, be made up of several letters or numbers and the special characters `.` and `-`:

```
any of (letter, digit, @
one of ".-") once or more
```

The `any character` expression does not work for the domain name because domain names prohibit the underscore `_`. The `letter` and `digit` expressions specify letters and numerals without additional characters. The top-level domain, which starts with a period, forms the end:

```
literally "."
```

At least two more letters follow:

```
letter at least 2 times must end
```

The developer explains that uppercase and lowercase are irrelevant by explicitly adding `case insensitive`.

TABLE 1: Character Strings

Keyword	Description
<code>literally "string"</code>	Representative of the character string <code>string</code> .
<code>one of "abc"</code>	One of the characters <code>a</code> , <code>b</code> , or <code>c</code> .
<code>letter from a to d</code>	One of the characters <code>a</code> , <code>b</code> , <code>c</code> , or <code>d</code> . <code>letter without the appendix from ...</code> represents any lowercase letter.
<code>uppercase letter</code>	Any uppercase letter.
<code>any character</code>	Uppercase or lowercase letter from <code>A</code> to <code>Z</code> , a number from <code>0</code> to <code>9</code> , or an underscore (<code>_</code>).
<code>no character</code>	All other (special) characters.
<code>digit from 1 to 4</code>	One of the digits <code>1</code> , <code>2</code> , <code>3</code> , or <code>4</code> , whereby <code>digit without the appendix from ...</code> represents any digit between <code>0</code> and <code>9</code> .
<code>anything</code>	Any character with the exception of a line break.
<code>new line</code>	Line break.
<code>whitespace</code>	A whitespace character (this includes the space character, the tabulator, and the line break).
<code>no whitespace</code>	Character that is not a whitespace character.
<code>tab</code>	Tabulator.
<code>backslash</code>	Backslash character (<code>\</code>).
<code>raw "[a-z]"</code>	Stands for the result of the regular expression <code>[a-z]</code> .

TABLE 2: Quantifiers

Keyword	Description
<code>exactly 4 times</code>	Something repeats exactly four times. The expression <code>exactly 1 time</code> can be abbreviated to <code>once</code> ; <code>exactly 2 times</code> to <code>twice</code> .
<code>between 2 and 4 times</code>	Something repeats between two and four times; the following keyword <code>times</code> is optional.
<code>optional</code>	Something may occur, but does not have to.
<code>once or more</code>	Something must occur at least once.
<code>never or more</code>	Something must occur multiple times or not at all.
<code>at least 2 times</code>	Something must occur at least twice.

TABLE 3: Groups

Keyword	Description
capture (condition)	Captures the condition and can be returned from the engine. Anyone who uses capture multiple times can also give names to the individual captured parts: capture (anything once or more) as "first".
any of (condition)	Each condition within the brackets could apply.
capture (condition1)	Captures the expression condition1 if
until (condition2)	condition2 does not already apply.

TABLE 4: Lookarounds

Keyword	Description
if followed by	Checks whether something particular follows (lookahead).
if not followed by	Check whether something does not follow.
if already had	Checks whether something was preceding (lookbehind).
if not already had	Checks whether something was not preceding.

TABLE 5: Flags

Keyword	Description
case insensitive	Uppercase and lowercase are not of any importance.
multi line	The text to be checked runs over multiple lines.
all lazy	The evaluation is performed according to the Lazy principle.

LISTING 1: Checking an Email Address

```
01 begin with any of (any character, one of ".-+") once or
    more
02 literally "@"
03 any of (letter, digit, one of ".-") once or more
04 literally "."
05 letter at least 2 times must end
06 case insensitive
```

Listing 1 shows the whole expression. The expression deliberately keeps the email address test simple; for example, the standard allows other special characters in front of the @. The domain name must also always end with a letter or a number.

Testing, Testing, 1, 2, 3

You can test your SRL expression directly at the SRL project website under the menu item *Build* [2]. Just enter the SRL expression under *Your SRL Query*, type a test text under *Test Input*, and have it checked via *Run Query* (Figure 1). At the bottom of the page, developers immediately find out whether the test text matches the SRL expression. In addition, the page supplies the corresponding regular expression for comparison.

Figure 2 shows the expression for Listing 1 as an example – which, by the way, is identical to the cryptic regular expression at the beginning of this article. If the tester places a check mark in front of *Save Query* (to the right of *Test Input*), the server keeps track of all entries. The tester can use the URL at the bottom of the page to access the page with the SRL expression at any time. It remains unclear where the stored data will reside, so testers should not use sensitive data with *Test Input*.

What Next?

You might be wondering what you can do with the finished SRL expression, since Grep and most other tools only digest conventional regular expressions.

If you just need a regular expression on the fly, you can always enter the SRL expression into the test at the SRL website (as described earlier in this article), and then copy the resulting regular expression to Grep or another tool.

Also, some languages have already begun to implement SRL support. You will find special SRL libraries for JavaScript, PHP, Python, and C++ at GitHub under the MIT license [3]. Java and C# libraries are still pending.

The functions and classes of these SRL libraries accept an SRL expression, evaluate it, and convert it into a regular expression. In the case of PHP, the user just has to create an SRL object and check the text using the method `isMatching()`:

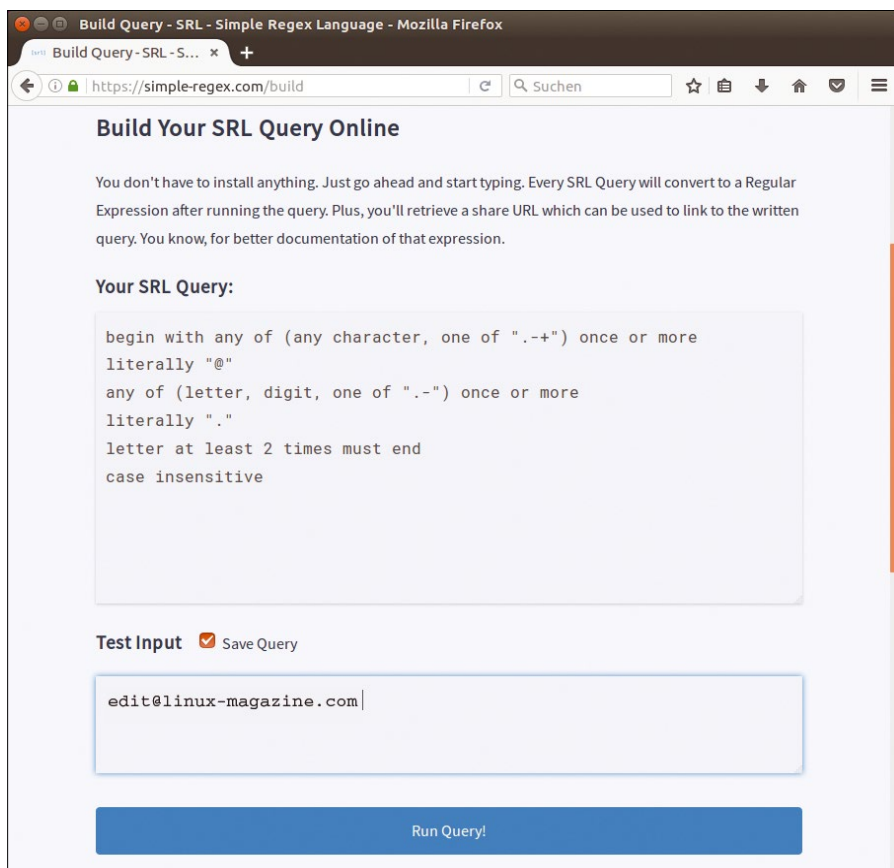


Figure 1: The SRL website uses the specified SRL expression to check whether `edit@linux-magazine.com` is a valid email address. The answer?

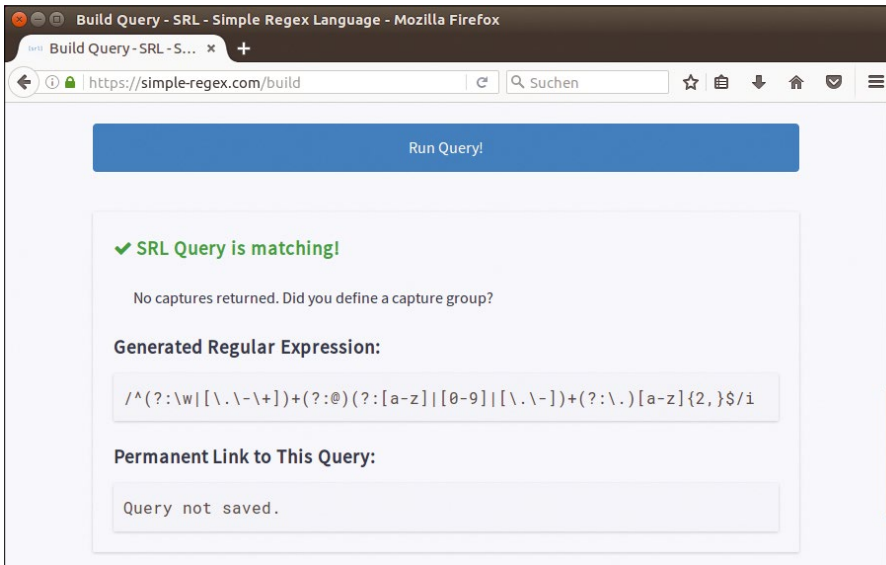


Figure 2: ... appears at the bottom of the page, along with the associated regular expression.

TABLE 6: Anchors

Keyword	Description
start with	Something explicitly refers to the start of a string.
must end	Something refers to the end of a string.

```
$srl = new SRL('one of "eE" literally "error"');$srl?
isMatching('Error'); // is True
```

In addition to featured keywords like `literally`, SRL also offers the keywords shown in Tables 1 to 6. You'll find a detailed reference and many other examples on the official SRL homepage [1].

Conclusions

SRL is built on the philosophy that, if regular expressions are easier to read, errors will stand out more quickly. It is

worth noting, however, that SRL expressions are also complex and difficult to understand if you aren't accustomed to the syntax. SRL does not currently feature comments, which would help to add clarity, and recursion is also missing. Classic Unix text tools such as Grep do not yet provide SRL support; however, you can convert the expression at the SRL website or use an SRL library with some programming languages.

Despite the problems, SRL is still worth a look. If you only use regular expressions occasionally, or if you are using them for the first time, you will get to your destination much faster with SRL. Even regex old-timers might find they can make their expressions more legible with SRL.

In the future, developer Karim Geiger wants to add support for additional programming languages and also standardize the SRL language to define the syntax and commands more clearly. In the long run, he imagines a kind of compiler that translates regular expressions into SRL. He has stated that a Bash version is not planned but is conceivable.

SRL commands are based on English. Geiger has resisted the suggestion to translate the natural language commands to other written languages. He fears that language proliferation would introduce a complexity that could lead to incompatible versions. ■■■

INFO

- [1] Simple Regex Language: <https://simple-regex.com>
- [2] Build Tool for the SRL: <https://simple-regex.com/build>
- [3] SRL Libraries: <https://github.com/SimpleRegex>

Shop the Shop shop.linuxnewmedia.com



RASPBERRY PI ADVENTURES



COOL PROJECTS FOR GEEKS OF ALL AGES

ORDER YOUR VERY OWN ISSUE!



ORDER ONLINE: shop.linuxnewmedia.com/se27

Exploring the Min web browser

In with Min



A simple design, efficient performance, and a built-in ad blocker are reasons for a closer look at the Min web browser. *By Mario Blättermann*

Google Chrome dominates today's market for web browsers, thus leading to a landscape dominated by the WebKit engine. Many free browsers, such as Epiphany, Midori, or Rekonq are based on WebKit, and WebKit is also the foundation for proprietary browsers such as Safari and Opera.

With all this consolidation in the underlying components of popular web browsers, it is both astounding and gratifying to see developers create a

browser that embodies a totally different approach. The Min browser [1] does not depend on a rendering engine but is built entirely from the Electron framework, which is used for creating cross-platform applications with JavaScript, HTML, and CSS. The name hints at Min's lean design and minimalist aesthetic: the screen looks pretty empty after the start, except for a menubar.

Min is a light, fast browser with some innovative features, including built-in ad

blocking, integrated search, and a unique approach to tab management.

Tabs, Search, and Shortcuts

Min's approach to tabs might be somewhat different from what you are used to with other browsers. The tabs, search, and address line are merged into a single widget. You just click in the area below the menubar to activate the input and start typing. A number of suggestions appear based on search results and previously viewed pages (Figure 1). Click on one of the suggestions, or highlight it with the arrow keys and press Enter, to load the page. Min shows the title of the page on the tab; you can view the URL by clicking on the tab again.

Open a new tab by pressing the plus icon at the right side of the current tab. Each tab has the same width, and the colors for the background only differ slightly; text in inactive tabs is grayed.

Min's search feature sends queries to previously viewed pages as well as the no-track DuckDuckGo [2] meta search engine. The search results were impressive in

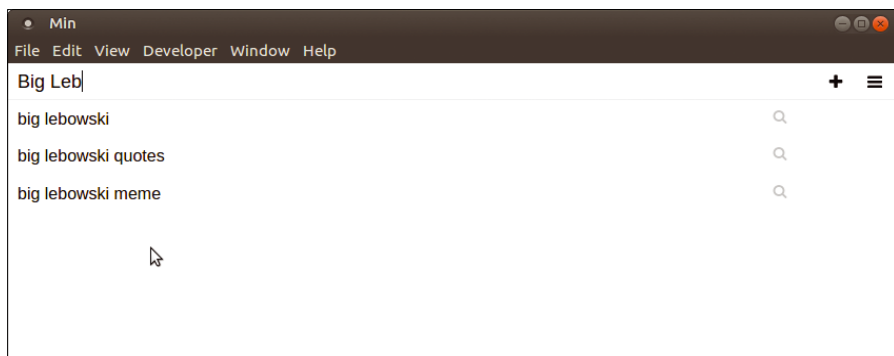


Figure 1: Min expands terms typed in the address bar with suggestions based on search results and previously visited sites.

Lead image © alphaspirt, 123RF.com

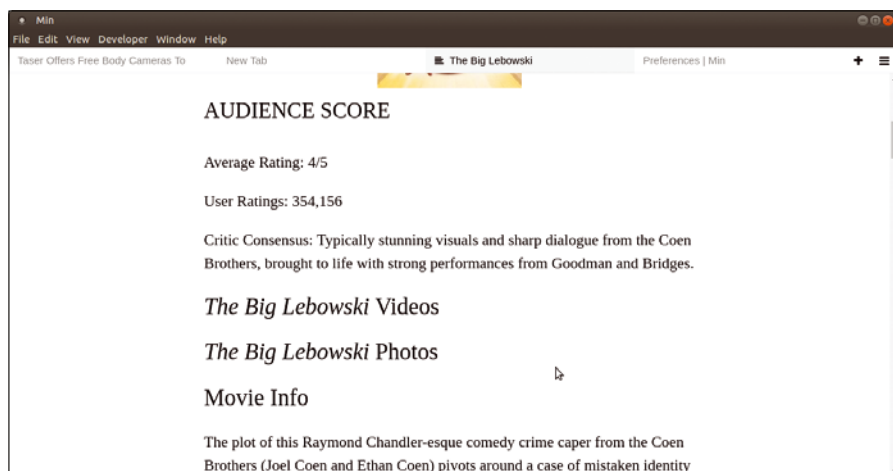


Figure 2: Pure text without decoration thanks to *Reading List* mode; unfortunately, this option also removes the links.

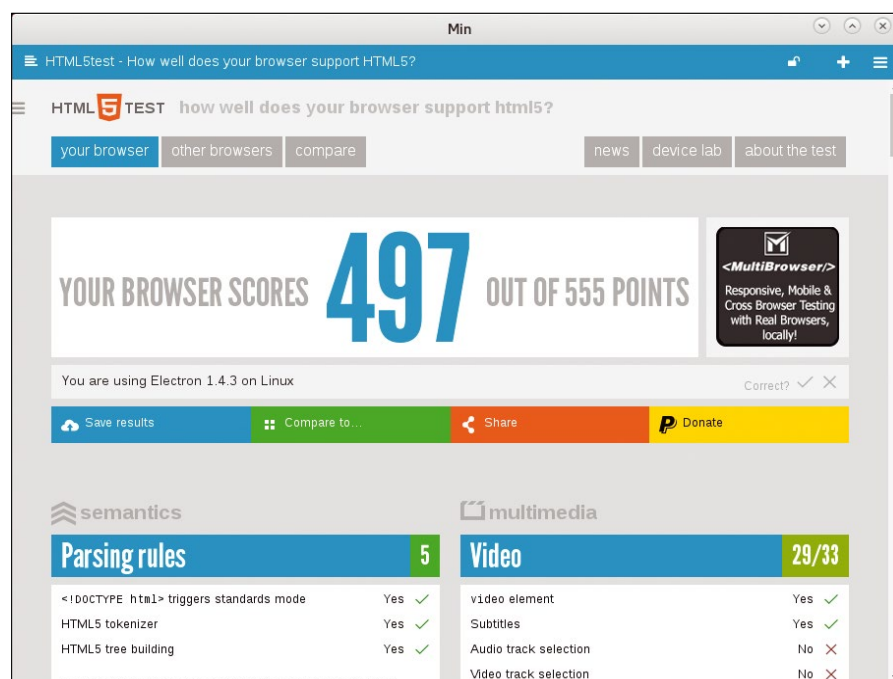


Figure 3: Min achieved a very good result in the test for HTML5 support.

our lab, which is just as well, because Min does not offer an easy means for changing the search engine: You need to edit the source code to change to a different search tool.

Min does not offer bookmarks: Instead, the program assumes you will find the right page by entering a few characters. This approach works reasonably well, but it still seems a little impractical. If you are starting from scratch, you might be fine without bookmarks, but many users have extensive bookmark collections gathered over the years and will not welcome the prospect of giving up their bookmarks.

At least Min understands a number of keyboard shortcuts; more detailed

information on shortcuts is available in the project wiki [3]. The shortcuts follow the standard used with other browsers and work environments, so you won't have to learn many new tricks.

Modes

Focus Mode hides all tabs except from the currently active tab. If you have opened a large number of sites, this feature is quite useful. To enter Focus Mode, select *View | Focus Mode* in the menu.

In Reading List Mode (Figure 2), Min will format some complex web pages for easier reading. Although Reading List mode makes the content more accessible, it does remove any hyperlinks embedded in the text.

The software saves all texts displayed in Reading List mode for 30 days. You can use the *View | Reading List* menu to access the page again.

Ad Blocker and Multimedia

The ad blocker in Min is fairly near the top of the feature list on the project website. But where other browsers use a plugin, Min builds ad blocking directly into its *Preferences* page. In our lab, the ad blocker worked well. Choose the *Edit | Preferences* menu to block scripts, images, or trackers and ads.

Min is up to date on audio and video playback. The browser also scores pleasingly high marks on the HTML5 support test (Figure 3). Google Chrome achieved just two points more on the same machine, and Mozilla Firefox ended up in third place.

Min generally had no problems with Internet radio stations or major platforms

AVAILABILITY

The *Download Min* link on the project website takes users directly to a Debian package for 64-bit systems. Min source code, as well as versions for Windows and Mac OS, are available on GitHub [4].

Installing from the source code is not trivial, and maybe also unnecessary. On a test system with Fedora 25, I only needed to unpack the Ubuntu package and the `data.tar.xz` tarball included in the package. The folder created subsequently (`usr/bin/`) contains an executable binary, which you can launch by double-clicking in the file manager. The program includes most of its run-time environment itself, only requiring in GTK version 2 to draw the graphical user interface.

To build Min from the source code, you also need the Electron [5] Javascript runtime environment. Electron does not yet exist in the form of packages for popular distributions, which complicates the build. Fortunately, the Electron project has precompiled packages for various platforms [6]; unpack the Linux package in the previously unpacked Min tarball first. On our Fedora test system, we also needed to install the `nodejs-grunt` and `npm` packages.

Run the commands in Listing 1. The last command launches the browser. You can convert this command to a desktop file that will let you launch the browser from the Start menu.

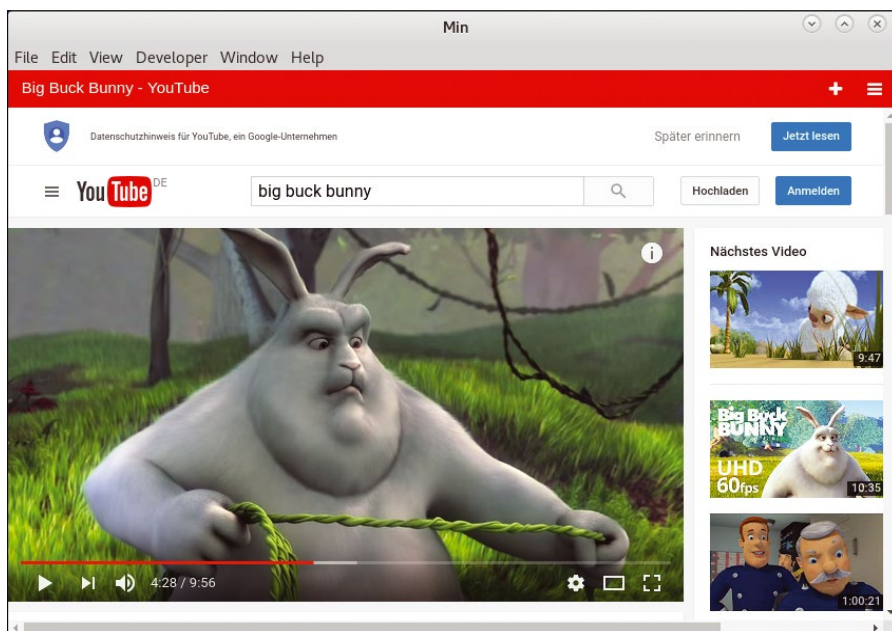


Figure 4: Big Buck Bunny takes up the fight in Min, even without Flash.

such as YouTube (Figure 4) or Daily-Motion. The only failures were in rare cases that required the dying Flash format.

To provide support for various multimedia formats, Min comes with the FFmpeg library. FFmpeg is one of the installation components when you

build Min from source code (see the “Availability” box).

For Developers

Min’s role as a browser for developers is evident from menu options such as *Developer | Inspect Page*, which opens a tool in a separate window that lets

you to examine the website code and check it for errors (Figure 5). If you hover the mouse cursor over individual

snippets of code, the program highlights the corresponding areas on the website.

Refresh the browser after changing the source code of the page by selecting *Developer | Reload*.

Conclusions

Min is quite fast and quite suitable for everyday use. After a short learning curve, work with the software is a pleasant experience, although the unusual implementation of tabs and the address bar diminishes the experience. Moreover, it takes some time to get used to the absence of bookmarks. On the other hand, Min is on par with its competitors in the field of multimedia.

The lack of support for Flash may be viewed as a shortcoming, but it is less of a problem in the face of Flash’s imminent extinction. More problematic is that there is no way to sync the browsing history across multiple devices, and the lack of a plugin interface could also affect Min’s chances of making a major breakthrough.

The *Help | Take a Tour* option is good way of getting used to Min. The tour reveals some features that you might not be familiar with from other browsers. For example, you can group tabs dynamically to form “Tasks.” The menu icon in the top right displays the open tasks, and the plus icon creates new tasks. You can assign tasks to a tab using drag and drop.

As of now, the Min browser really only works with English. We could not find a way to switch the browser interface to any other language. Although files for different languages exist, the menu does not have a matching entry to help users change the language. ■■■

INFO

- [1] Min: <https://minbrowser.github.io/min/>
- [2] DuckDuckGo: <https://duckduckgo.com/>
- [3] Min shortcuts: <https://github.com/minbrowser/min/wiki#keyboard-shortcuts>
- [4] Min download: <https://github.com/minbrowser/min/releases/>
- [5] Electron: <http://electron.atom.io/>
- [6] Electron download: <https://github.com/electron/electron/releases>

LISTING 1: Building

```
01 nonumber
02 $ npm install
03 $ grunt
04 $ ./electron-v1.4.4-linux-x64/electron .
```

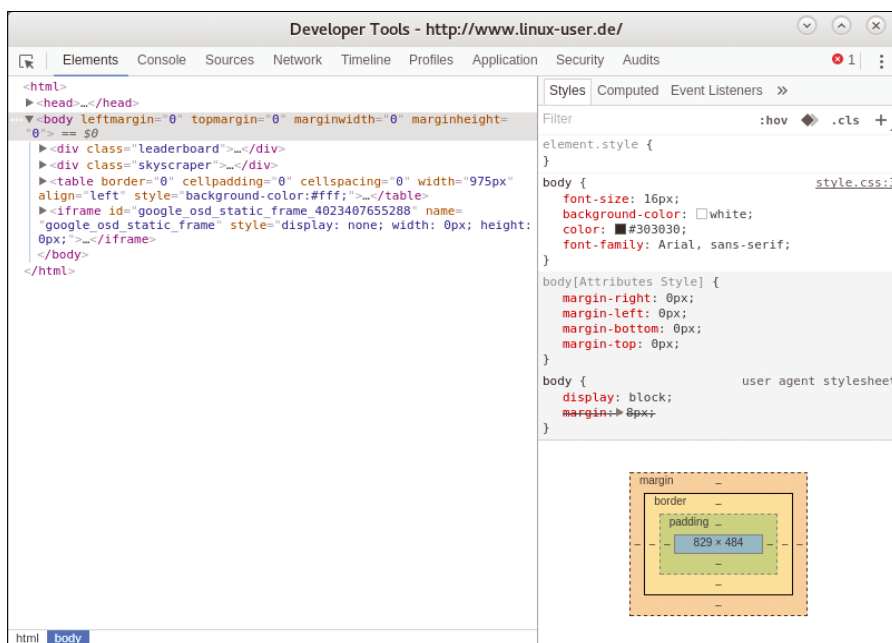
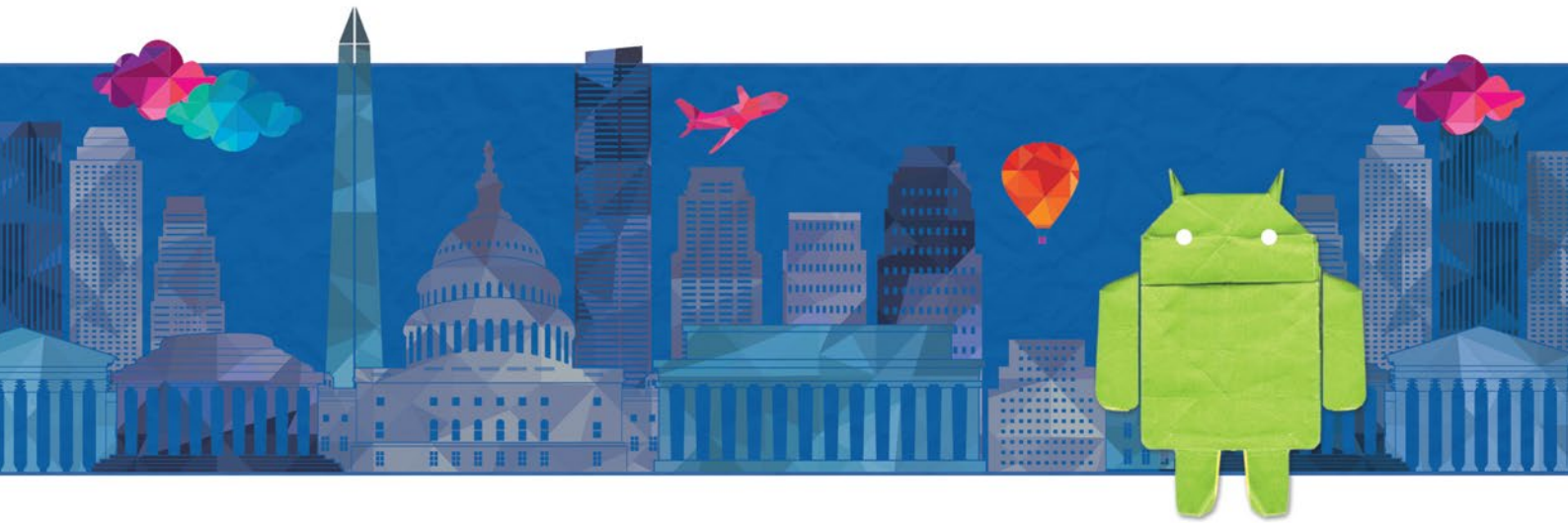


Figure 5: The Inspect Page option lets users inspect sites in depth.

➔ Register Early and SAVE!



Create/Design/Develop/Connect **AnDevCon** The Android Developer Conference

3 BIG TRACKS!

Android Fundamentals

A deep dive into the fundamentals all the way to the most advanced capabilities of Android and the Android ecosystem.



Cross-Platform Development

This track goes beyond native Android development to teach developers to master cross-platform development tricks, techniques, and tools.



Machine Learning/AI

An entire track at AnDevCon is devoted to machine learning and AI practices that are at the forefront of the next wave of Android and mobile technology!



Great classes, the reception is awesome and there's better technical content than Google I/O!

— Kevin Cousineau, Mobility Architect, Ideas Improved

AnDevCon is definitely worth attending. You get very useful information from very experienced speakers, and get to network with others.

— Anil K. Dokula, Software Engineer, Vedicsoft Solutions

July 17-19, 2017

Washington, DC

www.AnDevCon.com



A **BZ Media** Event

AnDevCon™ is a trademark of BZ Media LLC. Android™ is a trademark of Google Inc. Google's Android Robot is used under terms of the Creative Commons 3.0 Attribution License.

Security distribution

Under the Radar

Internet users can fly under the radar of hackers and data collectors with Discreete Linux. *By Erik Bärwaldt*

Banning uninvited guests from your computer these days can mean negotiating several obstacles. Leaks have revealed many of the sophisticated technical methods deployed by the intelligence services. Additionally, industrial spies and criminal hackers spare no effort to spy on users. The attackers now rely not only on software vulnerabilities but also on manipulated hardware, which poses problems for special occupational groups that rely on discretion – such as journalists, lawyers, or doctors.

Discreete Linux [1] seeks to put a stop to intrusion attempts. The former Ubuntu Privacy Remix [2] now uses Debian 8 as its basis and is available as a beta version. In this article, we take an in-depth look at the updated edition.

Hardened

The system is available as 1.6GB hybrid image [3]. You can launch and run this on a USB stick or an optical medium. Discreete Linux lacks a persistence mode that lets you store your own files on USB flash drives. The system only boots on read-only ISO 9660 filesystems and creates a temporary overlay filesystem to avoid loading malicious software unnoticed in the background.

It not only does without kernel modules for internal ATA drives (optical drives being the exception) but also

prevents the communication with the outside world over the network. Discreete Linux explicitly mounts USB flash drives or other external disks without the option of running programs – bad luck for malicious software and spyware programs. Last but not least, the user has no way of gaining root privileges.

GUI

Discreete Linux comes up with a fairly recent version of the Gnome desktop, 3.14.1, which has several smaller applications on board. Additionally, the operating system integrates software packages from the KDE collection and distribution-independent applications such as LibreOffice, Gimp, Firefox, and VLC. It also preinstalls applications for specific hardware, such as scanners. These

features make the operating system a solid all-rounder.

CryptoBox

To allow you to use Discreete Linux like a conventional distribution, the Debian derivative provides a special wizard, which you will find in the *Applications | Security | CryptoBox Wizard* menu. The wizard creates encrypted containers on a removable disk, but also uses available space on the boot drive. Setting up the containers couldn't be any easier: In a dialog, you specify where you would like to create a CryptoBox and what size you want it to be (Figure 1).

There is the possibility to create a container for automatically backing up your data, and you can determine the encryption format. When you are done, press *Continue* to go to a second dialog, where you enter the ID and a password with a length of at least 20 characters. In the same dialog window, use the two selection fields to choose the encryption and

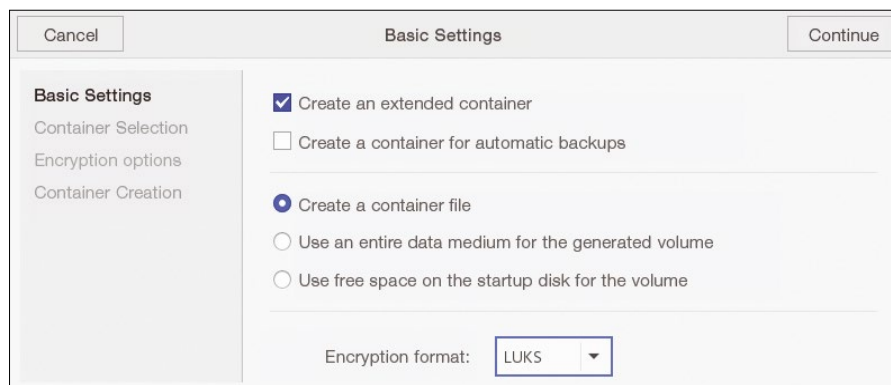


Figure 1: Discreete users can generate an encrypted container in a simple dialog.

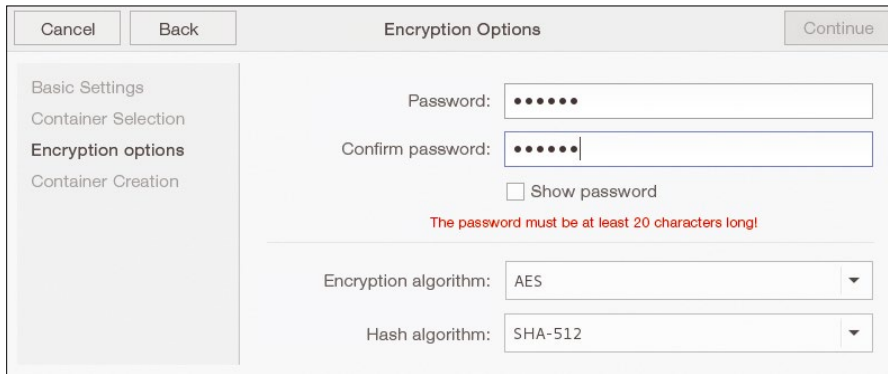


Figure 2: Users can choose from various encryption methods for containers.

hash algorithm; the options depend on the selected encryption format (LUKS [4], VeraCrypt [5], TrueCrypt [6]) (Figure 2).

To generate a key that is as random as possible, the software also prompts you to move the mouse for at least a minute. Then, it creates the container and is ready for use.

Decryption

Opening an encrypted container is a smooth experience: Select the *Scan for CryptoBox Volumes* item in the *Applications | System Tools* menu to select the desired drive in a simple dialog window and then enter the passphrase. The volume can then be used transparently in the file manager. To close an open container volume, click on the volume icon, and select the item the context menu.

Windowing

The CryptoBoxes also run on third-party computers – if an appropriate infrastructure such as LUKS or VeraCrypt is in place. The systems mount the boxes in read-only mode, so that no unwanted data ends up on the volume. To open the container, you need the passphrase specified on the original system.

Passwords and Keys

GnuPG [7], Seahorse [8], and a matching Gnome GUI help users keep the cryptographic keys safe that secure the communication. You can use USB sticks to pass these keys on, if required. Also, several other tools help users with GnuPG management.

Figaro's password Manager 2 [9] is a graphical tool to help you manage the flood of passwords required by today's application programs, thus offloading the task from the user. The software is found in the *Applications | Security*

menu. This where you will find version 1.19 of the TrueCrypt successor VeraCrypt [5] and the MAT program [10] (Figure 3). MAT anonymizes the metadata, which some file formats store in the background, as well as which users to be profiled under certain circumstances. The small tool makes such data unusable for unauthorized third parties.

In Practice

In a *Linux Magazine* hands-on test, we checked the Debian derivative's suitability for everyday use. The CryptoBox containers in particular will naturally tend to increase the overhead for storing and reconstructing data. But the developers have definitely done their homework here. The easy-to-use wizard and seamless integration with cryptosystems mean that there is little additional overhead for users of the encrypted disks: You only need to enter one password to work transparently with the containers just like with conventional mass storage.

The whole offline concept makes it extraordinarily difficult for attackers to



Figure 3: Discreete Linux has multiple security applications, including MAT and VeraCrypt.

even gain access to the system. Thanks to the live mode with a read-only file system and because mounting stationary disks is impossible, Discreete Linux also cannot be attacked via a manipulated host computer.

Conclusions

The prerelease version of Discreete Linux already covers most of the requirements that exposed groups of people usually impose on a hardened environment. This isolated solution with encrypted-only data transfer to the outside world implements the CryptoBox in a very user-friendly way. Last but not least, the stable Debian base minimizes problems due to software bugs.

But with attack scenarios constantly emerging, the makers of Discreete Linux are already planning further measures: In the future, they will only be offering signed kernel modules and hardening the USB interface, which is increasingly being used for attacks. The latter is intended to prevent the injection of bad USB trojans [11] via USB components. All in all, the beta phase Discreete Linux already presents itself as a successful niche product for users with strict security requirements. ■■■

INFO

- [1] Discreete Linux: <https://www.discreete-linux.org>
- [2] "Ubuntu for the Security Conscious" by Kristian Kissling: <http://www.linux-magazine.com/Issues/2013/157/Ubuntu-Privacy-Remix>
- [3] Download from: https://www.discreete-linux.org/howto.html#get_pre
- [4] LUKS: <https://gitlab.com/cryptsetup/cryptsetup>
- [5] VeraCrypt: <https://veracrypt.codeplex.com>
- [6] TrueCrypt: <http://truecrypt.sourceforge.net>
- [7] GnuPG: <https://www.gnupg.org>
- [8] Seahorse: <https://wiki.gnome.org/Apps/Seahorse>
- [9] Figaro's Password Manager 2: <http://als.regnet.cz/fpm2/>
- [10] MAT: <https://mat.boum.org>
- [11] Information on bad USB trojans: <http://www.golem.de/news/karsten-nohl-usb-geraete-aller-typen-lassen-sich-fuer-badusb-nutzen-1411-110520.html> (German only)

Graphical interfaces for systemd

WELCOME STARTUP SUPPORT

Systemd has won the race, as indicated by the several tools that already offer a service just a mouse click away. We look at six of these tools. *By Erik Bärwaldt*

Systemd [1], the initially highly controversial system and process manager in the Linux community, is now considered established – the old SysVinit system has largely had its day. However, critics still gripe about the binary format of the logfiles, the fact that systemd sometimes uses DNS queries on Google’s name servers, and the strong orientation toward the Gnome desktop.

Nonetheless, systemd also offers advantages compared with its predecessor. The fact that it starts processes in parallel significantly reduces the time required for booting the operating system, especially on multicore computers. And, the compiled program accelerates the process.

Systemd Basics

The processes managed by systemd are called units. They are formally reminiscent of traditional INI files, and they perform

various tasks: They manage services, define mountpoints, take care of the swap memory, or create devices.

Service units that manage services are similar to the init scripts from SysVinit or Upstart. The function a unit performs is revealed by its file extension; for example, variations with `.service` take care of services, and units with `.mount` as the extension mount filesystems. Units with a `.timer` extension control repetitive tasks and are reminiscent of cronjobs, whereas `.device` units create device files.

The `.socket` and `.target` unit configuration files take up a special position: `.socket` activates a service, whereas `.target` groups multiple units and partially replaces the runlevels familiar from SysVinit.

Admins control the units on the command line with:

```
systemctl <Options> <Commands>
```

This, among other things, is used for editing the configuration files of the respective units, which are available in text format. To do so, enter:

```
systemctl edit --full <Unitname>
```

at the prompt. Journald also acts as a service unit of systemd that takes care of the log entries in the central journal. The command `journalctl <option>` offers insight into the logfile containing numerous parameters for partial queries.

Graphical Tools

Admins usually configure, control, and monitor systemd using the command line. But, as the init system continues to spread, there are also more and more graphical tools, and their range of functions vary significantly. If you need a solution to suit your individual needs, take

Lead Image © Kheng Guan Toh, 123RF.com

TABLE 1: Graphical Tools for Systemd Management

	Kcmsystemd	systemd System Manager	systemd Manager	systemd-KCM/ SystemdGenie	Cockpit
Interface	KDE 4.x	GTK3+	GTK3+	KDE Plasma 5	Browser-based
Integrated in tools	Yes	No	No	Yes/No	No
Process management	Yes	Yes	Yes	Yes	Yes
Processing function for units	Yes	No	Yes	Yes/Yes	Yes
Start and stop units	Yes	Yes	Yes	Yes/Yes	Yes
Time control	Yes	No	No	Yes/Yes	Yes
Journal configuration	Yes	No	No	Yes/Yes	Yes
Journal display	No	No	Yes	No/No	Yes
Logind control	Yes	No	No	Yes/Yes	Yes
Snapshot	No	Yes	No	No/No	No
Dependencies display	No	Yes	Yes	No/No	No

a closer look at all the candidates before installation (Table 1).

Along with the software Cockpit [2] (developed by Red Hat), which partly has a similar functionality to Webmin [3] and is thus primarily used for server administration, several small projects now also manage systemd on single-user systems. However, they usually require a special desktop environment. These include the Systemd System Manager in the package `systemd-gui` or `systemd-ui` [4], although this is an interface based on the GTK+ toolkit.

Other options are Systemd Manager [5], with a GTK 3 interface, and Kcmsystemd [6], which is on the KDE desktop. A further development of this software is called Systemd-KCM [7]. The current version requires KDE Plasma 5 as a desktop environment and does not work with older versions of KDE. As of recently, the original KDE configuration module is called SystemdGenie [8] and exists as a standalone application. Besides this desktop-specific application, there are other applications – such as, Gnome Logs [9] – that only support part of systemd with Journald and thus are not fully fledged management tools.

The individual applications are also integrated differently in the desktop: Although the tools for GTK-based systems work largely as standalone applications, the KDE versions fit into the desktop’s control center as individual modules except for the new Systemd-Genie.

One advantage is that the users also have to configure the systemd components through a single interface and do not have to manage their system using

multiple applications with partially different control concepts. However, the naming can cause confusion: For example, the Systemd System Manager [4] is hidden behind the package `systemd-gui` in Linux Mint and Ubuntu (`systemd-ui` in Debian) – and inexperienced users often confuse this with Systemd Manager [5].

Kcmsystemd

For the KDE desktop in version 4.x, Kcmsystemd [6] integrates seamlessly (the first three letters stand for KConfig modules) into the management tools of the work interface. If the user installs the package of the same name, he or she will encounter the *Systemd* starter in the *System* area of the *Customize your desktop* front end. Clicking here opens a list window with several horizontal tabs, which make it clear that Kcmsystemd itself is a comprehensive solution for managing systemd (Figure 1).

The start activates the *Units* tab in the left pane. It specifically lists all units and shows their current status. Only active units are preset. If you check the *Show inactive* option, you will also see inactive units. Because there are usually several hundred entries, a search box for tracing the desired unit is also a great help.

The list with units shows the *Load state*, *Active state*, the *Unit state*, and the name of the process for each entry. It also reveals whether the system has loaded and activated the respective unit and indicates the current status. Click on any process, and details appear below the list.

You can stop or restart individual units using the *Stop* and *Restart* buttons; however, this does not work for all entries. You cannot stop or restart device files, which help systemd to manage connected devices. With such units, the software grays out the corresponding control fields.

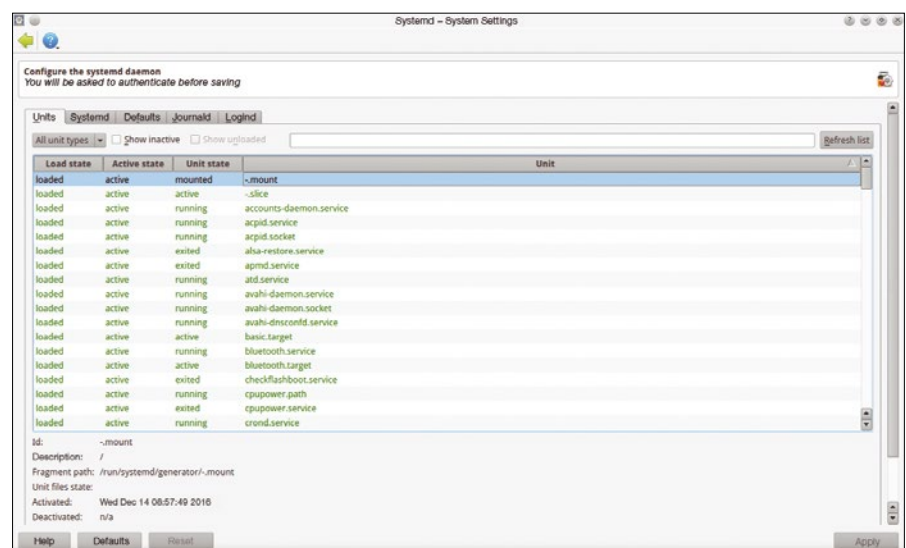


Figure 1: The Kcmsystemd graphical interface is clear but also has a number of control elements.

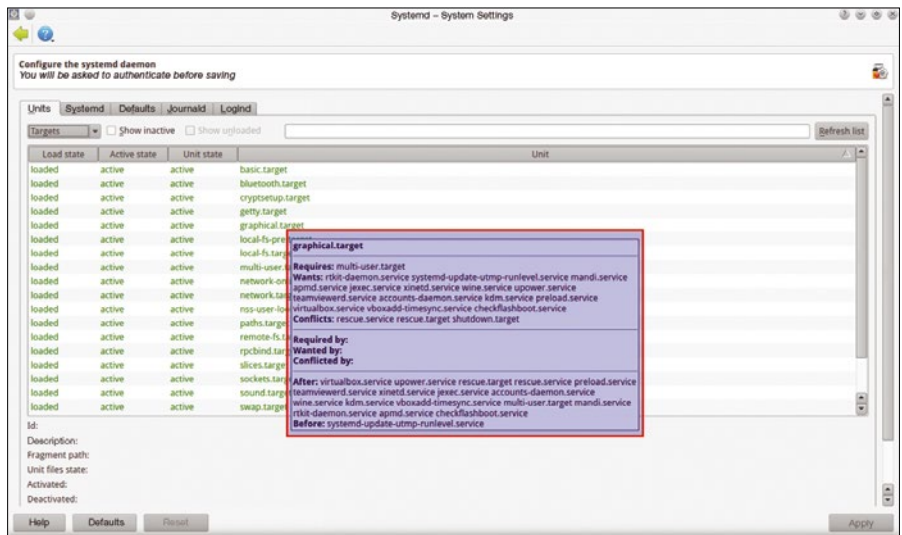


Figure 2: A box (highlighted in blue) with the display of processes shows information about the interaction of multiple Kcmsystemd units.

If the user passes over the individual units with the mouse, the dependencies of the unit from others appear in temporary pop-up windows. The admin thus immediately sees which other units are required in parallel (*Requires:*) or absolutely presupposes (*After:*), and which ones systemd wants to start in parallel regardless of other units (*Wants:*). Particularly with complex targets, this information helps solve problems with the init system (Figure 2).

Right-clicking on any process opens a context menu that – depending on the kind of unit – presents different options. If there are problems, the admin can use it to turn off certain services (*Disable Unit*). Regardless of the type of process, the admin can also edit the corresponding configuration files with the exception of the device units.

To do so, you need to click on the *Edit Unit File* option in the context menu and then be authenticated as an administrator. This opens the text editor KWrite with the corresponding configuration file for the selected unit, which can now be easily edited (Figure 3).

The admin then needs to restart the unit so that the modifications are valid. Cryptic startup scripts and the use of a shell interpreter are things of the past in systemd.

The tabs *Systemd*, *Defaults*, *Journald*, and *Logind* to the right of the *Units* tab are not dedicated to the configuration of individual processes and targets but rather deal with adapting systemd itself, including logging as

well as registration and session management.

On the other hand, the admin can adjust the logging system *Journald* separately. Along with path specifications to the journal store and the size of the journals, the configuration options of the four tabs also include CPU-specific information. This is how the user can, for example, assign CPU resources in the *Systemd* tab in the *Advanced Settings* or perform certain

hardware-dependent configurations (Figure 4). The session management *Logind* tab includes settings for system conditions such as for power management or virtual terminals.

The admin can write the modifications to the */etc/systemd* directory using the *Apply* button at the bottom right of the settings window. The software only initiates the write operation after authentication via a separate window.

Systemd System Manager

Systemd System Manager [4], designed for GTK work interfaces, comes with a very simple interface, which therefore also offers far fewer configuration and information options than Kcmsystemd. After installing the *systemd-ui* package, the *systemadm* entry is waiting for the user in the menu structure.

The Systemd System Manager interface, just like the KDE counterparts, lists

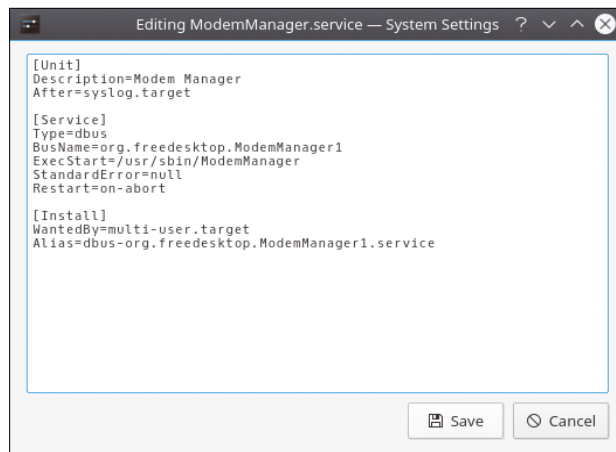


Figure 3: You can easily edit configuration files.

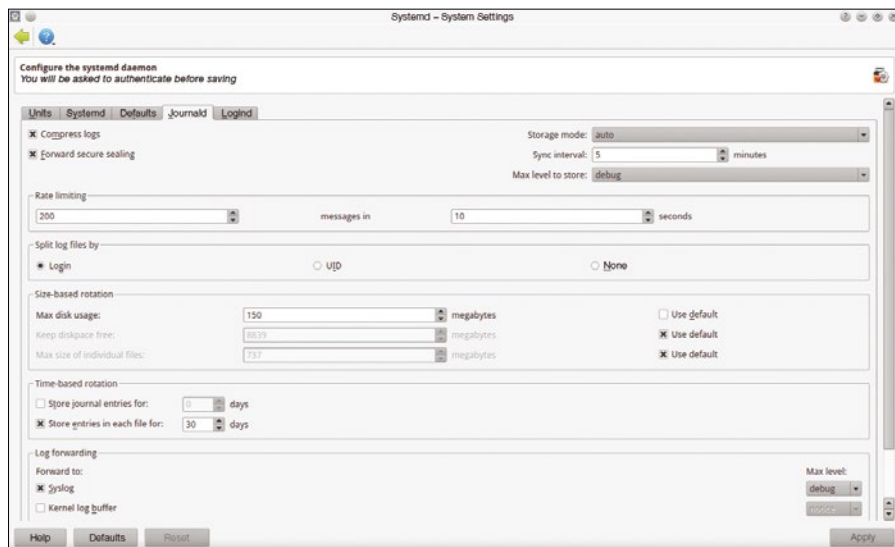


Figure 4: Interested individuals can perform very fine adjustment of the protocol function using Kcmsystemd.

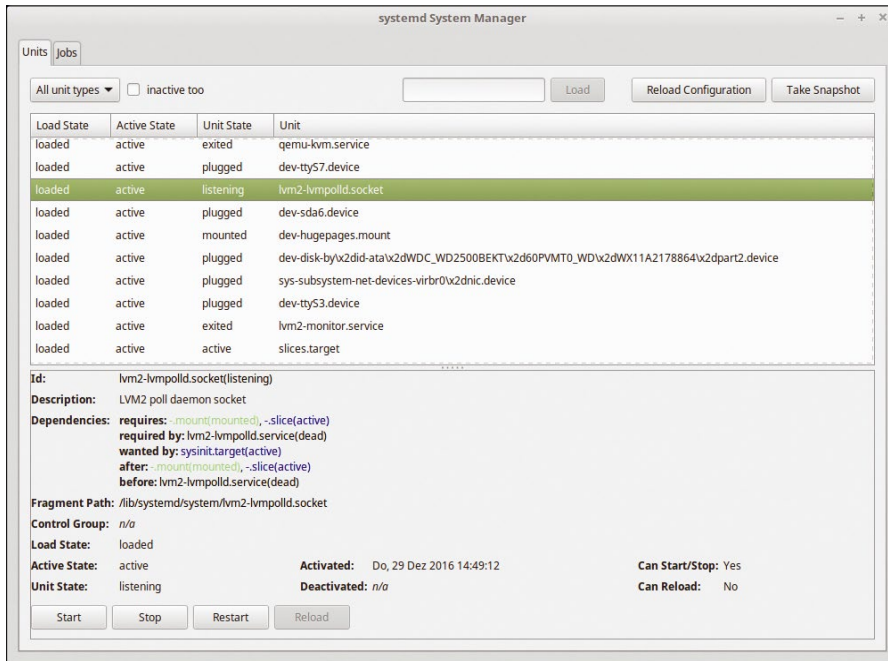


Figure 5: The program window of Systemd System Manager provides fewer options than Kcmsystemd.

the existing active units – including their states – in its main window. Here too, the admin can switch on inactive units via check marks, as required. Via the *All unit types* selection box, the admin can select certain unit types and thus get a grip of the numerous units of the system. Below the list area, the program window also displays in-depth information about selected units. Unlike Kcmsystemd, however, Systemd System Manager is missing the option to edit individual units via a context menu and modify their configuration files (Figure 5).

This manager displays existing dependencies at the bottom of the program window and uses different colors to do so: Correctly loaded and active units appear in green; red-colored units have crashed or were not loaded by the system for other reasons. You can activate and stop the units accordingly using the *Start*, *Stop*, *Restart*, and *Reload* buttons.

You can freeze the system in a defined state using the *Take Snapshot* button at the top right. Alternatively, you can restore a previous system state using the *Reload Configuration* button, which causes the system to reload disabled services, for example.

The Systemd System Manager does not adjust the journal function or session management. You need to access the terminal to do this.

Systemd Manager

The fledgling project Systemd Manager [5] is also designed for GTK + -based work interfaces, however in version 3. Furthermore, the software written in Mozilla’s programming language Rust [10] is currently only in the standard repositories of the Russian Rosa Linux and of Arch Linux. The developer provides an independent archive for Fedora. For Debian, there is a Debian package intended for 64-bit hardware that also runs easily on Ubuntu and Linux Mint. The

GitHub page also contains source code for the manual translation.

The Systemd Manager presents a clear, two-part interface: On the left of the window, the individual units appear with two status indicators *Enabled* and *Active*, and the right area shows the configuration files for the respective units. The *File*, *Journal*, and *Dependencies* tabs are located above the right-hand window segment. Whereas *File* is active by default and displays the configuration file, *Journal* presents the journal entries of the respective unit. *Dependencies* visualizes the existing dependencies in a tree shape.

Above the two segments is a line with information about the respective service. The admin can easily start and stop the service using the *Stop* and *Start* buttons on the right of the window. To the left, the admin can switch on or off the current unit using the slider *Enabled*: at system startup (Figure 6).

However, Systemd Manager does not display all the unit categories created by Systemd: The *Services* option is located at the top of the program window by default. Alternatively, you can select *Sockets* and *Timer* here. The program does not support additional, expectable categories such as *Targets* or *Devices* (Figure 7).

Modifications

The Systemd Manager allows the simple modification of existing configuration

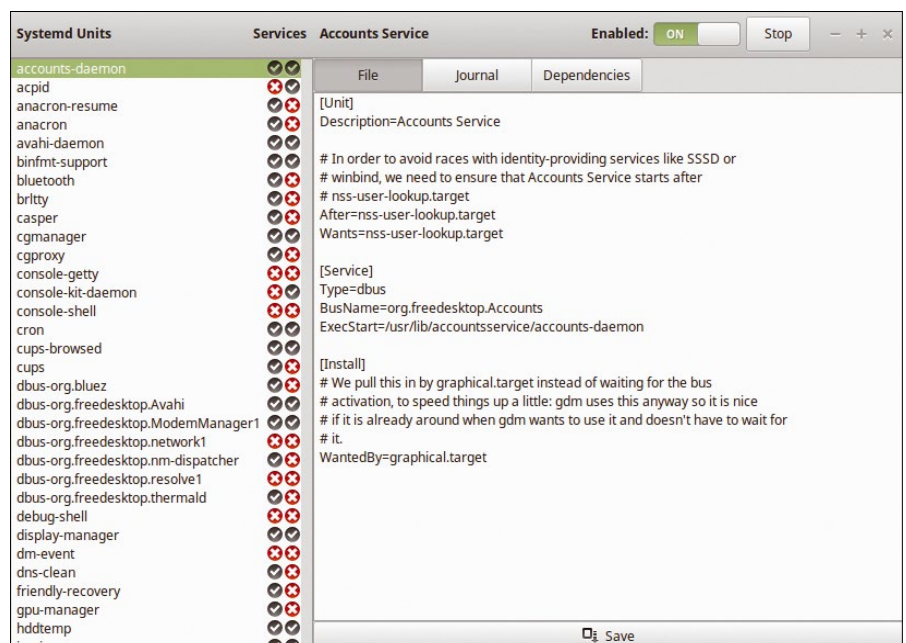


Figure 6: The Systemd Manager interface is very clearly presented.

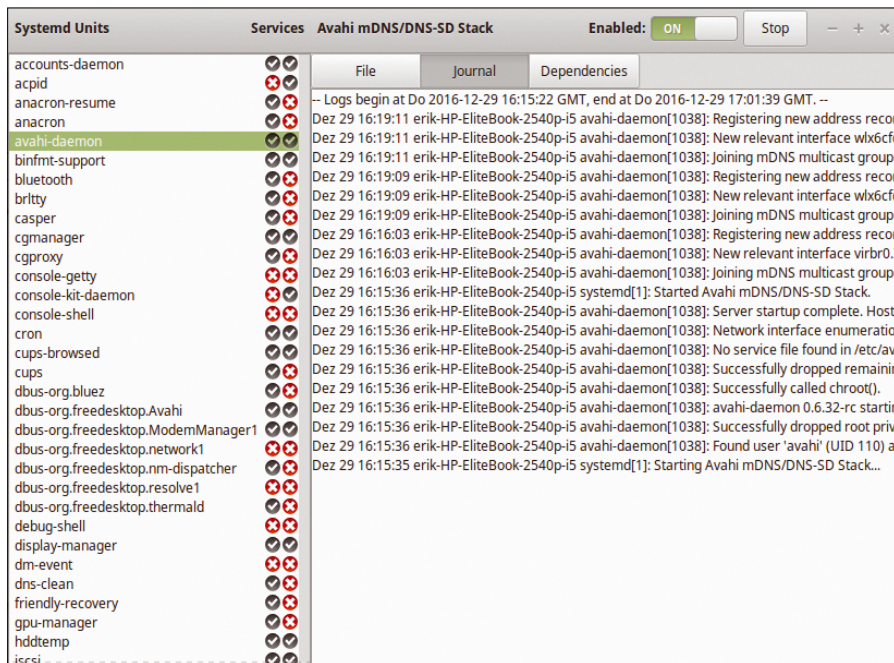


Figure 7: Systemd Manager displays the main unit types, though not all, in the selection box. This also puts the associated journal entries on the screen.

Systemd Analyze		Blame
Kernel: 9.169s	Time (ms)	Unit
+ Userspace: 39.404s	1952	qemu-kvm.service
= Total: 48.573s	2328	rsyslog.service
	2359	mdadm.service
	2384	thermald.service
	2456	apache2.service
	2474	console-kit-log-system-start.service
	2544	gpu-manager.service
	2558	NetworkManager.service
	3561	grub-common.service
	3776	lvm2-monitor.service
	3824	apparmor.service
	3984	systemd-udev.service
	4077	nginx.service
	4385	ModemManager.service
	5342	libvirt-bin.service
	9242	dev-sda1.device
	15289	NetworkManager-wait-online.service
	20538	mysql.service
	225077	ntp.service

Figure 8: You can easily determine hits and misses at startup with a mouse click for Systemd Manager.

files: If you have selected a unit and can see the associated configuration file on the right-hand side of the window in the *File* tab, you can simply write in it and make the changes there. Then just click *Save* at the bottom right to save and use the configuration file.

Analysis

If you click on *Systemd Units* at the top left, you can initiate an analysis of the loaded services by selecting the *Systemd Analyze* option in the selection box that opens. The Systemd Manager then

changes the window view, where the general loading times for the system appear in seconds on the left, and the loading times for the individual units are listed on the right. Here the admin can see which services need especially long loading times (Figure 8).

systemd-kcm

The configuration module *systemd-kcm* [7], which like its predecessor for KDE-4.x desktop seamlessly fits into the work interface's configuration dialogs, is especially suitable for KDE Plasma 5. *Systemd-kcm* also doesn't automatically end up on the hard disk with the major distributions; instead, you have to import it separately.

The chaos of names does cause a few problems here: Although the module appears as *systemd-kcm* on Arch Linux and Open Mandriva, it is called *plasma5-systemd-kcm* on Rosa Linux distributions and *kde-config-systemd* on Kubuntu.

However, the interface hardly differs from the module for KDE 4.x at first glance: *Systemd-kcm* also opens a program window with a large display area with several lines, and the display area only has a few control elements. When looking more closely, however, it becomes clear that the tab structure is different from the predecessor: Configuration options for session and login man-

agement are now missing, and the *User units*, *Conf*, *Sessions*, and *Timer* tabs reappear (Figure 9).

User units shows units that are active at the user level, and the *Units* tab lists all system units. The range of functions of these two windows is identical to that of the aforementioned items.

In the third tab – *Conf* – you can configure the individual systemd components. You can determine in a selection box at the top right whether to edit the *system.conf*, *journal.d.conf*, or *logind.conf* file. The individual parameters (which can be edited) of the selected configuration file then appear below in a table. Here, too, notes about the individual options appear when moving the mouse over them (Figure 10).

Systemd-kcm shows modifications in bold font, so that the user can get a quick overview of the changes. To save them in the respective file, you can use the *Apply* button at the bottom right and must then be authenticated, as the software starts from the conventional KDE system settings if there are no root privileges in place.

The *Sessions* tab manages the Logind settings. It displays all Logind sessions, which the admin can activate, terminate, or block with a right-click. In the final tab, *Timer*, the user receives a detailed overview of the existing timer units. They may already appear in the first tab if you select the *Timer* option from the selection list, but the appearance in their own tab is much more meaningful. Not only are the timer units displayed here (*systemd KCM* lists system and user timers), but also the exact times of the last and the next execution. The respective activated service unit also appears in the table.

SystemdGenie

SystemdGenie [8], the first version of which was released at the end of 2016, is in fact a variant of *Systemd-KCM* disconnected from the KDE system settings. So far, the software has only been available in source code [11], with the exception of Gentoo Linux [12]. After being compiled as a standalone application, it provides the same functions and a tab structure that is basically identical to *Systemd-KCM*. The only differences are a menubar and a buttonbar with some frequently used functions such as

for starting, stopping, and reloading the respective units.

The basic requirement for using SystemdGenie is Qt libraries from version 5.4; SystemdGenie does not work on older KDE variants. The developer provides a quick installation guide at [13].

Cockpit

The Cockpit project [2] primarily concerns server management, but, thanks to the close combination with systemd, also provides the option to administer the session manager on the target system. The current version is 126 – apparently the developers no longer use the major number before it.

Red Hat maintains and further develops the software, and systemd is a mandatory requirement. Prebuilt packages and repositories are available for Fedora Server, CentOS, Red Hat Enterprise Linux, and RHEL Atomic. The application can be found in separate repositories for Debian and Ubuntu, while Arch Linux has unofficial support [14]. Additionally, the developers make the source code of the software available

under LGPL on GitHub [15].

False Start

A shortcoming initially stood out during the test installation with the current Cockpit version 126: Unfortunately, the project developers failed to explain the exact specifications for using Cockpit with Ubuntu. If users want to install the program using the official archive in Ubuntu 16.10, the existing PPA archive does not integrate into the system because it is intended for Ubuntu 16.04. Installation on the slightly older Ubuntu version (which is, however, provided with long-term support) works easily using the three steps in Listing 1.

In the end, the user starts the systemd-socket, which, using

```
sudo systemctl enable --now cockpit.socket
```

is no problem. The software is now ready for use, and its interface appears in the web browser of your intranet if you type `https://<Server-IP-Address>:9090` into

LISTING 1: Install Cockpit in Ubuntu 16.04

```
sudo add-apt-repository ppa:cockpit-project/cockpit
sudo apt-get update
sudo apt-get install cockpit
```

the address bar. With the exception of the installation on the server itself, Cockpit only uses a connection secured with SSL/TLS, whereby it creates a self-signed certificate when first started.

If no connection is established, this is probably due to a blockade of port 9090 via the firewall, and you'll need to unlock the port – Fedora Server does this innately by the way.

The Cockpit dashboard then appears with the login screen in the browser. The admin is authenticated and thus moves into the home screen. On the left are selection lists for various tasks, and the corresponding content appears in a large area on the right (Figure 11). To configure and manage systemd processes on the server, select *Services* on the left of the selection area. The software lists the existing processes and services in a table, whereby Cockpit

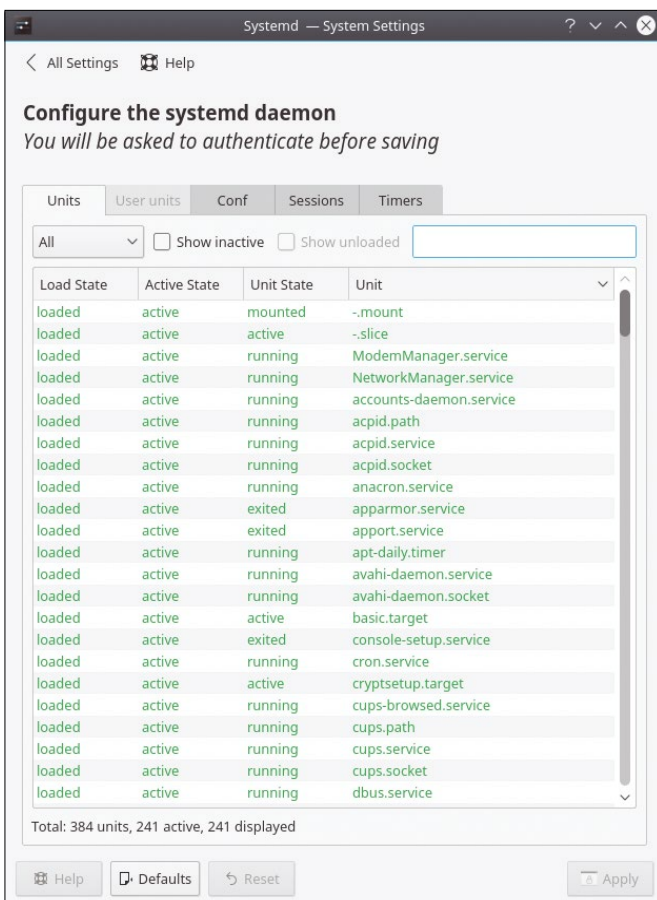


Figure 9: Systemd-KCM has a similar interface to Kcmsystemd but has a few modified options.

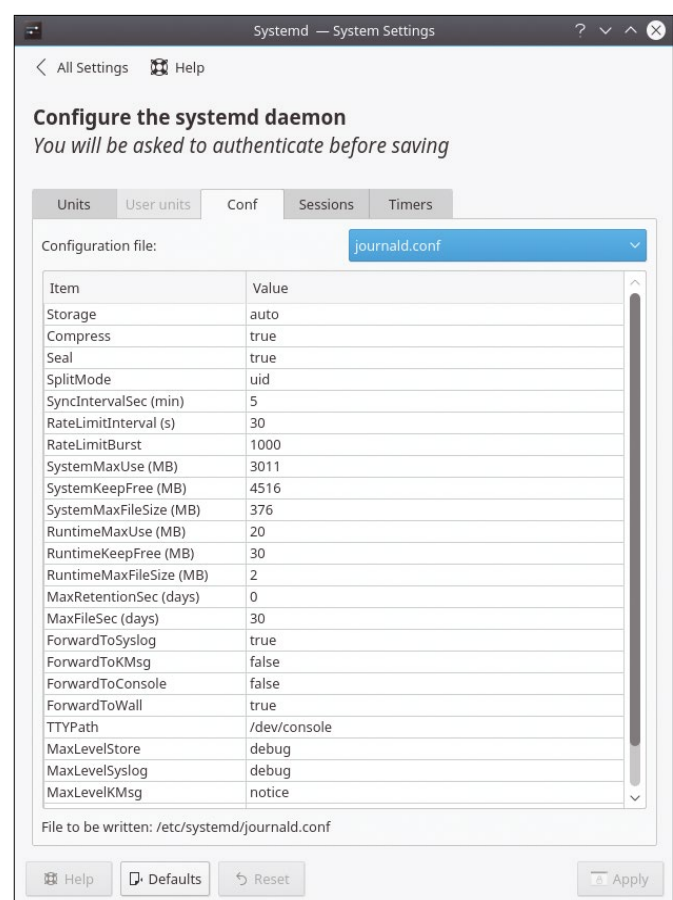


Figure 10: The parameters of the configuration files can be modified through entry in the program window.

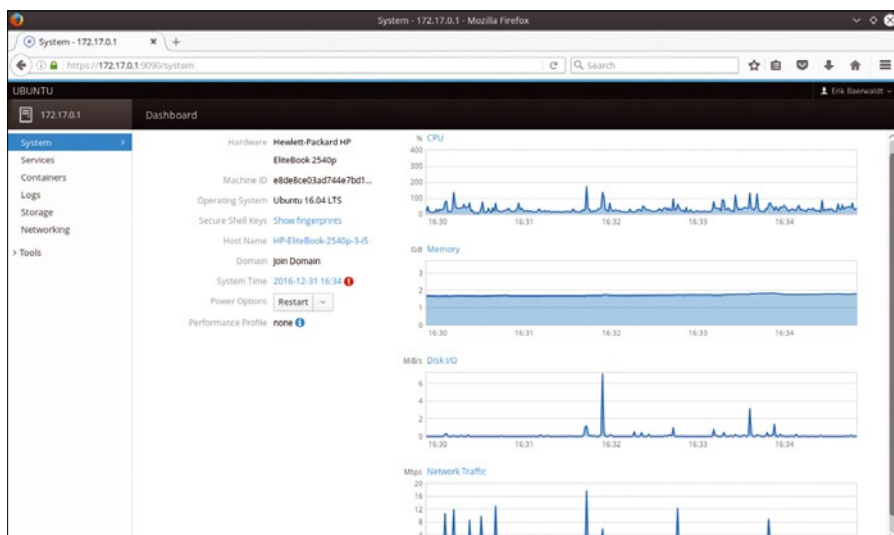


Figure 11: Red Hat's Cockpit provides a clear dashboard.

initially considers all types, even the inactive ones.

The Cockpit user further manages servers that can be found on the intranet by clicking *Dashboard* at the top of the main window and using the + button in the vertical center of the window. The server can now be integrated into Cockpit via its IP address. The management software then logs in with the user's authentication information with the new server and collects the necessary information from it. The server then appears under its IP address in the list of server systems waiting at the bottom of the window. Here, the user can call up the performance data of one of the servers via mouse click.

The user can also access the systemd units here via the *Services* menu item on the left of the dashboard – depending on the device. Cockpit divides the list into the categories *Enabled*, *Disabled*, and *Static*, which serves as an overview. At the top of the list view are the *Targets*, *System Services*, *Sockets*, *Timers*, and *Paths* buttons, from which you can very quickly identify individual units (Figure 12).

Regardless of a unit's operating status, after clicking on a unit, the user ends up on the unit in a log window, which displays the respective unit's log. This saves tediously searching for individual log entries in the journal.

Manager

To manage processes and daemons in Cockpit, you need to register the software with administrator rights beforehand. You

can start new processes from the dashboard in a terminal with *Tools | Terminal*.

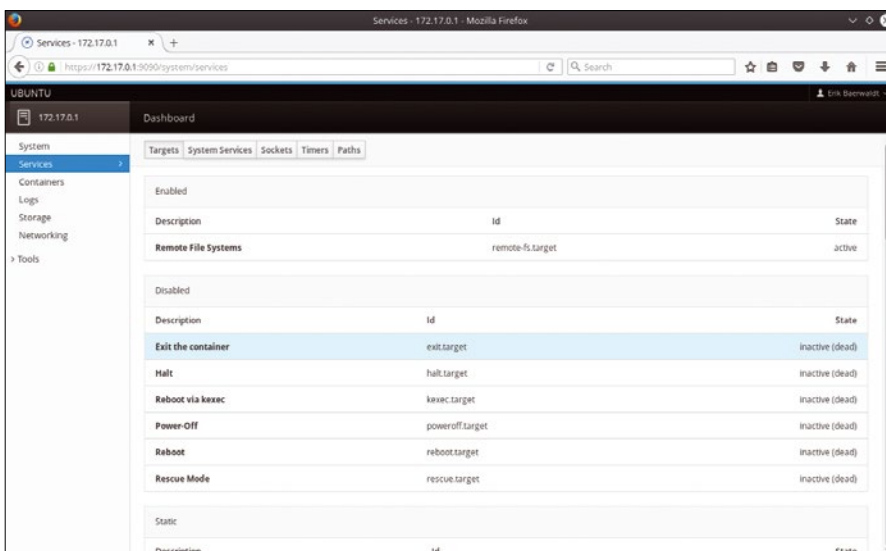


Figure 12: Cockpit displays all units in three-column tables.

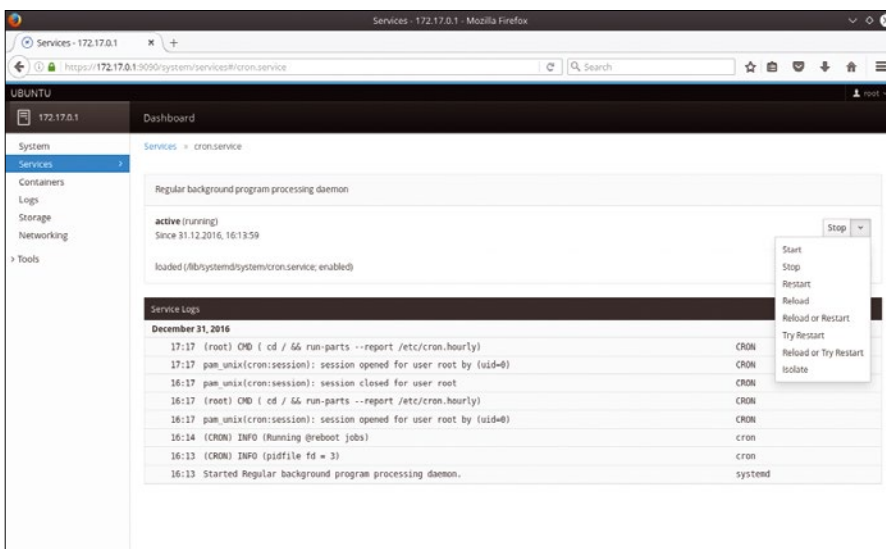


Figure 13: The Cockpit user can reload or disable units in the journal window.

Alternatively, you can select the desired unit from a list with the mouse. You will then see the unit's log entries and other information, including the path.

Two selection fields on the right provide various functions via mouse click: Here the administrator can enable, disable, or reload systemd units. These also allow a unit to be force started. The user can also modify the configuration file via the integrated terminal. In all, it's very easy to manage systemd processes using Cockpit (Figure 13).

Conclusions

The graphical tools for systemd meet the most diverse demands and enormously facilitate the work with processes and services. Basic knowledge of systemd is enough for users to control

their sessions. Users do not need to memorize cryptic parameters beforehand.

However, introductory documentation for explaining further options and configuration options beyond the mainly tabular unit lists is still missing from many of the fledgling projects. This shortcoming is particularly striking for Kcmsystemd. Here, you can configure everything down to the tiniest detail (as

is usual with KDE), but the software should provide a better explanation of a few technical peculiarities of the Journal and Logind options.

The GTK-based applications Systemd System Manager and Systemd Manager are much more rudimentary compared to the Qt-based tools and are limited to basic administrative tasks. They are more suitable for occasionally disabling and enabling processes.

Cockpit, the interface used in the browser for admins, breaks character a little. It is intended less as an administration tool for local systems and instead simplifies and centralizes complete server management on the intranet. The tool offers very easy access to systemd on the respective servers and allows the admin to manage the init system's services and processes from every workstation. ■■■

INFO

- [1] Development history of systemd: <http://Opointer.de/blog/projects/systemd.html>
- [2] Cockpit: <http://cockpit-project.org>
- [3] Webmin: <http://www.webmin.com>
- [4] Systemd System Manager: <https://anonscm.debian.org/cgit/pkg-systemd/systemd-ui.git/>
- [5] Systemd Manager: <https://github.com/mmstick/systemd-manager>
- [6] Kcmsystemd: <https://github.com/rthomsen/kcmsystemd>
- [7] Systemd-KCM: <https://www.linux-apps.com/content/show.php/Systemd-kcm?content=161871>
- [8] SystemdGenie: <https://rthomsen6.wordpress.com/2016/12/18/introducing-systemdgenie/>
- [9] Gnome Logs: <https://wiki.gnome.org/Apps/Logs>
- [10] Rust: <https://www.rust-lang.org>
- [11] SystemdGenie download: <http://download.kde.org/unstable/systemdgenie/>
- [12] Gentoo package: <https://packages.gentoo.org/packages/app-admin/systemdgenie>
- [13] Instructions for SystemdGenie: <https://github.com/KDE/systemdgenie>
- [14] Cockpit downloads: <http://cockpit-project.org/running.html>
- [15] Cockpit source code: <https://github.com/cockpit-project/cockpit>

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





The “removing systemd” experiment

Elective Surgery

The systemd service manager has been widely adopted by many Linux distros, so why would you want to remove it? The professor reveals why and how. *By Klaus Knopper*

Systemd [1], an essential component in major GNU/Linux distributions today, is used for system startup/shutdown as the first process, process group control, subsystem containers, logging, hardware detection and management, and communication interfaces between processes. All these features pretty much sum up the problem: systemd interconnects a lot of services that were formerly modular and independent programs, each one separately configurable. Squeezing all of them into a distro creates dependencies that greatly increase system complexity and make it difficult to look in a single place to find problems.

For example if, during the boot process, the system just hangs at a certain point, with no usable GUI or terminal login and no visible log messages, it could be that some dependent units are just “waiting for each other” (which is, from systemd’s point of view, not an error but a misconfiguration). Finding the culprit can then be difficult, especially when this happens in only one out of 100 (!) startup cases.

A lot of controversy is still flying around systemd concerning, on the con side, breaking Unix system principles of modularity and low interdependency of programs and, on the pro side, faster

startup through massive parallelization, “all-in-one” system design, and possibly more flexibility.

Personally, I would like to stay out of the controversy, but after running my own tests and having brief discussions with systemd’s main developer, it seems to me that systemd hasn’t been well tested in “real hardware” environments, especially in conjunction with old computers with small amounts of RAM and slow media like DVDs or USB (1 ... 2) flash disks, which are still accommodated in Knoppix. For virtualization and container-based systems on fast hardware, systemd is surely very useful.

You may know from last year’s CeBIT presentation that I’m working on a “Knoppix for Raspberry Pi,” which, on one hand, is less complicated than Intel/AMD hardware, because ARM architecture usually does not have a PCI bus system – instead using a “device tree” that defines the hard-wired addresses and I/O functions for the system – requiring less “hardware detection.” On the other hand, the Pi is – for modern standards – a very lightweight system than cannot be upgraded with more RAM; it doesn’t even have a SATA interface, and all data must pass through a controller with little available bandwidth. Therefore, I’d like to build a very light system for the

Pi that does not inherit too many requirements. Basically, I want this simple BusyBox SysVinit configuration, with a single Bash script for startup/autoconfiguration (`knoppix-autoconfig.sh`), one for the graphical desktop (`knoppix-startx.sh`), and one for shutdown (`knoppix-shutdown.sh`):

```
# /etc/inittab for busybox init z
(Raspi Knoppix)
::sysinit:/etc/init.d/z
knoppix-autoconfig.sh
::respawn:/etc/init.d/knoppix-startx.sh
::shutdown:/etc/init.d/z
knoppix-shutdown.sh
ttyAMA0::respawn:-/bin/bash
```

The last line is for the Pi2-specific serial interface, which should run a Bash shell for debugging purposes.

All commands for system startup and shutdown are included in the three scripts. Some are called one after the other (to wait for dependencies, if present), and some are parallelized in a way that is known to work well. One main goal is to reach a startup time of only a few seconds, from power-on to the ready-to-use desktop.

Unfortunately (for my purpose), the Raspberry Pi distribution of Debian now also has switched to systemd, so it will be

a challenge for me either to find a way to remove the systemd conglomerate of interconnected, parallelized services during startup and use the tiny SysVinit built into BusyBox instead, while also having to solve problems with Raspbian's systemd package dependencies when attempting to remove systemd, or to build a new distro from scratch.

In this article, I look at the first approach of removing systemd – at least as a boot system, if not more – and at keeping the system bootable and working.

Systemd – A Very Short Introduction

Before blindly attempting to remove a software package, it's a good idea to have a look at its purpose and function in the computer system. For SysVinit, this was rather simple: Depending on the runlevel (0 ... 6) given at the boot command line, start and stop scripts in `/etc/init.d/*` are called for each installed and activated subsystem (e.g., "load kernel modules," "mount hard disks," or "start up the network"). They are numbered in the order of their calling, so if a program depends on another program, it can simply be named in chronological order *after* its dependency.

Systemd comprises not only the startup process (process 1, the `systemd` daemon), but also other services that partly replace formerly independent programs or functions (Table 1). Whereas SysVinit has commands like `telinit`, `reboot`, and `shutdown/poweroff` for switching runlevels, systemd has a front end called `systemctl` with many options (Table 2) and that also supersedes the symlink-based SysVinit runlevel files in `/etc/init.d/*` → `/etc/rc${RUNLEVEL}.d/*`.

Systemd's `journald` replaces the formerly text-based logfiles in `/var/log` by context-sensitive binary files, and you need `journalctl` to read them. Important commands to know for debugging are shown in Table 3.

The `/etc/systemd/system/*` files override system defaults in `/lib/systemd/system/*`. The `/run/systemd/system/*` files are immediately generated and take precedence over defaults. Systemd's startup config files (Table 4) should be managed with the `Systemctl` GUI (Table 5) and not edited or copied directly (which you can still do, regardless of this recommendation). `Systemctl` even has equivalents for

TABLE 1: Systemd Suite Components

Daemon	Description
<code>systemd</code>	System startup/shutdown/control (like <code>init</code>)
<code>systemd-logind</code> [2]	Seat/session management
<code>systemd-journald</code>	Syslog, but different (context-based, binary logfiles)
<code>systemd-dbus</code>	Service process communication; formerly a separate program
<code>systemd-udev</code>	Hardware detection, device management, and driver/module loading; formerly a separate program
<code>systemd-networkd</code>	May replace <code>NetworkManager</code> ; does it exist yet?

TABLE 2: Controlling systemd

Command	Description
<code>systemctl start x.service</code>	Start program
<code>systemctl stop x.service</code>	End program
<code>systemctl restart reload x.service</code>	Stop+start program or tell it to apply configuration changes
<code>systemctl enable x.service</code>	Enable autostart at boot
<code>systemctl disable x.service</code>	Don't start at boot
<code>systemctl status x.service</code>	Show current running state of service
<code>systemctl list-units [--all]</code>	List [also not started] bootable services
<code>systemctl list-unit-files</code>	List all config files, regardless of their state

TABLE 3: Systemd Debugging Commands

Command	Description
<code>journalctl</code>	See all logs (also previous boots, formerly <code>cat /var/log/messages</code> in SysVinit)
<code>journalctl -b</code>	Current boot only
<code>journalctl -k</code>	Kernel messages only (<code>dmesg</code>)
<code>journalctl -b -u x.service</code>	Only current logs of a specific service

TABLE 4: Systemd Config Files

File Type Suffix	Content
<code>*.target</code>	Defines a synchronization point (like runlevels)
<code>*.service</code>	Start or stop services and define their order (like <code>S*/K*</code> scripts)
<code>*.socket</code>	Related to services: Defines an IPC socket for communication
<code>*.conf</code>	Configuration for systemd's own server processes
<code>*.device</code>	Additional device file setup not handled by Udev
<code>*.mount</code>	(On demand) mount point managed by systemd
<code>*.automount</code>	A mount point that's to be auto-mounted at some point
<code>*.swap</code>	Swap partition or file handled by systemd
<code>*.path</code>	Monitor path for changes to activate services/units
<code>*.timer</code>	Like <code>cron/at</code> , defines time-based events and actions
<code>*.snapshot</code>	Temporary file containing the system state at a certain point
<code>*.slice</code>	Part of <code>cgroups</code> definitions for controlling subsets of processes
<code>*.scope</code>	Created automatically; contains {D,K}-Bus information

TABLE 5: Managing systemd "Units"

Command	Description
<code>systemctl list-unit-files</code>	See Table 2.
<code>systemctl cat x.service</code>	Show unit content.
<code>systemctl show x.service</code>	Show the parsed/interpreted version.
<code>systemctl edit --full x.service</code>	Modify [all] UNIF files.
<code>systemctl list-dependencies [--all] x.service</code>	Show unit dependencies [recursively].
<code>systemctl daemon-reload</code>	Execute changes globally.

TABLE 6: SysVinit Equivalents in systemctl

Command	Description
systemctl poweroff	End all programs and power off
systemctl reboot	Restart system
systemctl rescue	Like runlevel 5 in SysVinit

the SysVinit reboot and shutdown commands (Table 6).

Even in this short introduction, you can imagine the full complexity of systemd. It is also important to know that systemd introduced its own pluggable authentication module (PAM) for session management in connection with systemd-logind. A process can ask about the owner of the console and graphical session by sending a message to D-Bus, which is then answered by systemd-logind, having learned about the user's login from the libpam-systemd component.

Experiment Prerequisites

For the experiments described here, I use a popular, lightweight distro – Lubuntu (the LXDE desktop variant of Ubuntu) – in the kvm virtualizer in order to reset everything to start after unsuccessful changes and be able to monitor every-

thing from a working host system.

Also, because the old SysVinit packages have disappeared from the main repositories, I prepared one sysvinit.deb contain-

ing /sbin/init, to replace systemd. This should be installed first; otherwise, Lubuntu will complain about missing “essential system packages.” The first step is always:

```
sudo dpkg --force-all \
-i sysvinit_2.88knoppix_i386.deb \
  initscripts_2.88knoppix_i386.deb \
  sysvinit-utils_2.88knoppix_i386.deb \
  sysv-rc_2.88knoppix_all.deb
```

Round 1: Just Remove systemd

For the first test, I wanted to see what would happen if I just removed systemd using the Apt standard packaging tool (Listing 1). On the plus side, this will free about 80MB of disk space Just kidding.

As expected, at command completion and reboot, the system comes up with a

black screen. Removing the quiet option and adding debug in the GRUB boot menu helps somewhat in understanding why. Part of the problem is that way too many services are just gone with systemd, because of soft dependencies set in their package metadata, which affects network configuration and most of the graphical user interface. The other part of the problem is covered in the analysis in round 2.

Round 2: Install the Magic Package and Then Remove systemd

To restore a working system right after installation, I'll introduce the no-systemd.deb package (Listing 2). Now this looks a lot better! Only systemd and a few other packages are removed – not the entire desktop system – because an empty package (no-systemd contains no files) is installed first, which fakes a systemd installation as the dependency for other packages (Listing 3).

libpam-systemd is clearly a dependency for systemd; rfc111 has a hard-coded dependency on a package named systemd, which is probably a mistake and unnecessary. Rebuilding the package without this dependency should help.

After removing systemd and libpam-systemd followed by a reboot, I still can't get past the point where the graphical subsystem should start up, because the root filesystem stays mounted read-only; therefore, Xorg can't open its logfile at /var/log/Xorg.0.log. Changing the root filesystem's mount mode to rw instead of ro in the bootloader's config is a quick workaround for now. A better fix would be to change the bootup procedure to remount/read-write at some point. Apparently, this is missing in my sysvinit.deb package.

Now I have a working desktop – kind of. After using it awhile, you will realize that some things are dysfunctional, like configuring the network with the graphical NetworkManager GUI.

Round 3: Install SysVinit over systemd and Do a Lot of Post-Configuration

The reason the previous round failed to produce a fully working system is that some desktop components require the session management introduced by

LISTING 1: Removing systemd with Apt

```
sudo apt --purge remove systemd
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be REMOVED:
  aptdaemon* dbus-user-session* friendly-recovery* gnome-disk-utility*
  gnome-software* gnome-software-plugin-snap* gnome-system-tools* gnome-time-admin*
  gvfs* gvfs-backends* gvfs-daemons* gvfs-fuse* hplip* init*
  language-selector-gnome* libnss-resolve* libpam-systemd* lubuntu-core*
  lubuntu-default-session* lubuntu-default-settings* lubuntu-desktop* lxsession*
  lxsession-default-apps* lxsession-logout* network-manager* network-manager-gnome*
  nplan* packagekit* policykit-1* policykit-1-gnome* python3-aptdaemon*
  python3-aptdaemon.gtk3widgets* rfc111* snapd* software-properties-gtk* synaptic*
  systemd* systemd-sysv* ubuntu-minimal* ubuntu-release-upgrader-gtk*
  ubuntu-standard* udisks2* update-manager* update-notifier* usb-creator-common*
  usb-creator-gtk*

WARNING: The following essential packages will be removed.
This should NOT be done unless you know exactly what you are doing!
  init systemd-sysv (due to init)
0 upgraded, 0 newly installed, 46 to remove and 0 not upgraded.
After this operation, 78.6 MB disk space will be freed.
You are about to do something potentially harmful.
To continue type in the phrase 'Yes, do as I say!'
?] Yes, do as I say!
...
```


LISTING 2: Setting Up no-systemd

```

sudo dpkg -i no-systemd.deb
Setting up no-systemd (2:1.0knoppix) ...
sudo apt --purge remove systemd
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages will be REMOVED:
  libnss-resolve* libpam-systemd* rfkill* systemd*
0 upgraded, 0 newly installed, 4 to remove and 0 not
upgraded.
After this operation, 10.8 MB disk space will be freed.
Do you want to continue? [Y/n]

```

systemd-logind, which is located inside the systemd package. After installing the SysVinit packages (which replace /sbin/init by SysVinit's version plus the necessary basic startup scripts), I also need to install the *systemd-shim* package to start session management and other services automatically around systemd clients. Also, the *policykit-1* package needs to be installed, and permissions for services like NetworkManager should be set to yes instead of no or `auth_admin*`, if the session can't be detected properly, as shown in the excerpt in Listing 4 from `/usr/share/policykit-1/actions/org.freedesktop.NetworkManager.policy`.

Also, newer versions of systemd require the cgroup mount point; otherwise, *systemd-shim* won't be able to start *systemd-logind*, so login sessions won't be reported to services that request information about the logged-in user. This would be a minimal subset of mounts that *systemd-logind* expects. I put the following commands in my own startup scripts:

```

mkdir -p -m 0777 /cgroup/cpu
mount -t cgroup cgroup -o cpu /cgroup/cpu
mkdir -m 1777 /cgroup/cpu/user

```

LISTING 4: Setting Permissions

```

<action id="org.freedesktop.NetworkManager.
  network-control">
...
  <defaults>
    <allow_any>yes</allow_any>
    <allow_inactive>yes</allow_inactive>
    <allow_active>yes</allow_active>
  </defaults>
</action>
... (same for all other needed actions)

```

LISTING 3: Building the no-systemd.deb Magic Package

```

Package: no-systemd
Essential: yes
Architecture: all
Depends:
  ${shlibs:Depends},
  ${misc:Depends},
Provides: libpam-systemd, systemd
Replaces: libpam-systemd, systemd
Description: Dependencies for being able to remove (most of)
  systemd

```

It Works!

After round 3, the system looks very much like a Knoppix installation, in which the startup system is run through SysVinit, with other systemd components still present to make programs happy that are hard-coded with systemd dependencies (e.g., on *systemd-logind* and the `libsystemd0` library). Still, LXSession pops up a confusing hard-coded message (Figure 1) that is produced if systemd is not listening on D-Bus. The best way I found to circumvent this problem is to remove the erroneous error message from the *lxsession* source and recompile the package.

Also, the `rfkill` tool has a confusing dependency on a package named (!) "systemd" instead of just the systemd provision in my *no-systemd* package. Downgrading `rfkill` or recompiling without the hard-coded dependency fixes this issue.

Conclusion and Other Resources

- Replacing the systemd startup from a systemd-controlled distribution by SysVinit or something else is comparably easy *if* dependencies are kept in place.
- Removing systemd-introduced dependencies causes side effects on session management that need to be resolved with some manual configuration work.
- The effort needed for maintaining a non-systemd-controlled distribution may increase with systemd's development progress (i.e., "it will never be finished"). Several HOWTOs are available for removing systemd partially or completely [3]-[5], recompiling packages without

systemd dependencies, or simply choosing a GNU/Linux distro alternative that uses a different system to manage software [6]. The "pinning" of packages, especially, seems to be a good idea to prevent semiautomatic upgrades from "updating" a new systemd version over the *no-systemd* package and thereby causing conflicts. ■■■

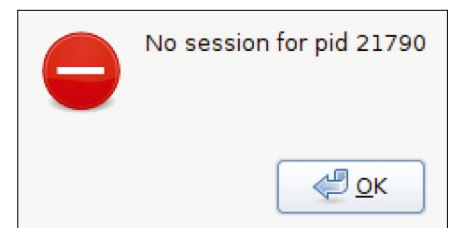


Figure 1: Systemd is not listening on D-Bus.

INFO

- [1] Systemd: <https://www.freedesktop.org/wiki/Software/systemd/>
- [2] Actually, *systemd-logind* will cause more trouble than *systemd* itself later.
- [3] Removing *systemd* from Debian: http://lkcl.net/reports/removing_systemd_from_debian/
- [4] Removing *systemd* from Ubuntu 16.04 and preventing its usage: <https://askubuntu.com/questions/779640/how-to-remove-systemd-from-ubuntu-16-04-and-prevent-its-usage>
- [5] Removing *systemd* from a Debian jessie/sid installation: http://without-systemd.org/wiki/index.php/How_to_remove_systemd_from_a_Debian_jessie/sid_installation
- [6] Systemd and alternatives: <http://without-systemd.org>

An elegant and simple Arch Linux-based distro

CHOOSE CHAKRA

KDE lovers can rejoice at Chakra Linux's beautiful and functional operating system. *By Nate Drake*

Chakra Linux is KDE – pure and simple. Its origins date back to 2006 when the late and great Jan Mette instigated the development of KDEmod – a light and modular way to bring KDE to Arch Linux. After releasing several versions, Jan decided to work on his own version of Linux, independent of Arch but inspired by “The Arch Way.” The Arch Way encompasses the overall philosophy behind Arch Linux, which focuses on simplicity, elegance, versatility, and being user-centric [1]. Thus the Chakra Project was born.

In March, the project released Chakra 2017.03, codenamed “Goedel” [2] after Kurt Goedel, the famous mathematician and philosopher. This release, like previous versions, combines KDE and the Qt framework. The Plasma desktop makes for a picturesque and customizable envi-

ronment. Although Chakra was developed under the auspices of Arch, it now stands as an operating system (OS) in its own right and has its own repositories [3].

Grabbing Goedel

The latest Chakra offering has a number of exciting updates. The 1.9GB Goedel ISO can be downloaded via BitTorrent or HTTPS from the Chakra Project download page [4]. Clear emphasis has been placed on selling the OS as it can easily be booted as a Live CD (Figure 1) or even within a VirtualBox virtual machine. Unlike most OSes, test driving in VirtualBox is made even easier by the inclusion of the Guest Additions on the DVD, which allow for extra features such as viewing the OS in full-screen mode.

Chakra's Heritage theme for Plasma has been overhauled to be easier on the eye; plasmoids now have a more unified

color scheme, and the panel is also a little more transparent to allow you to see wallpaper or windows in the background.

Most importantly, the Chakra project has updated the Calamares installer to version 3.0.1.91. This means Chakra can now be installed to Btrfs and LUKS-encrypted partitions. Sadly, Calamares still doesn't support RAID or LVM installations.

The installer itself can be started in a few minutes, by choosing a few basic options. You can enable encryption of the main and swap partitions via LUKS by checking a box and entering the password. By default, the system will log in to your desktop automatically without prompting for a passphrase, although you can change this in the installer options.

The system requirements are quite spartan. You will need 2GB RAM to run the installer or to load the system into VirtualBox; however, Chakra itself only needs 1024MB RAM once installed. An Intel Pentium 4 Processor with 64-bit support (or virtualized equivalent) is also required [5].

After installation, you may find that you still need to update a number of packages. (During testing, I found these amounted to around 450MB.) This is because Chakra implements a “half-rolling” release model for its repositories [6]. Fundamentally, this means the OS has a stable core of software with rolling applications on top of it.

Core software packages are updated according to predefined schedules. Unlike a garden-variety rolling release, there isn't one schedule for every single package; different groups of packages are updated at different times, ensuring a more stable system. Programs are updated following an application-based

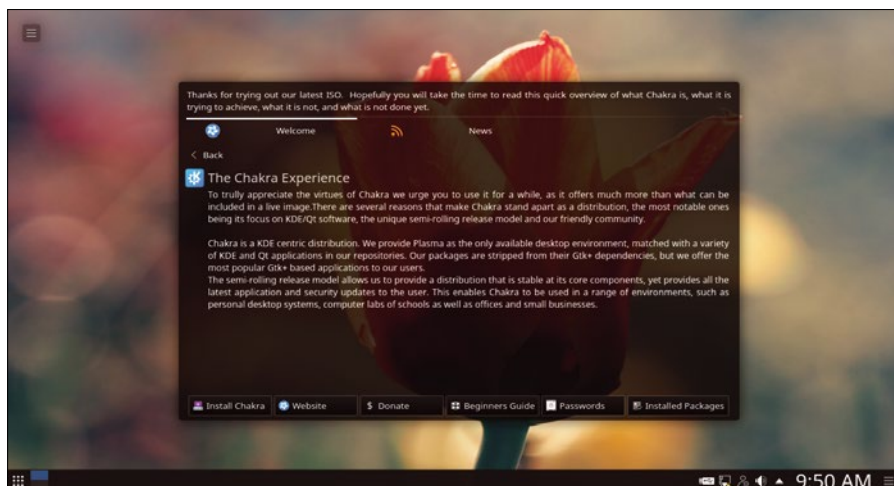


Figure 1: The LiveCD includes an excellent introduction to Chakra and the beautiful Plasma desktop.

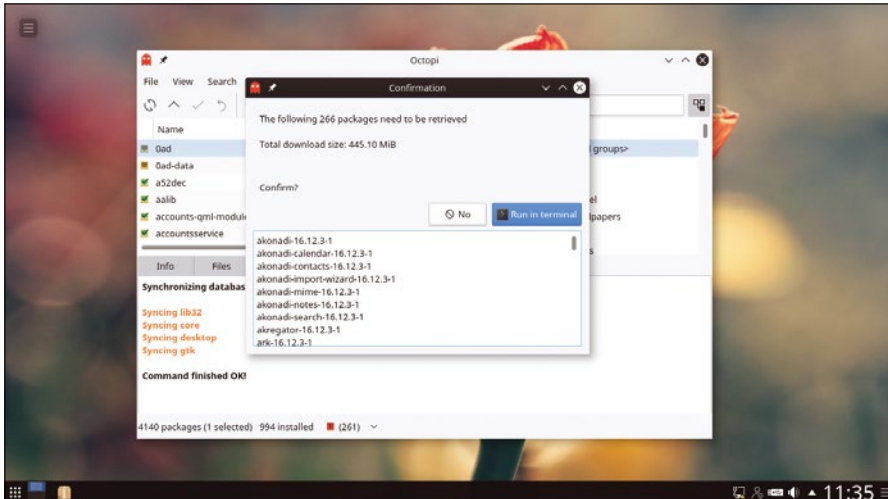


Figure 2: Use Octopi to run a system upgrade immediately after install to ensure your apps are up to date.

rolling release model, meaning you have access to the most recent version.

A major advantage of doing things this way is that you only have to install Chakra Linux via DVD once. The ISOs made available for download from the website represent current snapshots of the OS. However, you can also install any available updates via Octopi, a GUI front end for the Pacman package manager (Figure 2).

Although the Chakra project currently uses Pacman for package management, the developer’s ultimate plan is to replace it with their own package manager, Akabei. The program is under active development, but at the time of writing there’s no firm release date [7].

Goodbye Gtk

As a pure KDE/Qt framework, the Chakra project has not bundled Gtk, the GUI toolkit used by many Gnome applications. This also means that no Gtk-dependent applications, such as the Gimp image editor or the Thunderbird mail client, are preinstalled. However, Chakra does offer some respectable alternatives. The Chakra Project also maintains a gtk repository for more popular applications.

Chakra’s FAQ justifies doing things this way because including any one of these popular applications requires pre-installing around 50 Gnome dependencies for the sake of one or two programs. KDE also installs applications in a more transparent manner to /usr rather than to several directories as is the case with some Gtk programs [8].

Plasma Perfection

On first boot, Chakra will load the Plasma desktop. This will not contain many surprises for KDE veterans, who will be used to a single-click environment.

The upper-left corner of the desktop contains the *Desktop Tools* button. This allows you to edit the main theme, add handy widgets, and change your wallpaper as you see fit. You can also access these options by right-clicking anywhere on the desktop itself. The *Activities* feature deserves special mention as you can use it to run multiple desktops, each with their own widgets and running apps.

You can navigate your entire system from the Plasma desktop using KRunner (Figure 3), which is a very powerful tool that you can launch any time by holding down Alt+F2. You can then open an application or place in your system simply by typing the name. Click the configura-

tion on the left to customize further options such as searching bookmarks or performing sums.

Click the *Panel* button at the bottom right to fine tune your settings for panels, such as alignment and visibility. You can use the default *Always Visible* setting to ensure no panel intrudes on another’s territory.

You can also select the small arrow next to the system clock to display the system tray. This has several useful built-in applications, such as KDE Connect, which can connect your system to an Android device.

Minimized applications stow away nicely on the panel, simply showing their icon. Left-click once to maximize them again or right-click to see further options such as pinning the app to your panel or starting a new instance.

Take some time to browse the pre-installed applications by clicking the *Application Launcher* at the bottom left of the Plasma desktop.

Konsole Configuration

Despite the flowery desktop and aforementioned Octopi GUI front end for Pacman, command-line lovers can still enter terminal commands. You can do this either from KDE’s default Konsole terminal emulator or with the far cooler Yakuake. This drop-down terminal emulator functions in the same way as *Konsole*, but it can be launched by pressing F12 at any time, similar to the chat feature in classic first-person shooter video games.

Although I encourage you to find a KDE equivalent, Gnome users may want to pull up the Terminal when first installing

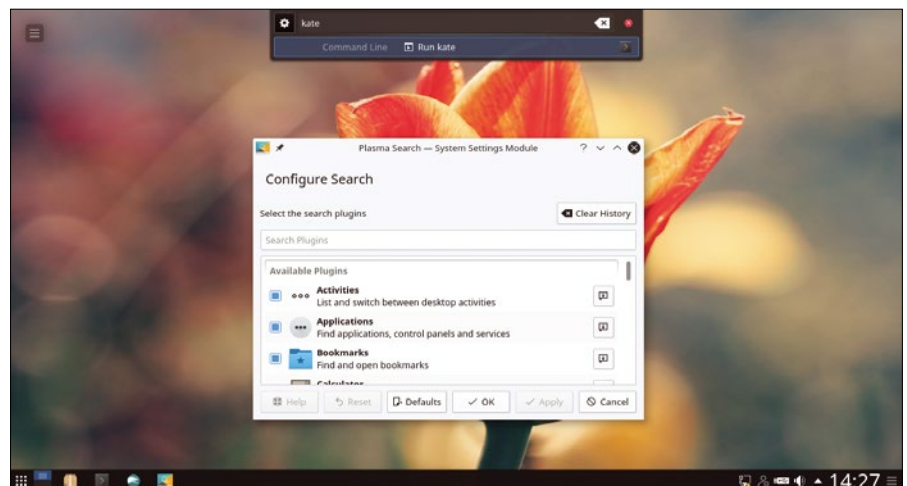


Figure 3: Use KRunner (above) to type the name of an app or location to open it. Click the settings wheel to fine-tune the search options.

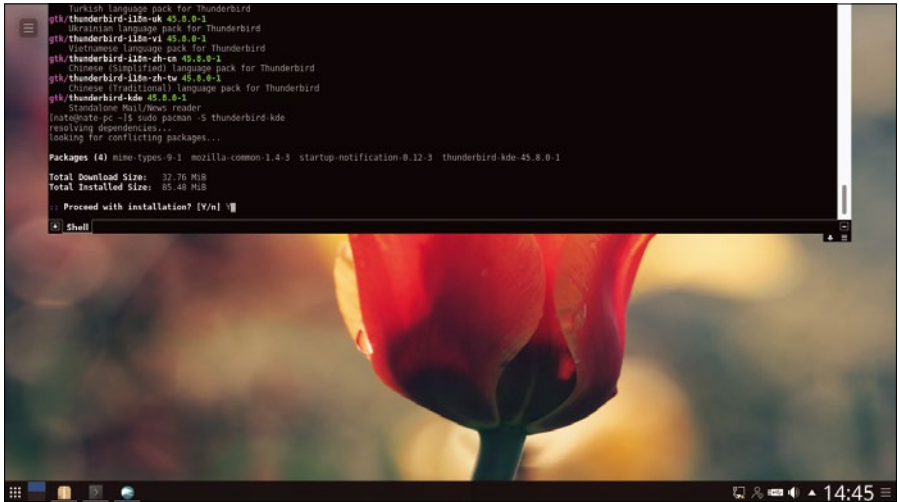


Figure 4: If you're unhappy with the default apps, access Pacman via Yakuake, Konsole, or Octopi to install your favorite apps.

Chakra to install some more familiar apps. Because Chakra follows a rolling release model, you should first update your packages post-install with:

```
sudo pacman -Syu
```

Once your system is updated, search for your favorite apps using

```
pacman -Ss <keyword>
```

For example, to search for Thunderbird (Figure 4), use the following command:

```
pacman -Ss thunderbird
```

This will list any package containing the text. You can install packages using:

```
sudo pacman -S <name>
```

Connecting Chakras

Although you can install others, Chakra's web browser of choice is the lightweight QupZilla, which uses the Qt Application Framework. QupZilla has excellent standard features that you'd expect in a web browser in that you can browse the web and bookmark pages; there's even a handy *Speed Dial* page for your top websites, similar to that used by the Opera browser.

Under the hood, QupZilla is privacy conscious (Figure 5), as it has built-in Adblock, and the default search engine is DuckDuckGo, which unlike some of the major players doesn't log your search results or IP address. This is also beneficial to the Chakra Project, which

shares in the revenue gained from users implementing DuckDuckGo itself.

Your web browsing experience is supplemented further with the KGet download manager, which allows you to

pause and prioritize downloads, as well as resume interrupted connections in most cases.

The Chakra wiki states that KGet also supports downloading files via BitTorrent [9], although I wasn't able to get this working in our tests. Fortunately, you can easily install a client, such as qBittorrent, using Pacman.

The default mail client is the lightweight and functional KMail, which should come as no surprise to KDE users. For Linux users who are used to Gnome, there's a KDE-specific version of Thunderbird available in the Chakra repositories, too.

Another noteworthy feature of Chakra is built-in support for instant messaging (IM). While the more well-known, Gtk-dependent Pidgin is notably absent, this is more than made up for by the integrated Konversation IRC client. Chakra also includes Kopete instant messenger, which supports other protocols such as

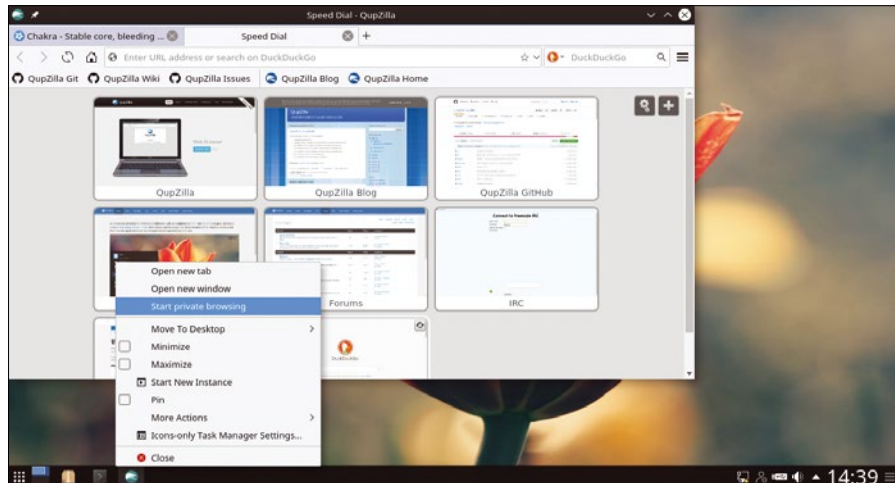


Figure 5: Right-click the Qupzilla icon in the panel and choose *Start private browsing* to leave no trace of your web activity.

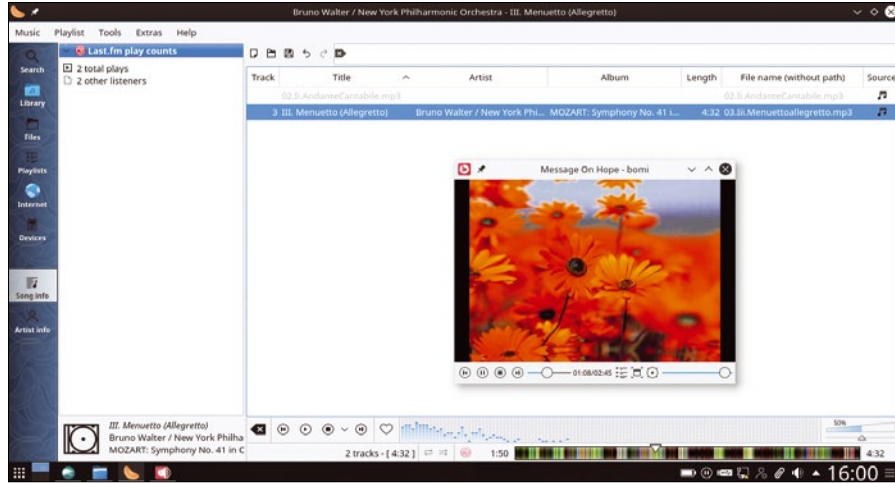


Figure 6: Use Clementine for music and bomi for video playback.

Jabber and Yahoo. Once you've configured an account, you can change your IM settings and see notifications from the system tray.

Singing Chakras

For music playback, Chakra uses the Clementine music player (Figure 6), which supports a wide variety of audio formats, such as MP3 and Ogg Vorbis, providing detailed track information. Playback can be minimized to the system tray, so you can listen to music as you use other applications if you prefer.

Those users who still prefer to buy music on CD will find that Clementine also supports playback from disk.

Video playback is provided by the bomi media player (formerly CPlayer), which is easy to use, and is based on mpv for Linux (Figure 6). Although not as fully featured as giants like VLC, bomi does support playback of common video formats, such as WMV and MP4, and can even play DVDs. It also has no trouble with common subtitle formats and can juggle audio tracks [10].

Head over to *Applications > Multimedia* to view Chakra's other bundled media-centric programs. These include K3b for disk burning and Kdenlive for video editing.

Productive Chakras

Click on *Applications > Office* to browse through Chakra's preinstalled productivity apps. Chief amongst these is the Calligra Office Suite (Figure 7), which includes the word processor Words and the equally imaginatively named spreadsheet software Sheets.

Chakra also bundles the useful vector graphics tool Karbon. The suite also includes the presentation software Stage, but you'll need to install this separately. If you prefer, you can also install the more well-known LibreOffice suite instead via *pacman*.

Working with PDF documents is a breeze; the Calligra suite apps can export files as PDFs. You can also view other PDF files using the preinstalled Okular document viewer (Figure 7).

The *Office* section also includes the *Storage Service Manager*, which allows you to set up and manage cloud providers such as Dropbox and Nextcloud. Any credentials you enter here are protected by KDE Wallet Manager (Kwallet), which stores sensitive data in an encrypted file. You can choose whether to encrypt using the classic Blowfish cipher or *gpg*.

Chakra Conclusions

The Chakra Project website cautions that its Keep it Simple Stupid (KISS) approach, where users can customize their desktop environment and apps, makes the OS unsuitable for people new to Linux. This said, they do have a comprehensive Beginner's Guide to help newbies get started [11]. The guide can also be accessed from the desktop of the Chakra LiveCD. Further help for those new to Chakra is available from the forums [12].

From using the Plasma desktop and navigating the preinstalled apps, it's clear that the Chakra Project has succeeded in their stated goal of creating an easy-to-use, clutter-free environment.

The installation process also works like a charm, although the half-rolling release model means that you must run updates early and often. On the plus side, this also means any OS upgrades are incorporated into the package manager, sparing you the trouble of going through the install process again.

Overall, while Chakra OS is undoubtedly a KDE purist's dream come true, it also offers a snappy and appealing experience for those willing to convert or for Gnome lovers willing to manually install their favorite applications. ■■■

AUTHOR

Nate Drake is a freelance journalist specializing in cybersecurity and retro tech.

INFO

- [1] The Arch Way: https://wiki.archlinux.org/index.php/User:Misfit138/The_Arch_Way_v2.0
- [2] Chakra 2017.03 "Goedel": <https://chakralinux.org/news/index.php?archives/195-Chakra-2017.03-Goedel-released.html>
- [3] Chakra repositories: https://chakralinux.org/wiki/index.php?title=Frequently_Asked_Questions#Can_I_mix_Chakra_and_Arch_Linux_packages.3F
- [4] Chakra download: <https://chakralinux.org>
- [5] System requirements: https://chakralinux.org/wiki/index.php?title=System_Requirements
- [6] Chakra half-rolling release model: https://chakralinux.org/wiki/index.php?title=Half-Rolling_Release_Model
- [7] Akabei: https://chakralinux.org/wiki/index.php?title=Akabei_Feature_Plan
- [8] Gtk-free Chakra: https://chakralinux.org/wiki/index.php?title=Frequently_Asked_Questions#Why_does_Chakra_want_to_be_free_from_GTK.3F
- [9] Downloading via BitTorrent: https://chakralinux.org/wiki/index.php?title=Finding_the_Application_for_the_Job
- [10] bomi: <https://github.com/xyloper/bomi/wiki/Feature-list>
- [11] Chakra Beginner's Guide: https://chakralinux.org/wiki/index.php?title=Beginner%E2%80%99s_Guide
- [12] Chakra forum: <https://chakralinux.org/forum/>

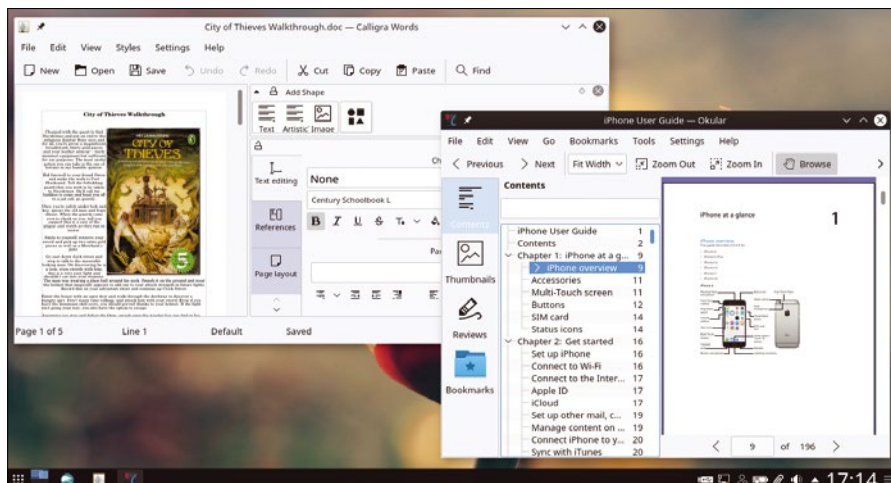


Figure 7: View PDFs in Okular; create and edit text documents using Calligra Words. The Calligra Suite also supports spreadsheets and presentations.



Securing and monitoring containers in enterprise environments

All Boxed Up

A recent flurry of activity in the container space raises several interesting questions about security among a number of operational aspects in the enterprise environment.

By Chris Binnie

Docker doubtlessly still reigns supreme in the container run-time space, but various industry projects mean that the Docker stronghold will almost certainly shift in one form or another over the coming months. The recent release of Docker Enterprise Edition (Docker EE) [1] shows that this fact hasn't escaped Docker, and to my mind, they should quite rightly take advantage of their market share and fully monetize their current standing.

The Docker EE offering advises you to meld all parts of your containerization and orchestration workflow together using one vendor to avoid sticking pieces of duct tape between the

components to integrate them. In their words: "An application-centric platform, Docker EE is designed [to] accelerate and secure across the entire software supply chain, from development to production running on any infrastructure" [1]. More easily digestible details can be seen in Figure 1.

Just One Moment

My interest from a DevSecOps perspective is security, and in Figure 1 you can

see that image scanning for common vulnerabilities and exposures (CVEs) [2] is indeed bundled with the EE flavor of Docker. However, that is not so for the less feature filled Docker Community Edition (Docker CE) [3], which is promoted for developers and small teams. However, it is thankfully available for free as a preview for a period and for those using a paid plan for private repositories. As you can see in Figure 2, it's highly efficacious.

Lead Image © Franck Boston, Fotolia.com

AUTHOR

Chris Binnie's new book *Linux Server Security: Hack and Defend* shows how hackers launch sophisticated attacks to compromise servers, steal data, and crack complex passwords so that you can learn how to defend against such attacks. In the book, he also talks you through making your servers invisible, performing penetration testing, and mitigating unwelcome attacks. You can find out more about Linux servers and security at his website (<http://www.linux-server-security.com>).

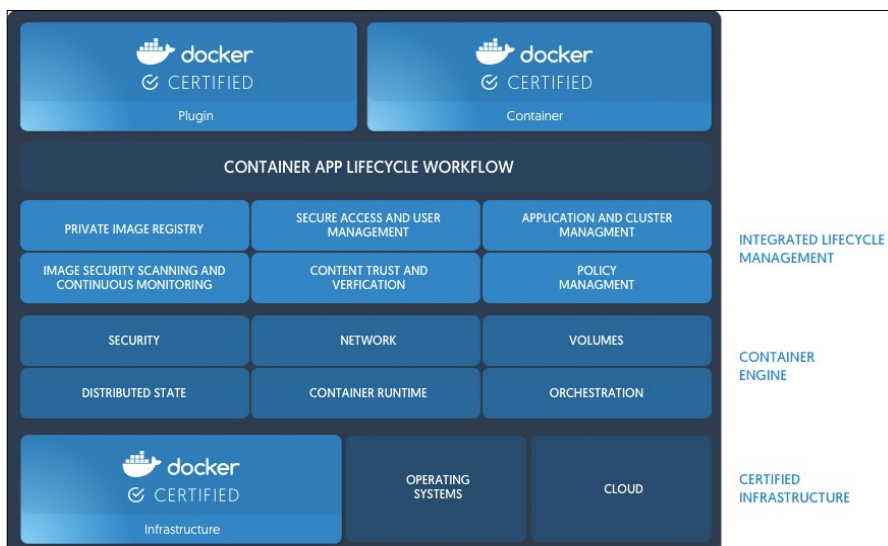


Figure 1: Docker EE (source: Docker for the Enterprise [1]).

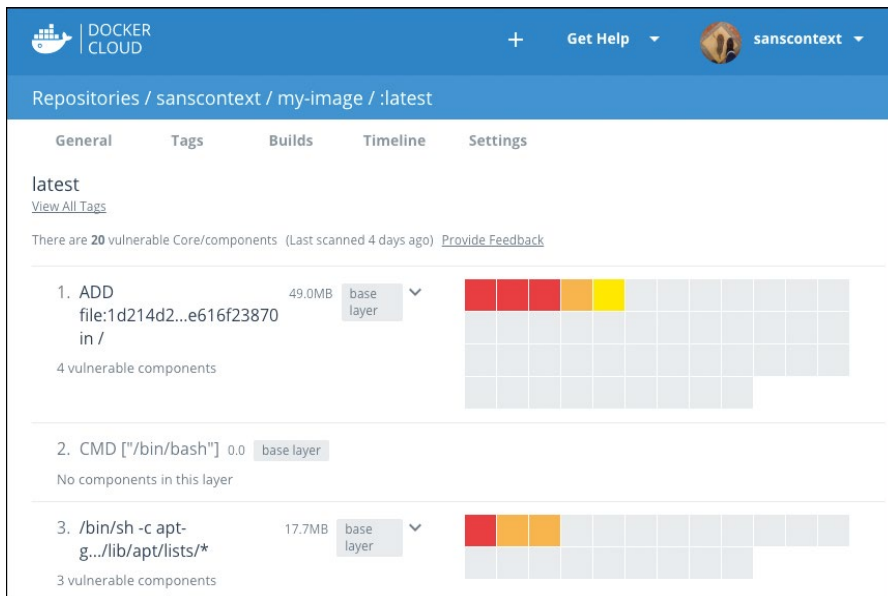


Figure 2: Docker offers image scanning for CVEs (source: Docker Security Scanning [4]).

The Docker site states: “During the free period, Docker Security Scanning scans the three most recently updated tags in each of your private repositories. You can push an update to an older tag to trigger a scan. The scan runs on each new image push and updates the scan results when new information comes in from the CVE databases” [4].

Although this functionality is undoubtedly very much required, unfortunately, it’s merely one critical aspect of securing your containers, and you might be forgiven for falling into a false sense of security.

Not Quite as Simple as That

Although I have no doubt that Docker EE offers a sophisticated security posture using a greater refined set of security principles, such as verifying images have come from a trusted registry (and not an unknown registry populated with a selection of images with nefarious intent), those businesses feeling satisfied that all their bases are covered by just running CVE scans are definitely missing a trick.

In simple terms, the three major areas to worry about when securing your containers and continuous integration/continuous deployment (CI/CD) pipelines are as follows:

- **Images.** It’s imperative that images are signed so that you know precisely which version of an image you are pulling from your trusted registry (with which you are authenticating and methodically logging each trans-

action). Once you know what you’re dealing with, you can scan that image for common vulnerabilities across multiple third-party feeds.

- **Run Time.** Another critical area is your container run-time security. For example, what if a container suddenly spawns an anomalous process or tries to create a volume mount when it has never tried to do that in the past? Monitoring these changes and automatically mitigating the level of damage that they can cause or alerting on-call staff to any such changes should they occur in production makes running an estate reliably much easier.
- **Host Security.** A third aspect often overlooked is access to the run-time daemon itself, which integrates into the host’s kernel with impunity. Take the popular Docker run command as a case in point. Think of a Jenkins job firing a `docker run` – the daemon acts as the superuser, the root user, on the host. Even the suggested Docker system group offers no protection because *any* access to the Docker daemon is effectively relinquishing super-user permissions on the host. The terrifying result of a successful, sophisticated compromise is that you would lose all the containers on that host followed by the host itself.

Approaching your container security from a traditional security perspective for a moment, remember that effective mitigation is all about layering your protection; providing defense-in-depth.

There is some solace in that containers are declarative. In other words, a Dockerfile might be treated as a shipping manifest of sorts (pun intended).

You can use that predefined shipping manifest to describe how a container *should* work and, following that, how it was intended to interact with your estate. With some forethought, you can translate this key information into a useful ruleset.

Thankfully, inside containers, developers mostly use popular software components, so these rulesets can be relevant to many decoupled containers, and it can be much easier to define those that might have been required for old-school bare metal servers running many different services simultaneously.

Clearly, a number of security challenges are introduced by running containers. However, undoubtedly, modern containers are fantastic for deploying software. Containers have been fully embraced by developers because of their ease of use, hence the unparalleled popularity across the industry relative to that of former technologies used in containers. That said, one fact has been repeated that cannot be denied ...

Containers Are Not Virtual Machines

For good reason, developers love containers for the convenience of packaging dependencies together into a portable unit. However, not unsurprisingly, they commonly expect containers to work like virtual machines (VMs) from a security standpoint. This is definitely not helpful to either party because they are distinctly different animals.

As vigilant security becomes increasingly important in this brave new containerized world, you need to accept that applications will always be compromised, and you therefore need a way of defending against these threats.

After all, containers are not like Solaris Zones or BSD Jails from the past, which are relatively prebuilt and uniformly defined. Modern containers are instead made up of a collection of configs that the Linux kernel has successfully made available over recent years. Their configuration is not set in stone, which makes them more flexible; of course, this complexity adds other security challenges.

Behind the scenes, containers now comprise namespaces and control

groups (cgroups), which are kernel primitives, or the system's Lego blocks, from which you build upward.

Kernel namespaces offer a running process a defined amount of visibility of a system (e.g., a specifically grouped set of other processes or a local routing table of its own). Separately, control groups offer a sys admin granular control over what that process can use.

Consider that namespaces are just a simple form of virtualizing system resources, and cgroups simply control, for example, how much CPU or RAM a process can use.

By combining the functionality of namespaces and cgroups, you can build a type of isolation between your containers and host (and indeed intracontainer isolation), but you certainly need to provide additional hardening on top, such as a Mandatory Access Control (MAC) system (e.g., SELinux on Red Hat derivatives or AppArmor on Ubuntu and Debian). It's no understatement to say that Red Hat has put a lot of effort into SELinux for containers, and for good reason.

Did You Lock the Front Door?

You might have heard of *Secure Computing Mode* (known as seccomp), with which you enforce a limit on a process' abilities or on syscalls that are needed for a process to request a service from the kernel. The addition of these profiles is also possible in Kubernetes [5], currently the most popular container orchestration engine with an impressive 15-year production provenance from Google.

Creating a seccomp profile allows you to define every available syscall in a list, and after lots of testing at Google, you can take some comfort in the knowledge that the default profiles that enforce these tricky-to-fathom limits will help keep most applications available and not break anything horribly.

From a Docker perspective, a number of syscalls are run by default, but many are also switched off. At one point, their default profile blocked more than 50 syscalls with the promise that things would keep running as expected.

Want to Eat Your Cake Too?

In addition to fine-grained seccomp profiling, Docker cleverly integrates with the kernel's *capabilities* [6].

How to switch on only the permissions you require and not get bamboozled with the need to allow `cap-add=all`, which opens the floodgates to the host on which a container runs, is definitely worth understanding. I find myself referring to Linux capabilities [7] continually these days.

Be warned, however, that even worse than opening up lots of kernel capabilities for your containers is the dreaded (and I've heard it called "lazy") *privileged mode*. With privileged containers, in addition to unleashing system capabilities with `cap-all=all`, you lose all the limitations enforced by the device cgroups controller.

In terms of protecting your host from a container running in privileged mode, you might say that you're effectively leaving your front door swinging open in the wind, never mind forgetting to lock it.

As for the future of host security, it's common knowledge that Google has been working hard at spawning containers as the non-root user (and the Gnome project apparently has, too, from a desktop perspective). By all accounts, however, it's not a trivial undertaking. In case you are wondering, the popular Google Chrome browser sandboxes its Browser tabs in a not too dissimilar (and very clever) way that containers run. The surprising crossover between desktop and container spaces shows how important fine-grained kernel access for processes is.

The future is sharing VMs between distinct tenants (different customers) with fully assured safety and little fear of a compromise taking down your entire customer base at once. Kubernetes itself is not far from orchestrating containers within hypervisors for safety.

Freedom of Choice

One change that might have crept in under your radar, among those sometimes difficult-to-follow changes in the industry I mentioned earlier, is that Kubernetes has moved to allowing you to state your container runtime of choice.

Kubernetes reminds you that there's more than one way to launch a container – namely, using the `cri=` switch (i.e., for the container run-time interface, CRI). The CRI means you are no longer tied to the Docker runtime but can choose other run-time engines, such as rkt or the soon to be a la mode

runtime that will comply with OCI standards (i.e., `cri-o`).

Conveniently, the container network interface (CNI) improvements also integrate nicely with the chosen container runtime's networking in Kubernetes. The project's page can be seen on GitHub [8].

Although easy to miss, progress is well underway on other lesser known projects to bestow more run-time choice. You might have a look at the README files for rktlet [9] and frakti [10] on their respective GitHub sites.

I recommend having a quick read of both pages for some valuable insight into what you might be working with very soon. Each choice is sophisticated and ultimately could have different agendas.

The End

With such a high feature churn rate in the DevSecOps area, it can be tempting to run your CI/CD pipelines with the latest bleeding-edge features. Needless to say, that path is likely to be a recipe for disaster. Not only will you constantly be chasing the leader, but also introducing costly stability issues and unwelcome attack surfaces about which you might not even be aware.

Testing is clearly of paramount importance. ■■■

INFO

- [1] Docker EE: <https://www.docker.com/enterprise-edition>
- [2] Common vulnerabilities and exposures: <https://cve.mitre.org>
- [3] Docker CE: <https://www.docker.com/community-edition>
- [4] Docker Security Scanning: <https://docs.docker.com/docker-cloud/builds/image-scan/>
- [5] Kubernetes: <https://kubernetes.io>
- [6] Run-time privilege and Linux capabilities: <https://docs.docker.com/engine/reference/run/#runtime-privilege-and-linux-capabilities>
- [7] Linux capabilities: <http://man7.org/linux/man-pages/man7/capabilities.7.html>
- [8] OCI-based implementation of the Kubernetes container runtime interface: <https://github.com/kubernetes-incubator/cri-o>
- [9] rktlet: <https://github.com/kubernetes-incubator/rktlet>
- [10] frakti: <https://github.com/kubernetes/frakti>



RISE HIGHER

EACH ISSUE OF DRUPAL WATCHDOG OFFERS TOOLS, TIPS, AND BEST PRACTICES FOR BETTER DRUPAL WEBSITES.

NOW
PUBLISHED
4 TIMES
PER
YEAR!



Renew or subscribe now!

SUBSCRIPTIONS NOW AVAILABLE WORLDWIDE!

Visit <http://drupalwatchdog.com/subscribe>

Equipping Alexa with self-programmed skills

More than a Word

Asking Alexa only for built-in functions like the weather report gets old quickly, and add-on skills from the skills store only go so far. With a few lines of code, Mike teaches this digital pet some new tricks. *By Mike Schilli*

Okay, I admit, I don't like getting up from the sofa to check out something on the Internet if my phone is out of reach. But who does? In the household of the future, a device like Amazon's Echo will be around to help you with things like this in the form of a tin can in the living room. The Amazon Echo Dot, which

looks much like a hockey puck (Figure 1), is the successor to the tin-can-like Echo device and offers the same voice interface at a lower price point.

Amazon's dream, of course, might be that customers at home will just shout "Alexa, order toilet paper," to send the language assistant scurrying off to place the order with the Internet discounter, who then quickly dispatches a drone to deliver the much-needed household item to the anxiously waiting consumer.

Supposedly Safe

While the device constantly listens in the room, waiting to respond to the spoken word, it is in a kind of twilight state most of the time and does not forward incidental noise picked up by the microphone to the Amazon Cloud for analysis, according to the



Figure 1: The Echo Dot Bluetooth speaker.

provider. Instead, the voice assistant waits for the user to speak what is known as the wake word, usually "Alexa," before activating the speech recognition-enabling connection to the Amazon Cloud. Instead of "Alexa," "Amazon," or "Echo," Star Trek fans can switch the wake word to "Computer" (Figure 2) and then talk to the assistant just as Captain Jean-Luc Picard spoke to the on-board computer on the bridge of the starship Enterprise.

MIKE SCHILLI

Mike Schilli works as a software engineer in the San Francisco Bay area of California. In his column, launched back in 1997, he focuses on short projects in Perl and various other languages. You can contact Mike at mschilli@perlmeister.com.



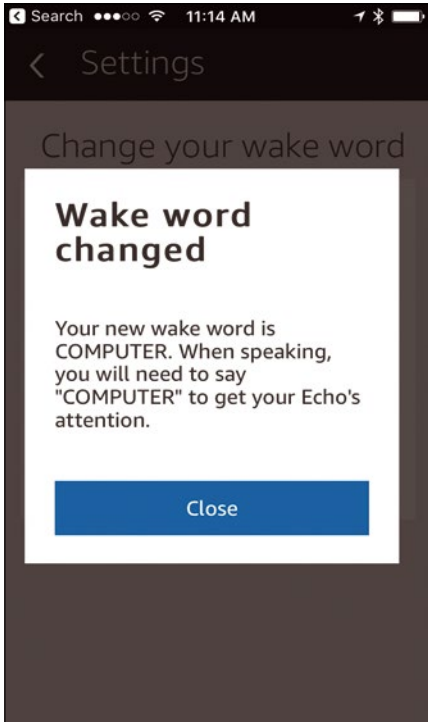


Figure 2: Star Trek fans can now wake up Alexa by saying "Computer."

Teaching Alexa New Tricks

Out of the box, Alexa can answer simple questions. Skills [1], which are free thus far (but might well switch to the app payment model sometime in the future) are available from the Skills Store [2] (Figure 3). Skills are basically software packages programmed and published by third parties, much like apps in the App Store. From reading current share prices in a customized stock portfolio ("Motley Fool") through playing a game of Jeopardy, to controlling lamps in your smart home, you have a lot to choose from.

To teach Alexa a skill, you activate one in the Alexa smartphone app or simply speak into the microphone (e.g., "Alexa, enable The Fool"). A short time later, Alexa is familiar with the stock market via the Motley Fool skill and answers questions like "Ask The Fool how is the market doing today" or "Ask The Fool to add Microsoft to my watchlist."

If your fingers are itching at the thought of teaching Alexa to dance to your own tune, you can visit Amazon Developer Services [3] to set up an Alexa account, which is separate from Amazon's AWS business, designed for the development of Android apps (Figure 4). After approving the agonizingly long terms of service, click on the *Alexa* tab and then on *Alexa Skills Kit* [1] to view

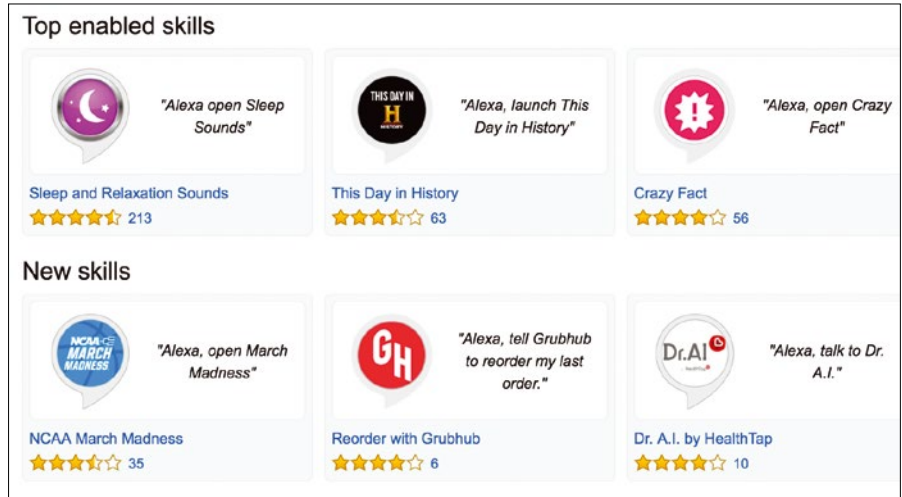


Figure 3: The most popular skills in the Alexa Skills Store on Amazon.

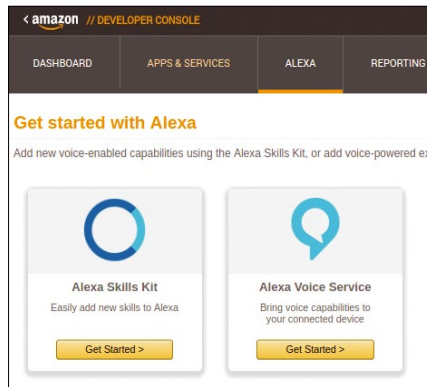


Figure 4: Developing new Alexa skills on developers.amazon.com.

a list of home-grown skills programmed thus far (Figure 5), along with a button that reads *Create New Skill*.

Snapshot Output On-Call

As a practical example, I want Alexa to report on demand the title and date of the

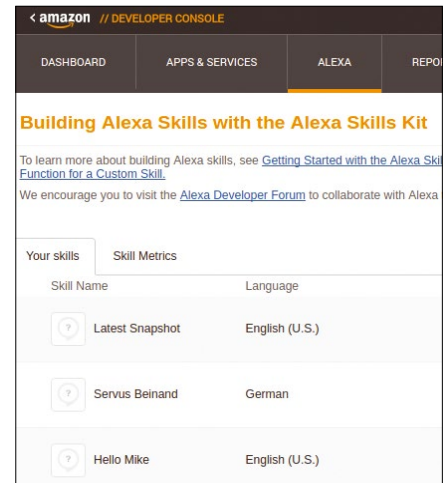


Figure 5: User-developed Alexa skills.

latest edition of the Snapshot programming column. To do so, Alexa reads a JSON file `articles-en.json` from my site at *Perlmeister.com* behind the scenes. The file lists all the issues in reverse chronological

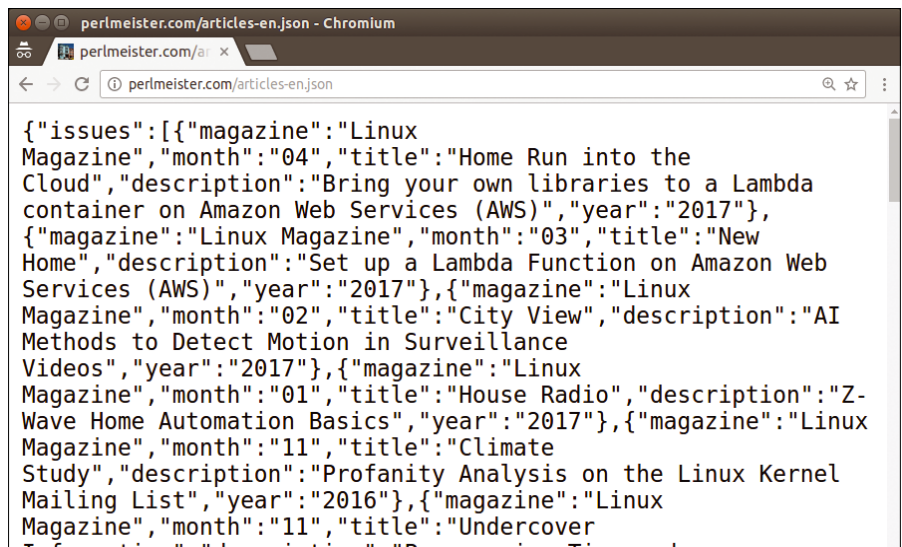


Figure 6: The Snapshot articles in JSON format as a data source for Alexa.

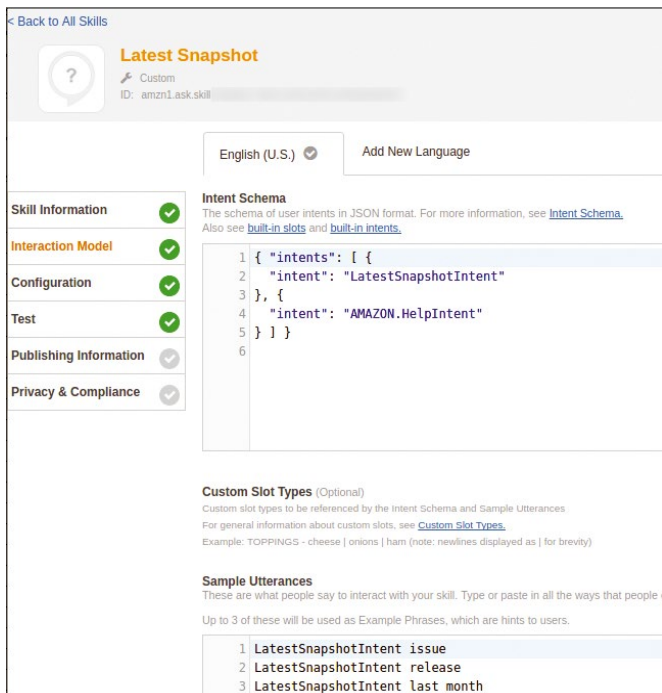


Figure 7: Intent and utterances in the Alexa Skill Setup.

order (Figure 6), picks up the first entry from the long list, extracts the month and year, and passes the two values back to the language processor, which in turn announces them to the user via the Echo Dot.

If I say, “Alexa, ask Latest Snapshot for issue,” then Alexa forwards the question to the Latest Snapshot skill, which then retrieves and processes the data and returns an answer such as *04 2017, Home Run into the Cloud* (if the April issue happens to be the latest published *Linux Magazine*).

The developer defines the name of the newly taught skill to which Alexa delegates requests below `invocationName` in the web flow for new skills. Several words are allowed, but they must not contain keywords used elsewhere (e.g., “Alexa”).

LISTING 1: IntentSchema.json

```
1 { "intents": [ {
2   "intent": "LatestSnapshotIntent"
3 }, {
4   "intent": "AMAZON.HelpIntent"
5 } ] }
```

LISTING 2: SampleUtterances.txt

```
1 LatestSnapshotIntent issue
2 LatestSnapshotIntent release
3 LatestSnapshotIntent last month
```

skill, you need to enter the data from Listings 1 and 2 [4] into the *Interaction Model* box (Figure 7).

Listing 1 uses JSON format to define the intents identified later on, and – in addition to `LatestSnapshotIntent` – also defines a generic Help menu, which Alexa recites if the user says “Help.” Listing 2 assigns various spoken sentences to each intent.

My Old Pal, Lambda

The next screen in the development flow asks for the *Endpoint* of the request – that is, the web service to which the language assistant should forward the request. In the simplest case, this is a Lambda function in the Amazon Cloud, whose configuration was discussed in a previous edition of the Snapshot column [5]. You’ll need to

Utterances and Intent

Speech processing gurus divide user communication with the device into utterances and intent. The user might say, for example, “latest release” or “newest issue” (utterance), thus revealing the same intent in both cases – that is, calling the function that outputs the month and year of the current Snapshot in the Latest Snapshot skill. In the web flow of the newly created

enter the Lambda function’s Amazon Resource Name (ARN) in the text box shown in Figure 8, for which you can use cut and paste.

The developer may write the Lambda function on AWS either in Java or in JavaScript in a Node.js environment; the newly released Alexa SDK [6] makes this very easy. Because the Lambda server won’t have the `alexa-sdk` Node.js package installed, developers are required to bundle both the script in Listing 3 and the `alexa-sdk` distribution by running the command:

```
npm install --save alexa-sdk
```

After stepping into the newly created `node_modules` directory, copy Listing 3 there as well, name it `index.js`, and zip up the whole enchilada with

```
zip -r upload.zip index.js alexa-sdk/
```

to then upload it to a newly created Lambda server (Figure 9).

The handlers for the defined intents to which the web service responds are found in lines 5-12 in Listing 3 as an associative array. For the new `LatestSnapshot` skill, line 7 calls the `getData()` function (lines 14-32) and passes the response object to it so that the function can send the response it finds directly back to the client.

Asynchronous HTTP

Line 15 then uses the `http` package included with Node.js and employs its `get()` method (line 16) to pick up the JSON salad from the Perlmeister website. In line with the asynchronous program flow in Node.js code, lines 19 and 23 register handlers for incoming data chunks and completion of the HTTP

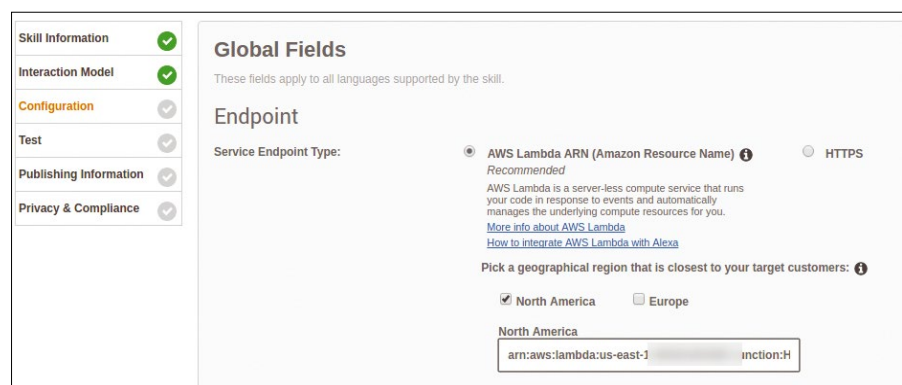


Figure 8: In this case, the endpoint of the request is a Lambda function.

transfer. The former glues together the bits and pieces that arrive to create the json_string; the latter launches a JSON

parser, which returns a populated data structure derived from the string. In production operations, the code should

include error handlers, so that Alexa can inform the user via voice output what went wrong.

LISTING 3: index.js

```

01 // Alexa helper to fetch latest snapshot issue
02 // Mike Schilli, 2017 (m@perlmeister.com)
03 var Alexa = require('./alexa-sdk');
04
05 var handlers = {
06   "LatestSnapshotIntent": function () {
07     getData(this);
08   },
09   "Unhandled": function () {
10     this.emit('ask', "Hello", "Hello");
11   },
12 };
13
14 function getData(response) {
15   var http = require('http');
16   http.get("http://perlmeister.com/articles-en.json",
17     function(res) {
18     var json_string = '';
19     res.on('data', function(data) {
20       json_string += data;
21     });
22
23     res.on('end', function() {
24       var snapshots = JSON.parse(json_string);
25       var latest_issue = snapshots.issues[0];
26       var answer = latest_issue.month + " " +
27         latest_issue.year + " " +
28         latest_issue.title;
29       response.emit(":tell", answer);
30     });
31   });
32 }
33
34 exports.handler = function(event, context, callback) {
35   var alexa = Alexa.handler(event, context);
36   alexa.registerHandlers(handlers);
37   alexa.execute();
38 };

```

IT Highlights at a Glance



The collage features several overlapping magazine covers and news snippets. On the left, there's an 'ADMIN HPC' cover with a red header. In the center, a 'LINUX UPDATE' cover with a blue header. On the right, a 'RASPBERRY PI GEEK' cover with a red and white header. Other smaller snippets include 'ADMIN Update - Hotest Links', 'FEATURED ARTICLES', 'FURTHER READING', and 'Apps World'.

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

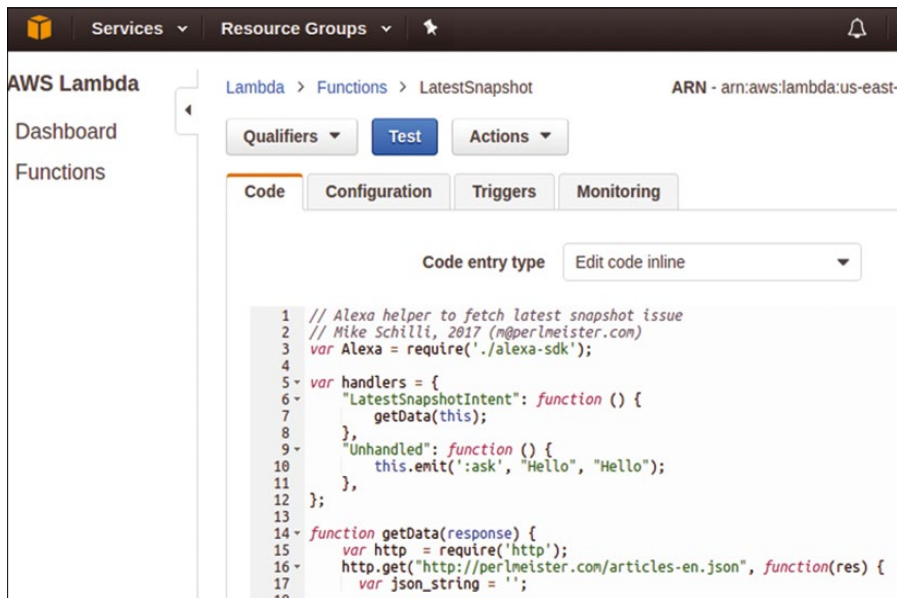
Linux Update • ADMIN Update • ADMIN HPC • Raspberry Pi

Keep your finger on the pulse of the IT industry.

Admin and HPC: www.admin-magazine.com/newsletter

Linux Update: www.linuxpromagazine.com/mc/subscribe

Raspberry Pi: www.raspberry-pi-geek.com/mc/subscribe



```

1 // Alexa helper to fetch latest snapshot issue
2 // Mike Schilli, 2017 (m@perlmeister.com)
3 var Alexa = require('./alexa-sdk');
4
5 var handlers = {
6   "LatestSnapshotIntent": function () {
7     getData(this);
8   },
9   "Unhandled": function () {
10    this.emit(':ask', "Hello", "Hello");
11  },
12 };
13
14 function getData(response) {
15   var http = require('http');
16   http.get("http://perlmeister.com/articles-en.json", function(res) {
17     var json_string = '';
18   });
19 }

```

Figure 9: The endpoint web service in Lambda as JavaScript code.

Because the entries in the JSON document are in reverse chronological order, line 25 just picks the first element from the array below `issues`. The code then extracts the month and year, as well as the title, from the data structure.

Using the response object's `emit()` method with the `":tell"` option, line 29 sends the reply to the speech processor. To register the previously defined handlers, the script sets the `exports.handler` attribute to a function that first obtains a new Alexa object, calls `registerHandlers` on it, and then runs `execute` to start the control flow.

For the Lambda function to accept

Alexa events, it needs the *Alexa Skills Kit* option set, which developers can accomplish by activating it under the *Triggers* tab in the web UI. The user selects this by clicking on the grey-ish dashed empty box (Figure 10), from which an arrow points to the orange shutters of the Lambda function. After this, back in the configuration flow of the Alexa skill, the *Test*

section now has an orange button, set to enabled, with a message stating that the skill has been enabled for the user's personal account. If the user now says, "Alexa, ask Latest Snapshot for release | issue | last month", Alexa answers "04 2017, Home Run into the Cloud" after a short pause for reflection, because that is indeed the date and title of the latest published edition of this column, as I write this in March 2017.

If anything goes wrong, you can check the Alexa log on the phone app to discover what Alexa actually understood and then browse the logfile of the Lambda function to see whether the code crashed or logged any errors.

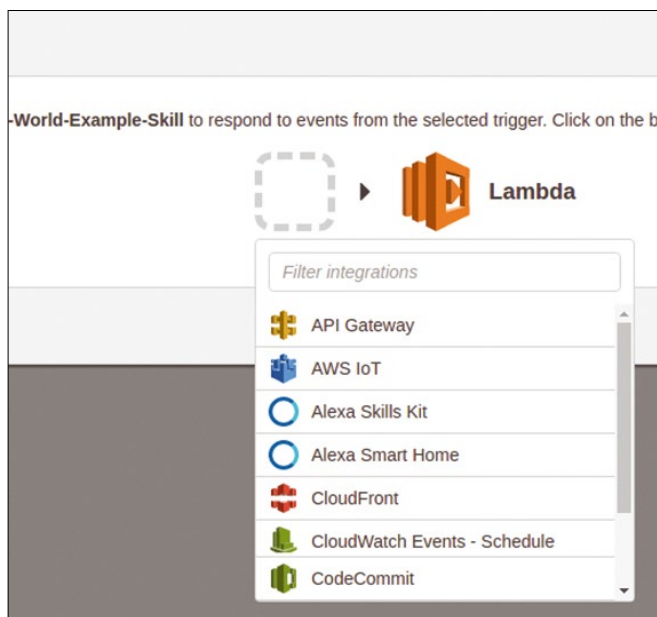


Figure 10: A selection of Alexa events.

Skills can support multiple languages, so the endpoint must respond in the language in which the question was asked. Currently, Alexa supports English, both UK and US variants, and German. Developers can define multiple intents and utterances for each language for the same skill, and the speech processor will decide which language to use on the back end according to which language was selected on the Alexa device that took the request.

If you want to control your smart home with Alexa, you do not have to delve into the depths of programming to the extent demonstrated here. Instead, you can select the *Smart Home Skill API* as the *Skill Type* at the beginning of the flow (in the *Skill Information* tab) instead of selecting *Custom Interaction Model*. This gives you access at a higher command level, which is correspondingly easier to handle. *Flash Briefing Skill API* is another option for playing back previously bundled radio broadcasts.

If you prefer to use your own web server as the endpoint instead of a Lambda function on Amazon's Web services, you need to use SSL to do so, as well as deposit the certificate with Alexa. Of course, before any skill appears in the public list on Amazon's Alexa app store, Amazon wants to verify and certify it manually. ■■■

INFO

- [1] "Getting Started with the Alexa Skills Kit": <https://developer.amazon.com/public/solutions/alexa/alexa-skills-kit/getting-started-guide>
- [2] "Alexa Skills Store": <https://www.amazon.com/b?node=13727921011>
- [3] Amazon Developer Services: <https://developers.amazon.com>
- [4] Listings for this article: <ftp://ftp.linux-magazine.com/pub/listings/magazine/199/>
- [5] "Programming Snapshot – Amazon Web Services" by Mike Schilli, *Linux Magazine*, issue 196, March 2017, pp. 52, <http://www.linux-magazine.com/Issues/2017/196/Programming-Snapshot-Alexa-Web-Services>
- [6] Alexa Skills SDK on GitHub: <https://github.com/alexa/alexa-skills-kit-sdk-for-nodejs>

Celebrating 25 Years of Linux!

ORDER NOW!

Get 7 years of *Ubuntu User*

ON ONE DVD!



THE COMPLETE

UBUNTU
 **user**
ARCHIVE



Over
3,000
PAGES!
7 GREAT YEARS
OF UBUNTU
USER

Searchable DVD!

All Content Available in Both HTML and PDF Formats



Ubuntu User is the only
magazine for the
Ubuntu Linux Community!



Order Now! Shop.linuxnewmedia.com

The sys admin's daily grind: XMLStarlet

Health Spa

Charly and XML have never been best friends. However, because it was vital for him to have an excellent climate indoors, he plucked up his courage and considered XMLStarlet. *By Charly Kühnast*

A few smart home components from the HomeMatic system are permanent guests in my bathroom – they are for controlling the temperature in the room. A radiator thermostat and three sensors detect whether doors and windows are open. I use a central unit called CCU2 to set the climate I want and in which scenario. Their web interface seriously tests my patience. If, for example, I name a window contact Bathroom_Window_North instead of NEQ1651969 – because that's what it's called – I can't rely on the web interface to use the new name in all views (Figure 1).

The issue of security is another low point: The user interface is not password protected by default. I may be able to set one, but because CCU2 does not speak HTTPS, the password would fall into the clutches of the next sniffer anyway. Access to the script engine on port 8181 is also completely insecure. Nevertheless,

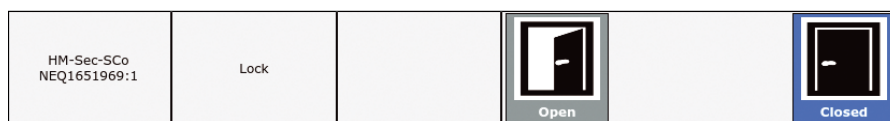


Figure 1: Window with a poor view: CCU2 cannot remember the name of window sensors.

the benefit is great because it allows me to use HTTP requests to read out the status of HomeMatic components and even set values.

Given this information, however, it should be clear that nobody is allowed to simply put the CCU2 onto the Internet via port forwarding, at least not anyone who doesn't want to invite all hackers to play poltergeist in their own home. Anyone wanting to view their home heating system from Jamaica will need a good VPN.

XML Instead of a GUI

So, I have a tool at hand for operating the CCU2 without its web interface. To receive the status of a window contact, I type the following into the console:

```
wget http://10.0.0.231/x.exe?Response=2
dom.GetObject(%27BidCos-2
RF.NEQ1651559:1.STATE%27).Value()
```

The CCU2 responds very quickly with (brrr!) XML:

```
<xml>
<exec>/x.exe</exec>
<sessionId/>
<httpUserAgent>
  User Agent: wget
</httpUserAgent>
```

```
<Response>>true</Response>
</xml>
```

I will suppress my XML aversion to help machines communicate with other machines – because that's just what XML is designed for. The true between the Response tags means that the window is closed. I can also retrieve the thermostat using similar commands and then set new temperature values. This means I can set up my own CCU2 GUI-free control using a few Bash scripts. Now I'm still missing an XML parser for the command line – with which I can finally angle for the XMLStarlet [1].

I previously had the output of the `wget` command above written to the `bathwindow.xml` file. In this way,

```
xmlstarlet sel -T -t -m xml 2
-v Response $WDIR/bathwindow.xml
```

I now get the value between the Response tags. Fancy a ride through the hell of parameters? Just read the 17-page (!) PDF document [2]. ■■■

INFO

- [1] XML Starlet: <http://xmlstar.sourceforge.net>
- [2] Documentation: <http://xmlstar.sourceforge.net/docs.php>

CHARLY KÜHNAST

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

Subscribe now!

Don't miss a single issue of the magazine that delivers the in-depth technical solutions you'll use everyday!



GET IT NOW!
SAVE TIME ON DELIVERY WITH OUR PDF EDITION



shop.linuxnewmedia.com/subs

WordGrinder and the escape from distraction

The Write Tool

WordGrinder offers distraction-free writing; we look at how realistic that concept is in everyday use. *By Bruce Byfield*

Would-be writers often believe that the fewer distractions software offers, the more efficiently they can write. This is the idea behind Focus-Writer [1] and Calligra Suite’s Author [2]. Now, WordGrinder [3] takes the concept even further, stripping down the word processor to the bare minimum, and running on the command line, primarily through keystrokes.

In my experience, this belief is a fallacy. Most professional writers are attached to a favorite application mainly out of habit, and the tool does not make the writer. Still, it remains the assumption behind WordGrinder. David Given, its writer, states on the project’s GitHub page [4] that “WordGrinder is not WYSIWYG. It is not point and click. It is not a desktop publisher. It is not a text editor. It [does] not do fonts and it barely does styles. What it does do is words. It’s designed for writing text. It gets out of your way and lets you type.” Click on the About page in WordGrinder’s menu, and you get nothing except a reference to “cat-vacuuming:” tasks that distract from actual writing.

Still, starting with this assumption, Given has produced an old-fashioned application. To use it efficiently, you need to make some preparations and be willing to learn a few dozen keyboard shortcuts, but the result is curiously refreshing, especially if you are working without a graphical interface.

Setting Up

WordGrinder opens on the fewest possible number of tools (Figure 1). In the

middle of the screen is a cursor between the indicators of the start and end of the file (terminators, as WordGrinder calls them). On the status bar at the bottom of the screen are indicators for the file name, its saved state on the left, and for word and page count on the right.

If you choose, you can begin typing immediately, using the arrow keys to navigate, and pressing the Esc key to summon the menu in the upper left corner of the screen (Figure 2). However,

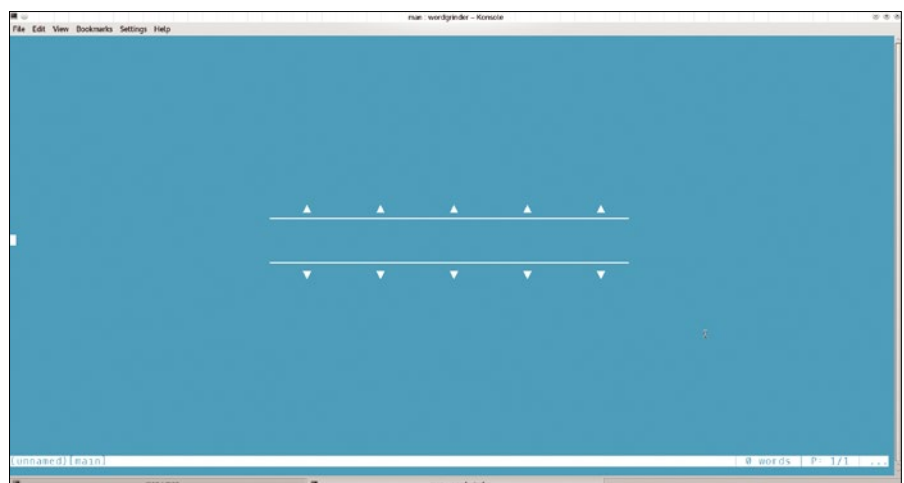


Figure 1: True to its philosophy, WordGrinder opens with a minimal interface.

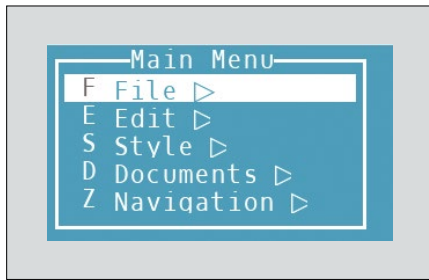


Figure 2: Press the Esc key to open the menu.

unless you are writing a completely unformatted document, the concentration that WordGrinder is intended to offer almost completely disappears. To take full advantage of the application, you need to memorize or note keyboard shortcuts you are likely to use, and perhaps make some adjustments to the default settings.

To start with, I recommend that you select *File | Global Settings* from the menu (Figure 3). The default of 80 characters per line makes for a long line that is hard to read, and I suggest a more standard 72-75 characters. Depending on your language preferences, you might

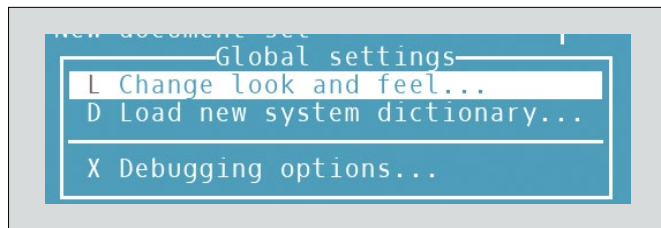


Figure 3: You may need to adjust some global settings for best performance.

also want to load an ispell dictionary for your locale.

Next, open *File | Document Settings* (Figure 4). You should change the page count from the default count of 250 words, which is short even for letter sized paper – let alone legal or A4 paper – to at least 325 words to give yourself a more accurate page count. You will also probably want to enable Autosave as a precaution and enable smart quotes (rounded quotes) to save yourself the effort of changing them before submitting a manuscript. Depending on your needs and preferences, you might also want to enable the Scrapbook, a buffer for storing snippets of text, along with HTML export, and Spellchecker.

If you really feel the need to minimize distractions, turn off the display of terminators from *File | Document Settings* and the display of

the status bar from *Style | Toggle Status Bar*. The result will be a screen that is blank except for your input and occasionally the menu.

Other modifications can only be supported for the display, and not the actual document. For example, you cannot select document fonts, but you can display fonts or foreground and background colors for the display in a profile in which WordGrinder is running.

In theory, further modifications can be done using the Lua scripting language. You can call a script by starting WordGrinder with the `--lua FILENAME` extension. You are also supposed to be able to set your personal configuration file

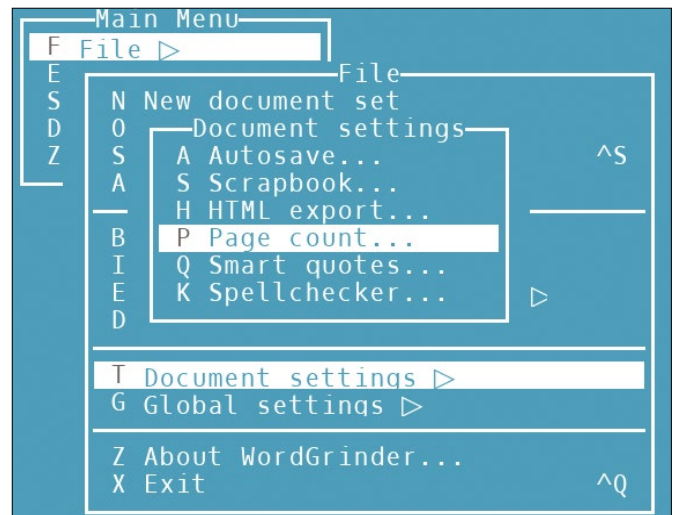
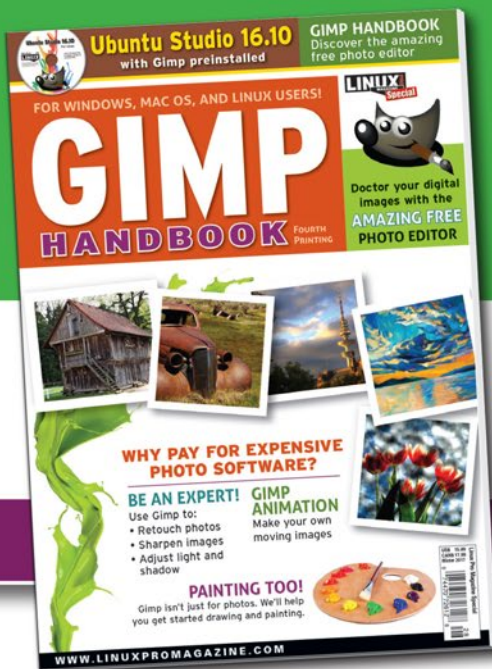


Figure 4: Document settings give you additional tools with which to work.

Shop the Shop

shop.linuxnewmedia.com



GIMP HANDBOOK

- Fix your digital photos
- Create animations
- Build posters, signs, and logos



Order online: shop.linuxnewmedia.com/specials

FOR WINDOWS, MAC OS, AND LINUX USERS!



```

Navigation
Cursor up UP
Cursor right RIGHT
Cursor down DOWN
Cursor left LEFT
Selection up SUP
Selection right SRIGHT
Selection down SDOWN
Selection left SLEFT
Select word ^W
Goto previous word ^LEFT
Goto next word ^RIGHT
Goto next paragraph ^DOWN
Goto previous paragraph ^UP
Select to previous word S^LEFT
Select to next word S^RIGHT
Select to next paragraph S^DOWN
Select to previous paragraph S^UP
Goto beginning of line HOME
Goto end of line END
Select to beginning of line SHOME
Select to end of line SEND
Goto beginning of document ^PGUP
Goto end of document ^PGDN
Select to beginning of document S^PGUP
Select to end of document S^PGDN
Page up PGUP
Page down PGDN
Selection page up SPGUP
Selection page down SPGDN
Delete previous character BACKSPACE
Delete next character DELETE
Delete word ^E
Toggle mark ^@

```

Figure 5: WordGrinder is most efficient when you know its keyboard shortcuts.

using Lua. So far, however, no examples seem to be posted, so modifications with Lua will need to wait for another day.

Making all these changes each time you start a file would detract from WordGrinder's purpose, so you should make these changes in a new file and save it as a template. When you start a new document, you can then open the template and save it with a new name, to save yourself repeated work.

Working with WordGrinder

WordGrinder supports 33 navigational shortcuts (Figure 5). Most are obvious, such as the Page Up, Page Down, Delete, and arrow keys. Others add the Ctrl key, which is indicated by ^ in the menu. Other options are to use the Find Next and Go To tools in the Edit menu. You can switch between open files from the Documents menu. However, no tools are available for comparing documents, let alone for advanced functions like mail merge.

One advantage that WordGrinder has over text editors is that copy and paste functions, as well as Undo and Redo, use the same keyboard shortcuts as the desktop. For example, instead of the usual Ctrl + Shift + V in a virtual terminal, past-

ing requires only Ctrl + V. Although Ctrl + Shift + V will work in WordGrinder, if you are like me, you may welcome not having to press the extra key. Character and paragraph styles are supported – but “barely,” as Given notes (Figure 6). So-called character styles are limited to italic, bold, and underline. Similarly, paragraph styles are limited to a few items, such as plain text, three levels of headings, list items, and indented paragraphs, none of which can be edited. Basically, the selection is limited to formatting supported by HTML without any CSS files. Although WordGrinder can display styles within the manuscript, you may want to select *Style | Margin Mode* to have styles listed in the margin, as well as the number of words in each paragraph.

Saving a file in WordGrinder can be confusing. To save a file with a native .wg extension, you must first designate it as a Document Set in the File menu, so that it includes the Scrapbook, and then save the document set. Exporting to HTML, Open Document Format, LaTeX, or troff is more straightforward, but all saves require a full path name, a requirement that can momentarily trip you up.

The Price for Reducing Distractions

WordGrinder provides fewest distractions when you enter unformatted text. Despite Given's claims, formatting from

```

Style
I Set italic ^I
U Set underline ^U
B Set bold ^B
O Set overline ^O
P Plain text
M H1: Heading #1
S H2: Heading #2
H3: Heading #3
H4: Heading #4
Q Indented text
LB List item with bullet
L List item without bullet
V Indented text, run together
PRE Preformatted text
RAW Raw data exported to output file

```

Figure 6: WordGrinder offers minimal support for character and paragraph styles.

the menu or pausing to add keyboard shortcuts feels to me as much of a distraction as formatting in any other text editor or word processor. The amount of formatting can be reduced by using a template, yet the distraction remains.

To make matters worse, WordGrinder is at best the work for a first draft. Before actually submitting a manuscript, you will still need to format headers and/or footers so that you can add the title, name, and page number to every page. Margins and possibly even a standard font must also be formatted. In other words, using WordGrinder means basically preparing a manuscript twice, even when exporting to Open Document format or LaTeX. Unless formatting seriously distracts you, you are probably better off using a regular word processor or text editor, especially with a template to take care of all the formatting.

WordGrinder may have a use in embedded systems, where space is limited. But, as intriguing as WordGrinder may be – particularly in 2017 – the quest for distraction-free writing seems quixotic. Inescapably, some formatting is required, and, in the end, some distraction is inescapable. ■■■

WordGrinder may have a use in embedded systems, where space is limited. But, as intriguing as WordGrinder may be – particularly in 2017 – the quest for distraction-free writing seems quixotic. Inescapably, some formatting is required, and, in the end, some distraction is inescapable. ■■■

WordGrinder may have a use in embedded systems, where space is limited. But, as intriguing as WordGrinder may be – particularly in 2017 – the quest for distraction-free writing seems quixotic. Inescapably, some formatting is required, and, in the end, some distraction is inescapable. ■■■

INFO

- [1] FocusWriter: <https://gottcode.org/focuswriter/>
- [2] Calligra Suite Author: <https://www.calligra.org/news/calligra-2-6-alpha-released/attachment/calligra-author/>
- [3] WordGrinder: <http://cowlark.com/wordgrinder/>
- [4] WordGrinder GitHub: <https://github.com/davidgiven/wordgrinder/blob/master/README.wg>

MORE UBUNTU!



Can't get enough Ubuntu? We've got a whole lot more!

Ubuntu User is your roadmap to the Ubuntu community. In the pages of **Ubuntu User**, you'll learn about the latest tools, best tricks, and newest developments in the Ubuntu story.

Ubuntu User helps you explore the treasures of open source software within Ubuntu's expansive repositories. We'll bring you exclusive interviews with Ubuntu leaders, keep you current on the exciting Ubuntu community, and answer your most perplexing Ubuntu questions. Learn how to choose a video editor, find the perfect tool to customize your desktop, and configure and manage Ubuntu systems using the best admin tools.

DON'T MISS ANOTHER ISSUE!



HUGE SAVINGS OFF THE NEWSSTAND PRICE!

SUBSCRIBE NOW: SHOP.LINUXNEWMEDIA.COM



Intro to the Gnome Flashback desktop

SIGNS OF LIFE

If you struggle with the appearance and behavior of the Gnome desktop, the classic features of fallback mode offer an alternative in a familiar style. Lamented by many as dead and gone, Gnome 3 fallback is still alive and kicking in Gnome Flashback. *By Mario Blättermann*

In spring 2011, Gnome 3.0 launched with a new design approach that threw many well-known features overboard and caused bitter opposition. As a concession to long-time fans of the classic desktop, the developers ported its basic components to GTK 3, provided them as an alternative to Gnome Shell, and called it “fallback mode.”

Just two years later, in Gnome 3.8, the project replaced fallback mode with “Classic Mode,” which comes as a customized Gnome Shell requiring 3D-capable hardware. The modular structure of session management, the notification system, the window manager, and an application level were thus officially laid to rest.

Little impressed by the regular Gnome and its new workflow, a small team led by Latvian developer Alberts Muktupavels soon convened to keep the previous Gnome alive as Gnome Flashback [1]. However, neither users nor developers

took notice of it, so the legend was substantially restricted to bug fixes.

PARALLEL WORLDS

Besides Gnome Flashback, a number of other projects are still flying the flag of the classic desktop. Most closely associated with Gnome 2 is MATE, which first appeared in August 2011. Back then it was only a version of Gnome 2, to which the developers made relatively minor changes with respect to the names of applications and the installation paths, which allowed users to install it parallel to Gnome.

Although Gnome Flashback is limited to a very compact core that looks to replace Gnome Shell, the intrepid MATE developers simply grabbed the complete Gnome 2.32 release. At the time, nothing had been ported to GTK 3, and the situation today does not look much better. The move to the latest version of the GTK libraries is still underway and might not even be completed by the time support for GTK 2 ends. Work on applications is very low key, although the developers repeatedly borrow

Nevertheless, Gnome Flashback is still worth a look, if only because the popular

from new Gnome software and have adapted many things to the latest technologies under the hood.

In contrast, Flashback opted for GTK 3 from the outset. However, in the absence of peripheral software, it was always necessary to coordinate interaction with the latest versions of Nautilus and the rest of Gnome Core. The developers had a huge task, which explains why innovation is limited.

Ultimately Xfce, LXDE, LXQt, and even KDE follow the classical approach, so neither MATE nor Flashback is without alternatives. Only Gnome Shell and Cinnamon can be viewed as fully integrated, because they do not separate window management and the work environment. If you are primarily interested in replacing the window manager, almost all of the distributions mentioned here are candidates, apart from Gnome Shell and Cinnamon.

Lead image © Barmaliejus, fotolia.com

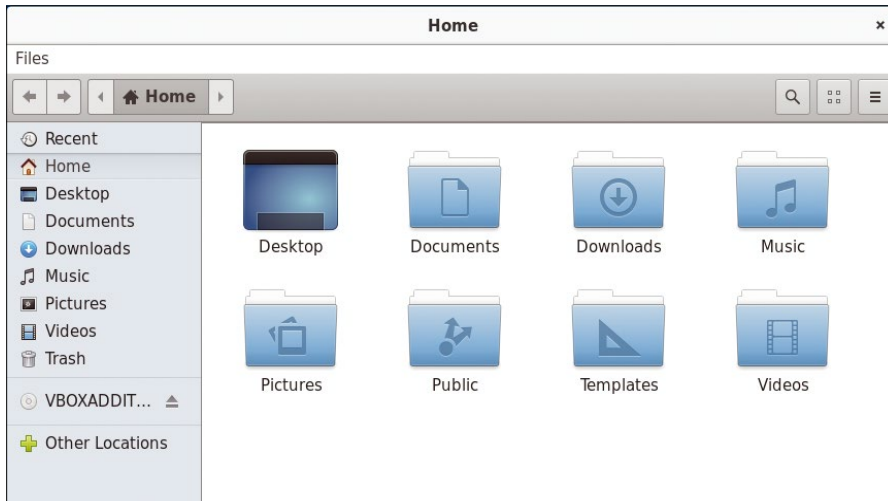


Figure 1: In our lab, I deployed Gnome Flashback on Xubuntu 16.10, among other distributions.

AVAILABILITY

The current packages of Gnome Flashback are available for Debian and Ubuntu and their derivatives. You just need to install the *gnome-session-flashback* package; everything you need is automatically stored on disk. With the exception of the Nautilus file manager, this only gives you the session and does not install any additional Gnome components. You even need to choose to install the *gnome-applets* package explicitly, to avoid just having the basic panel extensions on board.

Users of Arch Linux will find the software in the AUR community repository. For RPM-based systems, however, things look slightly less clear-cut. The less widespread ALT Linux is up to date; however, openSUSE is unable to deliver anything suitable. Sporadically maintained packages for Fedora are found in a Copr repository [3], but a strange error occurred in our lab: Although the external source offered a sufficiently recent Metacity, a completely outdated version (3.12) ended up on the system. This version found it difficult to interact with the rest of Flashback; for example, it did not let me configure the window theme. Metacity 3.20 only arrived with the next update.

Installing from the source code turns out not to be too complicated. The source code packages needed for this can be found on the Gnome project's FTP server [4]. To begin, you install Metacity, the panel and the notification daemon, and then the Flashback module itself. The panel applets are optional. You will find notes on the installation with the typical three-step process (`configure`, `make`, `make install`) in the archive with the source code. The proprietary JHBuild Gnome tool [5] significantly simplifies the installation of the latest source code from the Gnome Git repositories in particular.

MATE project [2] takes a similar approach, but from a different starting position (see the "Parallel Worlds" box). If you struggle with the appearance and behavior of the Gnome desktop, the classic features of fallback mode offer an alternative in a familiar style.

Test Run

To test Gnome Flashback, I used Fedora 25 and Xubuntu 16.10 (see the box titled "Availability"). After installation, logging out and logging back in again, just select the *GNOME Flashback* from

the list of display managers. After a minute of silence while the system finds the settings for the first setup, the classic desktop with panels at the top and bottom, as well as some preconfigured applets (Figure 1), appears.

Flashback is still encumbered with some of the shortcomings of its ancestor Gnome 3.0: Direct access to the System menu disappeared from the original three-part menu at the top and never returned. Something similar happened to the tooltips in the Application menu and the free placement of (the meagre number of) applets in the panel. Most of the applets from Gnome 2 are available, but that's all.

Adding applets is a somewhat cumbersome process – a simple right-click on the bar would be too easy. You need to hold down the Alt key at least to open the Context menu; on many systems, you have to hold down the Meta (Windows) key, too. The applets (Figure 2) are exclusively old friends from Gnome 2.

As is typically the case in Gnome, the window title bars in Flashback only have a single button for closing the window, but you do not need to leave things that way: In the GSettings configuration database, you can adjust

this setting. With just a little typing in the graphical dconf Editor you can put the missing buttons back (Figure 3).

Since version 3.20, GSettings also lets you adapt the window theme under *org.gnome.metacity.theme* in the dconf Editor. The default value is *gtk*, which means that Metacity takes the value from the GTK theme.

Alternatively, you can enter the name of a theme stored on your system under */usr/*

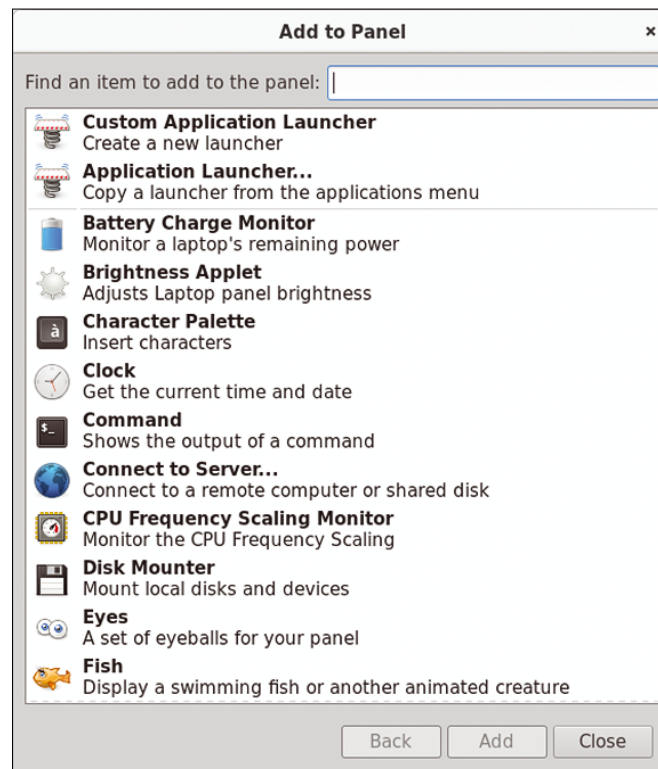


Figure 2: The panel applets hold no surprises, being old friends from Gnome 2.

share/themes, ~/.themes, or ~/.local/share/themes. You then just need to reset the slider next to *Use defaults* to apply the setting.

Window Changer

Unlike Gnome Shell or Cinnamon, which hardwire the window manager to the desktop, the window manager can be relatively easily replaced in Flashback. Flashback comes with an entry for Compiz, which was already very popular in the days of Gnome 2, in the login manager menu. To choose Compiz at login, simply select *GNOME Flashback (Compiz)*.

You can try out the Openbox window manager using the `openbox --replace` command. This minimalist manager is very friendly on system resources. If you like, you can even add a command in the startup folder. Just create a text file and save the lines

```
[Desktop Entry]
Exec=openbox --replace
Name=Openbox
NoDisplay=true
Type=Application
```

in ~/.config/Autostart. From now on, Metacity starts first when you next log in and is then quickly replaced by Openbox.

Openbox did leave a slightly bitter aftertaste, however: Pressing the Alt + F2 keyboard shortcut did not work out of the box, because earlier versions came with a binary that launched a quick-start window without the need to define it on the command line. On Debian- and Ubuntu-based systems, you will still find this practical tool in the *gnome-panel-control* package, but the developers officially removed Gnome support as of Openbox 3.6.

In this case, you can fall back on alternative tools such as Synapse [6], which you should set up to launch when you press Alt + F2. The corresponding key binding is available in the <keyboard> section of the ~/.config/openbox/rc.xml file. In contrast, the Compiz Settings Manager still knows what to do with Alt + F2 and requires no modifications: The quick-start window appears in the usual way.

Fresh Fruit

Like Gnome 3, the global application menu from Ubuntu's Unity at the top of the screen splits users into two camps. The menubar, which is heavily reminiscent of Mac OS, pervades all desktops, even Flashback.

Thanks to support for the indicator applets, a Flashback version is available for Ubuntu. Simply install the *indicator-applet-appmenu* package.

To make space in the top panel, you could remove the two-part Gnome menu and replace it with the simple alternative of just one button. After context-clicking on the panel and selecting *Indicator Applet Appmenu*, you can now add the menubar to the panel. The results are impressive, as Figure 4 shows.

Although the application name in front of the menu, as known from Mac OS applications, is missing, interaction with the program windows does work initially – as long as you ignore the fact that new Gnome application menus are fairly monosyllabic and that you generally need to use a mouse to control them. This solution is very user friendly, because the menu of the application currently in focus always remains fixed at a central location.

On the downside, access by keyboard revealed a fairly serious bug: Pressing Alt + D failed to expand the File menu from the panel and instead popped up the (nonexistent) line in the program window. This behavior will be especially irritating among power users, who often prefer to work with the keyboard. The strange program behavior unfortunately is not an isolated case; a similar implementation in Cinnamon also exhibits the same symptoms. Whether you accept the menubar in this form or not is an entirely different question.

The second most important feature of the Apple desktop is undoubtedly the dock at the bottom. You can conjure up something pretty similar to the desktop using Plank dock, which might not be bursting with features but perfectly fulfills the objective of creating a launch bar with a built-in window list (Figure 5, bottom).

Both visually and in terms of the control concept, Plank is oriented on the role model of the Mac software. You can perpetuate a program called via the Start menu or the command line in the Dock by right-clicking on the icon and selecting the *Keep in Dock* option. As you mouse over the icons, the software enlarges the icons and a tooltip displays the name. This means that you can keep the icons pleasantly small using

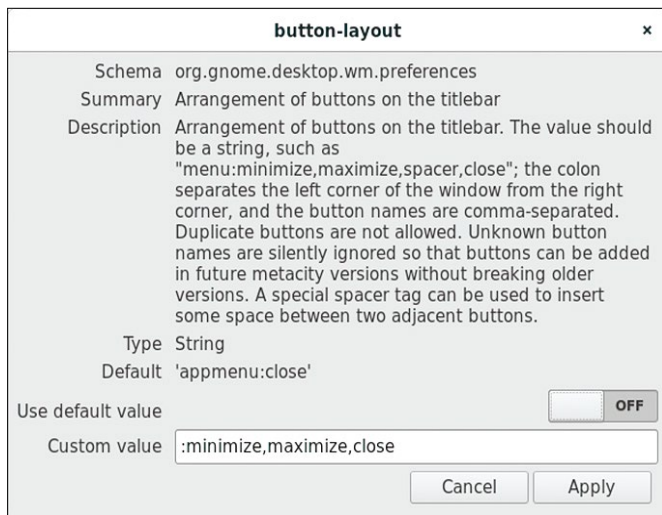


Figure 3: The dconf Editor helps you to make Flashback a little more compatible with everyday life.

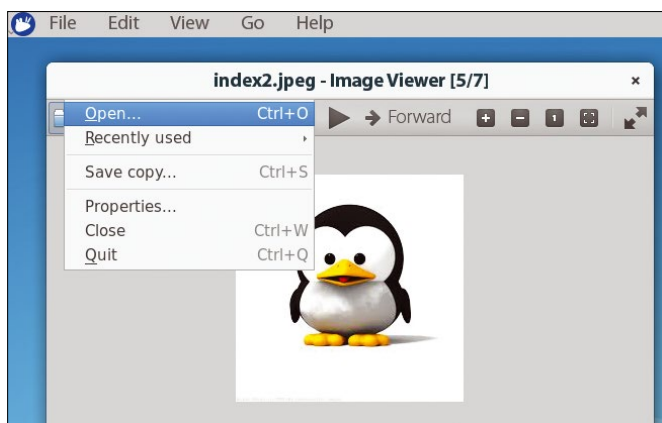


Figure 4: A touch of Mac: The developers attempted to remove an element from the desktop, the global menubar, that was hugely popular with many users.



Figure 5: The Plank dock adds an application dock and launch bar at the bottom of the desktop.

the default settings, because Plank is amazingly fast – even on high-resolution wide screens.

Moreover, Plank fades from the desktop when a window near it is opened. You can restore Plank again by moving the mouse to the bottom of the screen. Unfortunately, enlarging icons did not work on the second test system with Fedora 25: The developers removed the wave-shaped animation to avoid a possible patent dispute.

Conclusions

Gnome Flashback must be understood as a replacement for Gnome Shell. Other features, such as window frames drawn by the application or the isolation of settings in the official Gnome Control Center on the one hand and the Gnome Tweak Tool on the other, are clearly part of the Gnome concept, whether Flashback or Shell. I did not note any particularly good integration, nor incompatibilities, of core applications (e.g., Nautilus)

with the classic desktop in this test. If you accept the fairly wide gap between expectations and reality that shows up in some places, then Flashback has already found its place in the desktop environment landscape.

Flashback only gives you a small core desktop of session management, panels, and a few other components. If your objective is the kind of modularity that makes it possible to exchange core components, then Flashback is a good choice, but not the best. If development continues to be limited to cosmetic improvements, interesting alternatives come to mind: Desktops like Xfce, LXQt, or MATE are superior to Flashback in every way when it comes to modularity, and they offer a wider choice of specific extensions. ■■■

INFO

- [1] Gnome Flashback: <https://wiki.gnome.org/Projects/GnomeFlashback>
- [2] MATE desktop: [https://en.wikipedia.org/wiki/MATE_\(software\)](https://en.wikipedia.org/wiki/MATE_(software))
- [3] Gnome Flashback (Fedora): <https://copr.fedorainfracloud.org/coprs/yselkowitz/gnome-flashback/>
- [4] Gnome Flashback (source code): <https://wiki.gnome.org/Projects/GnomeFlashback#Releases>
- [5] JHBuild: <https://wiki.gnome.org/action/show/Projects/Jhbuild>
- [6] Synapse project: <https://launchpad.net/synapse-project>

LINUX UPDATE

Need more Linux? Our free Linux Update newsletter delivers insightful articles and tech tips to your mailbox twice a month.

- 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



The Core Infrastructure Initiative revisited

Core Values

How does the Core Infrastructure Initiative fare three years in? *By Bruce Byfield*

The Linux Foundation started the Core Infrastructure Initiative (CII) [1] after the discovery of several security vulnerabilities in 2014. As serious as the bugs themselves was the

discovery that many core Linux projects were unequipped to respond to them. The CII was started in an effort to alleviate and prevent such bugs in the future. But how is the CII doing three years after its founding?

For answers, I talked with Nicko van Someren of the CII.

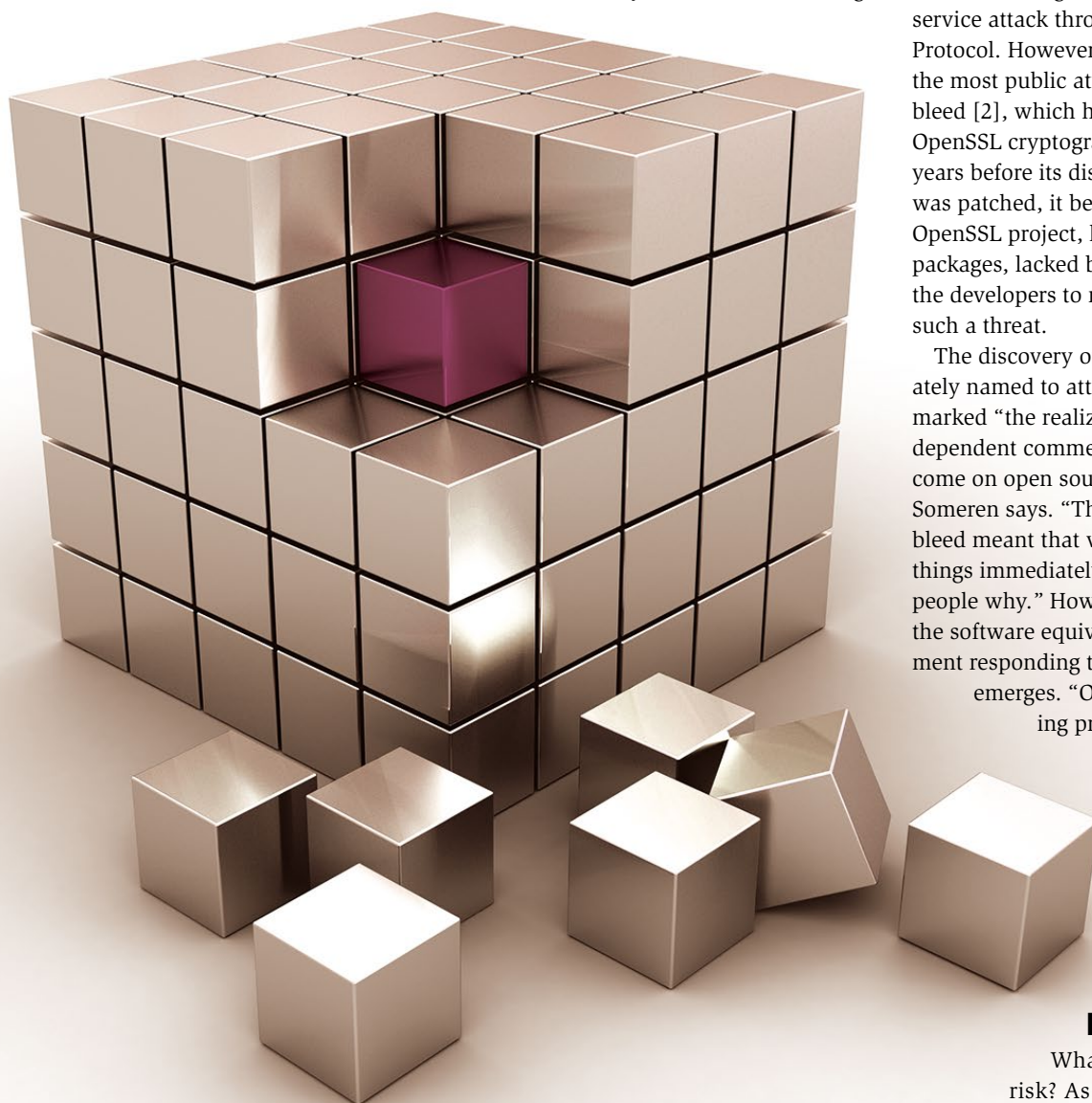
According to van Someren, 2014 was marked by the discovery of a number of vulnerabilities. They included the ShellShock bug in Bash and a denial of service attack through the Network Time Protocol. However, the bug that received the most public attention was Heartbleed [2], which had infected the OpenSSL cryptography library for two years before its discovery. As Heartbleed was patched, it became obvious that the OpenSSL project, like many other Linux packages, lacked both the funding and the developers to respond adequately to such a threat.

The discovery of Heartbleed – deliberately named to attract attention – marked “the realization as to just how dependent commercial software has become on open source components,” van Someren says. “The urgency of Heartbleed meant that vendors needed to fix things immediately and needed to tell people why.” However, the CII is not just the software equivalent of a fire department responding to each crisis as it emerges. “Our goal with identifying projects that are at risk

is so that we can provide help and resources – before – there is an urgent problem,” von Someren says.

Identifying Projects at Risk

What puts projects at risk? As von Someren notes,



the answer depends on the definition of risk. “There were a handful that needed urgent attention when we started the CII, and we have already worked with most of these,” he says. However, “there are many more, dozens or hundreds of projects that are widely relied upon, but which don’t have an active community to support them and which would struggle to respond in a timely manner if a vulnerability was ever found.”

With limited resources, the CII must triage. Its first priority is projects “where providing resources will substantially improve the security of the Internet and of the businesses and society that depend on it.” Another priority is underfunded projects that are not receiving funds from other sources. In both of these priorities, as I heard at a CII event at the 2015 LinuxCon, the CII is well positioned to help, because the Linux Foundation is often perceived as relatively neutral, and developers who would refuse direct funding from business interests may accept funding from the CII.

These priorities mean likely candidates for CII assistance include:

- Tools and libraries critical to the functioning of the Internet, such as NTPD or Bash, that are not security tools themselves. Because such packages are widely used, vulnerabilities in them are likely to have a large effect.
- Security tools and libraries, such as OpenSSL and GnuPG.
- The development of security testing tools such as the OWASP Zap project, which assists in discovering security issues during development.
- The development of tools that educate consumers and developers about security and encourage better practices and processes (see below).

“We try to look at all investment from a cost-benefit point of view,” von Someren says. “While we do provide funds for maintenance activities, we would rather pay for work that has long-term value to the community rather than preserving the status quo.”

Projects looking for CII support must explain how funding them will improve security in free software and describe the milestones they expect to reach. The CII and its Advisory Board members vote on proposals, sometimes

setting conditions on support, such as requiring a Git repository. Some funding is by definition for limited duration, but, if necessary, funding can be renewed so long as “there is still work to be done, the team is still making progress, and the project is still a priority.” Projects that fail to deliver are not renewed, and at times, shifting priorities means that “we will divert resources to where we think that they are needed the most.”

Preventative Programs

As important as triage can be, CII is also attempting to prevent security problems before they emerge – a practice that should be more cost effective both financially and in terms of effort. With this goal in mind, it has used its Census [3] program to try and identify at-risk projects more reliably. The first Census collected data on package usage, bug density, developer response time, and community activity, resulting in a table of more than 170 potentially at-risk projects – many of which are tools that users depend upon on a daily basis. In 2017, says von Someren, “we are hoping to revamp this [program] to be a more continuous process, drawing data from more places, and looking at a larger set of packages.”

CII also runs the Best Practices Badge Program [4], whose goal is to encourage projects to operate more securely on a daily basis. “We have found that many projects benefit from having a checklist of steps that they can take to make their process more secure.” Many of these steps seem self-evident, but von Someren observes that even well-run projects “often discover things that they should have been doing all along.”

Getting Results

CII has not always succeeded in its efforts. In particular, the early decision to fund both the Network Time Protocol project and its hostile fork NTPsec seems to have done little to solve chronic underfunding and staffing, leaving instead two rival projects [5] where only one existed before. Perhaps as a result of this chaos, the funding for neither project was renewed for 2017.

By contrast, von Someren reports that team members of OpenSSL, which was

instrumental in the founding of CII, “now feel that they have moved from a firefighting mode to being able to work on developing new functionality” as a result of CII funding. Similarly the Best Practices Badge Program has had more than 600 projects signed up, with more than 60 meeting the standards required to obtain a badge.

“I think that the CII is making a valuable impact,” von Someren says, “but there is always more to do. We have moved some projects off of the endangered list, and I think that, like some of our projects, we have moved from firefighting to a more forward-looking approach. Right now, we are resource constrained – we have more work that needs doing than money to do it, but I think that where we have been able to apply resources, we have enabled a bunch of open source projects to make themselves more secure.”

Looking into the future, von Someren sees more of the same: more pinpointing of at-risk projects, more education of consumers, and more encouragement of developers to make security tools easier to use. “Many people like to say that security is not a feature, it’s a process,” he says. However, “I think that in open source software, security is a culture as much as it is a process” – and culture can be much harder to change than processes. Evaluating the last three years, the wonder is not that the CII has sometimes been restrained by its budget and the vagaries of projects, but that it has managed to become an effective presence in free software at all. ■■■

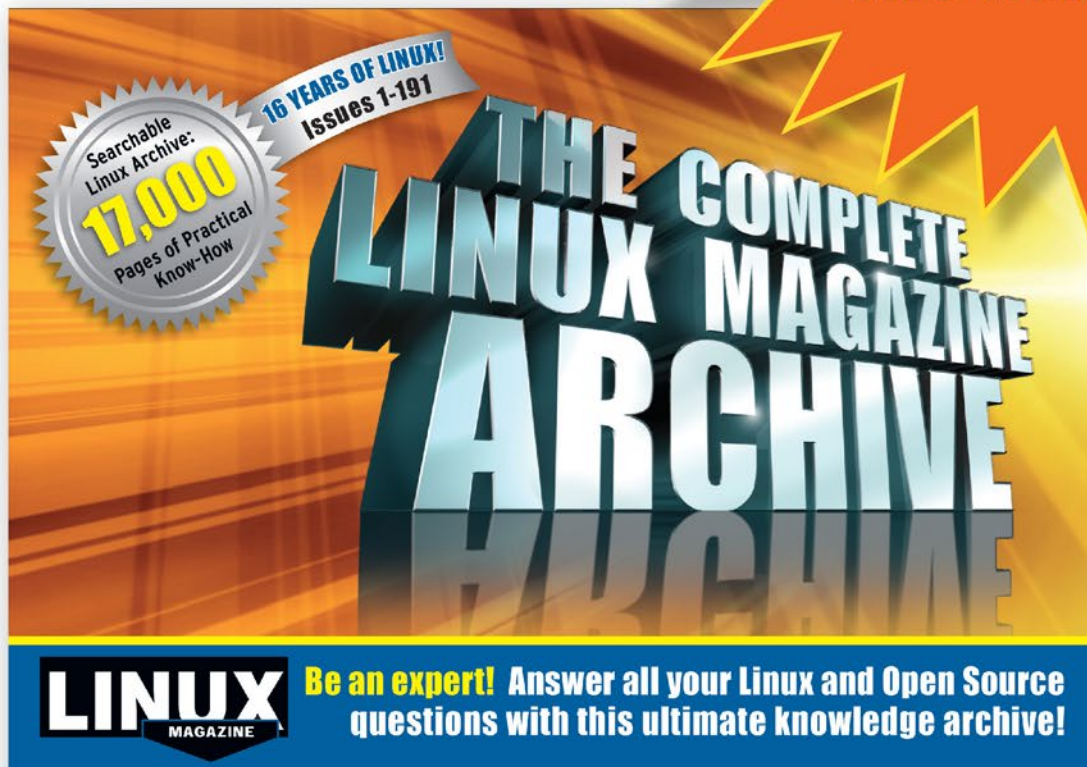
INFO

- [1] CII: <https://www.coreinfrastructure.org/>
- [2] Heartbleed: <https://en.wikipedia.org/wiki/Heartbleed>
- [3] Census: <https://www.coreinfrastructure.org/programs/census-project>
- [4] Best Practices Badge Program: <https://www.coreinfrastructure.org/programs/badge-program>
- [5] “Paving with Good Intentions: The Attempt to Rescue the Network Time Protocol” by Bruce Byfield. *The New Stack*, March 13, 2017: <https://thenewstack.io/paving-good-intentions-attempt-rescue-network-time-protocol/>

Happy 25th Anniversary to Linux!

Help us celebrate
25 years of Linux
with the All-Time
Archive DVD of
Linux Magazine!

Order today and
get 191 issues of
Linux Magazine on
one handy DVD!



You get 17,000 pages of practical know-how on one searchable disc - that's 191 issues including 40+ issues that were not included in the previous release.

Order Now! Shop.linuxnewmedia.com



Ben Everard

This month I've been working on processing large volumes of data. I won't bore you with the details, but it involved using a tool that brought some data together, then pushed it up into a commercial cloud infrastructure so we could process it on lots of machines at the same time. The tools have proved to be a bit flaky, but with one important difference: The tool I was using to pre-process the data and get it into the cloud infrastructure was open source, whereas the tools in the cloud infrastructure were closed (at least the bits we had trouble with were).

There are lots of reasons to use open source software, but a huge one for me is that I can fix stuff. When the open source tool broke, I would look at the code. Sometimes I could fix the problems, sometimes I could work around them, but at least there was something I could do. When the closed source software broke, I was stuck waiting for a customer service agent to get to my email and see if they could do something.

Now, either of these things is a bit of a lottery. It's possible that the bug in the open source program is in some byzantine code that I can't understand let alone fix, and it's possible to get a quick fix from customer service agents. However, my small amount of anecdotal evidence is starting to convince me that neither of these are very often the case. I've frequently been able to fix little issues with FOSS software, and battled endlessly with customer service people. I thought about this when reading Mike's piece on FOSS advocacy this month. For me, advocacy always starts with understanding why something's important to you, and using that passion to explain your point of view to someone else.

It's not all about other people this month. You can find out what Andrew discovered when he ventured out of Yorkshire to a little estate called Bletchley Park, discover the latest software with Graham's FOSSPicks, write your own graphical programs with Mike's guide to Python, and much more. Spring is here, so turn the page and dive into a pool of Linux learning.

– Ben Everard



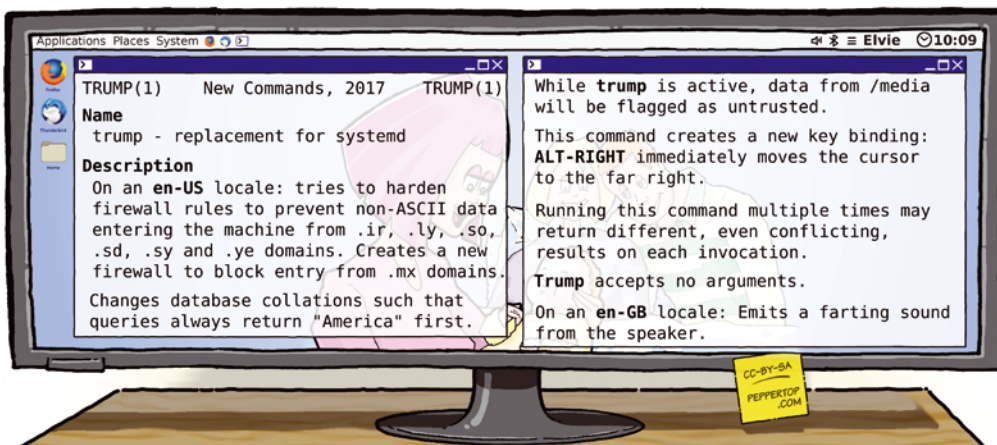
Andrew Gregory



Graham Morrison



Mike Saunders



LINUX VOICE

News	66
<i>Simon Phipps</i>	
OSI approval guarantees the freedom to innovate.	
Computing's Golden Age	67
<i>Andrew Gregory</i>	
Everything is awesome – the past is a foreign country.	
Doghouse – Legacies	68
<i>Jon "maddog" Hall</i>	
How our actions affect others in ways we cannot predict.	
FOSS Advocacy	70
<i>Mike Saunders</i>	
Learn the tricks, tips, and techniques for converting friends, family, and colleagues to free and open source software.	
FAQ – TensorFlow	74
<i>Ben Everard</i>	
Welcome our new artificial intelligence overlords by tinkering with their gray matter.	
Core Tech – Network Sniffers	76
<i>Valentine Sinitsyn</i>	
Learn what's going on in your network using Linux and its arsenal of packet capture tools.	
FOSSPicks	82
<i>Graham Morrison</i>	
Sonic Visualiser 3, Latte Dock 0.5.91, Tizonia 0.7.0, HFS+ Rescue 3.3, Project: Starfighter 1.7, and more!	
Tutorial – Server Security	88
<i>Ben Everard</i>	
Fear not the barbarians of cyberspace and follow our guide to shoring up your digital defenses.	
Tutorial – TkInter	92
<i>Mike Saunders</i>	
Expand your Python knowledge and write GUI apps with a smattering of code, thanks to the TkInter toolkit.	

NEWS ANALYSIS

The Linux Voice view on what's going on in the world of Free Software.

Opinion

Why OSI License Approval Matters

OSI approval guarantees the freedom to innovate. **BY SIMON PHIPPS**



Simon Phipps is a board member of the Open Source Initiative, the Open Rights Group, and The Document Foundation (makers of LibreOffice).

Does it really matter if a copyright license is Open Source Initiative (OSI) approved? Surely if the license looks like it meets the benchmark that's all that matters? I think that's the wrong answer; OSI license approval is the crucial innovation that's driven the open source revolution.

"Open Source" describes a subset of free software that is made available under a copyright license approved by the OSI as conforming with the Open Source Definition (OSD). Having a standards body for licenses – one which ratifies the consensus of an open community of license reviewers – saves individuals from needing to each seek out a legal advisor to tell them whether a given license does in fact give them the rights they need to build or deploy the software they want. By providing easy certainty, open source gives people permission in advance to meet their own needs and innovate with technology.

The only OSD compliance arbiter is the license review process conducted collaboratively by the open source community and summarized and ratified by the OSI Board of Directors. Others have no role outside this process and are not entitled to assert that a non-approved license satisfies the OSD. As such, licenses that

have not received OSI approval don't satisfy the process and can't be considered open source.

The strength of OSI's approach is that it is objective; a license is either on the approved list, or it is not. Licenses on the list are known to give permission in advance and unlock software freedom; those not on the list cannot be guaranteed to do either. The Free Software Foundation (FSF) uses a subjective approach that encourages speculation about whether a license is "free." Meanwhile, there are many with vested interests in diluting free and open source software who want a subjective approach where every individual may act as their own arbiter. Despite these pressures, it's the OSI's approach that has made open source succeed.

That's not because a legalistic tick-in-the-box is really interesting. Rather, it's because developers can gain certainty as to whether they can use a project simply by checking its approval status. No one has to be asked for permission or clarification. Significantly, there's no need to retain a lawyer just to check that the license is in fact safe to use.

It's easy to get overwhelmed by all of the details of the many open source li-

censes, losing sight of the reason they are important. They're important because every open source license guarantees the freedom to innovate without seeking permission first. OSI approval means you have the unconditional right to use the software in question for any purpose (sometimes calls "freedom zero"). You also have an unconditional right to make new software based on that software for your own use, and a conditional right to share the software – modified or not – with other people. The final case comes with some complexities beyond the scope of this article, especially for copyleft licenses.

That freedom to innovate, unlocked by the permissions the OSI-approved license guarantees in advance, is the powerhouse of open source. Developers know they can incorporate open source components without seeking legal advice. Users know they can deploy the software with the confidence that they have a license and won't be persecuted by rent-seeking proprietary software companies. Together, this liberty has realized the potential of free software and propelled open source to dominance over the course of the last decade. ■■■

...Together, this liberty has realized the potential of free software and propelled open source to dominance over the course of the last decade.

Everything Is Awesome

The past is a foreign country **BY ANDREW GREGORY**

After years of promising myself that I would go there, last week I went to Bletchley Park, home of the code breakers, a sculpture of Alan Turing, and an excellent day out.

Next door to the Bletchley Park museum (but occupying the same wartime site) is The National Museum of Computing. The two museum trusts have had a falling out that my tiny mind can't understand, but the end result is very sad: They are separated by a wire fence, have separate admission charges, and there's an obvious animosity towards Bletchley from some of The National Museum of Computing volunteers.

Apart from my having to walk a bit further and pay a bit more, the effect of this is that the bombes (the mechanical calculators that cracked the German military's Enigma cipher) and Colossus (the first computer, which was built to crack the Lorenz cipher used by the Nazi high

command) are kept separate, even though they form part of the same story. Humans are silly.

Human silliness aside, the other thing that stuck with me from the computing museum is how lucky we are. Nostalgia is pernicious; it's easy, and lazy, to say that things were better in the olden days. But in the field of computing, we've never had it so good.

This was brought home to me by the ICL 2966 mainframe. This machine is enormous – the size of four or five fridge freezers. It only runs when the man who knows how it works is there, and it powers a few dumb terminals running Conway's Game of Life and a version of noughts and crosses. The electricity and maintenance costs must be huge, not to mention the cost of buying the thing itself.

On the days when the man who knows how the ICL 2966 works isn't

there, they switch it off and use a Raspberry Pi instead.

Yes, a Raspberry Pi. For just a few quid and three decades, the ICL 2966 is rendered obsolete by a cheap, mass-produced machine that a child can set up. Low power, low cost, low maintenance – the Raspberry Pi is better in every way than the ICL 2966 it replaces. Progress is fantastic. Computers are cheaper, faster, and better than they used to be. A gray Amstrad or Psion has nothing to teach me about computing, but it does tell me that today is much better than yesterday.

Things are great. And they're getting better. We're lucky to have people like the volunteers at The National Museum of Computing for their stewardship of the past, but we shouldn't think that the olden days were a golden age – the golden age is now. Unprecedented power is at your fingertips, so go and hack something, and make it beautiful. ■■■

LOST YOUR BOOKSTORE? LET US BE YOUR BOOKSTORE

Browse our shop for single issues of *ADMIN*, *Linux Pro*, *Linux Magazine*, *Raspberry Pi Geek*, *Drupal Watchdog*, and *Ubuntu User* – delivered right to your door.

■ shop.linuxnewmedia.com/single

Better yet, subscribe, and you won't need a bookstore.

■ shop.linuxnewmedia.com/subs



shop.linuxnewmedia.com

DIGITAL AND PRINT EDITIONS AVAILABLE!



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

MADDOG'S DOGHOUSE

“maddog” ponders how our actions affect others in ways we cannot predict. BY JON “MADDOG” HALL

ROI

The uncle of a friend lies in a hospital bed, dying. The last few days I have had a writer's block waiting for a horrible, deadly disease to run its course, and I think about another man's life and what it has meant to my own. We touch each other in ways we may not recognize.

The month started out fairly well. I went to CeBIT, one of the world's largest computer and telephony shows in Hannover, Germany.

I've been going to CeBIT for 16 years, representing Free and Open Source Software (FOSS) – first with a tiny 10x10 foot (3x3 meter) booth, handing out CDs filled with various distributions of GNU/Linux. Over time, one booth grew to 20 booths and gradually became a pavilion filled with vendors. The pavilion then became the “Open Source Park,” and I have been asked to present several talks every year about the benefits of using FOSS.

This year, as I was preparing for my first talk, a man came up to me and introduced himself. It turns out that 15 years ago I had traveled to Hong Kong and given a talk there. He had listened to my talk and started to follow the Free Software path; now he was a successful businessman in Hong Kong, and he wanted to know if I would visit there again and give more talks. I told him that in order to do so, I would need an invitation as well as transportation and accommodations. He acknowledged those issues and said he would be in touch.

Later that day, another man told me that 10 years before at CeBIT, when he was a college student, he had listened to me talk about Free Software. He told me that I sat with him and went over the reasons Free Software was the way to go. He later worked at a company that had a product that was “mostly” Open Source, but as the company went on, they forked the product and created a completely Free and Open Solution, and the product had simply taken off. He was now the Chief Technical Officer of his company, and this company was doing fine with an “all FOSS” product.

Then, at the coffee bar, a man offered to buy me a cup of coffee. He was from Italy, and his company made rack-mounted switching gear. He had convinced his management to go Open Source, following the advice I had given to him years before, and his management told me how much this guidance had meant to them.

Every year while attending CeBIT, a group of us go to a restaurant called the Munich Halle, where we enjoy great food, drink good beer, and talk about FOSS. This year was no exception, and as I sat there I thought about the friends sitting with me and how FOSS had brought us all together. One of the people was a young man who has been coming to CeBIT for several years. One year, he sat through all of my talks, taking notes on what I said. Naturally, I was impressed by this and invited him to join us for dinner, which he has done every year since then. This year he pulled out a Raspberry Pi, told me it was one of several that he had, and said that he loved “developing things that only geeks would love.”

After I returned home from CeBIT, a Facebook friend told me that they had followed my advice and had moved from a job working in a supermarket in Brazil to being a programmer, but was now going to pursue their interest in art, having placed some of their art in galleries in New York City, London, and other large cities. My advice to that friend had been “work hard, be visible, don't give up.”

My friend's uncle is going to die, but the uncle was lucky enough to see the results of his life affecting other people. The uncle was lucky enough to feel the love of those people that he inspired with his actions. I am lucky enough to see people who have been inspired by my talks and by the principles of FOSS.

Take some time from your busy lives and reach out to someone outside your immediate family. You may never see the final return, but it will be there. ■■■

REAL SOLUTIONS FOR REAL NETWORKS

FREE DVD

openSUSE Leap 42.2

Parrot OS 3.5 Full Edition (32-bit)

.NET on Linux

ADMIN
Network & Security

FREE DVD!

.NET on Linux

Will Microsoft's flagship framework sail without Windows?

ANGULAR 2
Web development framework

TROUBLESHOOTING NETWORKS
Solving problems with DNS, AD, and Group Policy

MICROSOFT OPERATIONS MANAGEMENT SUITE
Monitor Windows systems and Linux servers

Package Security
How Linux distros secure their...

Microsegmentation
Enhanced security with segmented data centers

Optimize PowerShell scripts with loops

INTERVIEW
openSUSE Chairman
Richard Brown

ADMIN Issue 38 £7.99
9 772045 070003 38

E.COM

FREE CD or DVD in Every Issue!

Each issue delivers technical solutions to the real-world problems you face every day.

Learn the latest techniques for better:

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

on Windows, Linux, Solaris, and popular varieties of Unix.

6 issues per year!

ORDER ONLINE AT: shop.linuxnewmedia.com

Spreading the word

The Art of Advocacy

Learn the tricks, tips, and techniques for converting friends, family, and colleagues to free and open source software.

BY MIKE SAUNDERS

When was the last time you saw a TV advertisement for your favorite Linux distribution? Or heard a radio spot about GNU? Chances are you've never come across anything like that – and you probably never will. Awareness of GNU, Linux, and free and open source software (FOSS) is spread largely by word of mouth, using grass roots movements and social media. Many of us got into Linux and FOSS because we knew of friends who were using it or read an article describing someone's experiences in a magazine. We certainly didn't start using Linux back in the late 1990s because of some shiny TV ad (Figure 1).

As a reader of this magazine, you probably don't need anyone to advocate Linux and FOSS to you. Chances are that you made the decision to use this software long ago, for your own reasons. Even if you dual-boot and just tinker with Linux as a hobby, you've still gone far enough that you're not interested in hearing someone else's arguments for using it.

Figure 1: GNU/Linux hasn't had much in the way of TV advertising, although IBM did create a video spot many years ago.



But because you're familiar with Linux and FOSS, and have your own (hopefully positive!) experiences with it, why not start advocating for it yourself? There are many different ways to contribute to the FOSS movement, including code, documentation, graphics, and QA, but spreading the word can be equally as useful. Linux and other free software projects benefit from fresh blood and new ideas, so by advocating for the FOSS community, you can help improve the software that we all use and benefit from.

But there's an art to advocacy. Saying the right things to the right people, and at the right time, is critical. You may have seen efforts to advocate Linux on web forums and online chat channels, but they often boil down to "LOL M\$ sucks, stop being a n00b and use Linux." The people posting such messages think they're doing the right thing – after all, they want more people to try the software – but such approaches are usually counterproductive.

In this issue, we'll take a closer look at the best ways to advocate Linux and FOSS among your friends, family, and colleagues. We'll analyze areas you can focus on, ways to get people interested, and how to respond to common criticisms. So, let's get started.

Choose Your Audience

First off, think about who you're trying to tempt into using FOSS and Linux (your target audience). Different people will have wildly different requirements and interests, and you should cater for them. Telling a mate who's struggling with performance problems in Windows 10 on his home PC that Linux scales up to a jillion processors and has 45 different filesystems won't have much effect – in fact, that person will just be put off. Similarly, telling a company IT boss that the (very awesome) Arch User Repository has the latest, bleeding-edge software minutes after it's released upstream won't sway him or her – they are interested in long-term support and stability.

We'll look at specific arguments later in this article, but it's important that you keep your target's needs and desires in mind. Once you have a clear picture of what they want, you can start to formulate your approach. A very important thing to bear in mind is that geek jokes and references to hacker culture can torpedo your efforts – and this happens a lot. If you've been in the FOSS world for many years, you might forget that some things sound completely alien to the wider world.

Richard Stallman, for example, gives many speeches around the world espousing the benefits of free software. By and large, they are excellent: He carefully lays out the importance of FOSS, describes how it benefits society as a whole, and makes listeners aware that with proprietary software they're not truly in control of their computers. His speeches aren't highly technical, nor are they overly emotional, but they get the point across effectively.

But one thing always makes us cringe, though: When Stallman explains the four freedoms that define free software [1], he starts counting from "freedom zero" (Figure 2). We get the reference here – in many programming languages, elements in sequences are counted from zero onwards – but this "in joke" simply baffles the unaware. Instead of focusing on the highly important message in Stallman's speeches, listeners can get distracted by this small point.

So try to avoid such complications in your own advocacy. You might be extremely passionate about something – such as using the name "GNU/Linux" to refer to the operating system (OS) as a whole – but you might want to hold back on getting into the naming debate before your target is actually interested in (or actively using) the software. Later, you can talk about how the GNU project relates to the OS we use today, its importance, and its areas of focus.

Areas to Focus On

Now let's turn to specific things you can discuss when advocating Linux and FOSS. We all have our own reasons for using it, but the world is always changing and arguments that may have been solid 10 years ago might have less impact now.

Take stability, for instance. Back in the days of Windows 98, ME, and early releases of XP, we used to make jokes about the Windows Blue Screen of Death (BSOD; Figure 3). Even though desktop Linux distributions of the time were far from perfect, and had their own flaws, we could sell Linux on stability alone: "Sorry to hear about losing your work – you may want to try Linux. I've been using it for three years, and it has never crashed on me."



Figure 2: Richard Stallman delivers superb speeches on FOSS – we just wish he didn't count the "four freedoms" from zero! (Photo source: NizoBZH, <http://tinyurl.com/m4zmsju>)

Since Windows 7 onwards, though, that argument isn't quite as powerful; Microsoft has beefed up the stability of Windows (although we still think it doesn't compare to a well-tuned Debian installation). Telling potential convertees that they can avoid BSODs and major crashes by switching to Linux won't have as much effect. You might have some success in this area when talking about massive server deployments that need extremely high availability, but many home users running Windows probably haven't seen a full-on crash for a while.

Next up is security. This is definitely an area where we can advocate Linux and FOSS over proprietary software. However, we have to be careful: Just because a piece of software is open source, that doesn't make it inherently more secure, as major bugs like OpenSSL's Heartbleed [2] have demonstrated (Figure 4). If we just wave our hands and say, "Linux and FOSS are simply more secure," and a gaping hole is discovered a few days later, we end up with egg on our faces.

Figure 3: We all used to joke about Blue Screens of Death in Windows, but today, we have to be more subtle when advocating Linux based on stability.

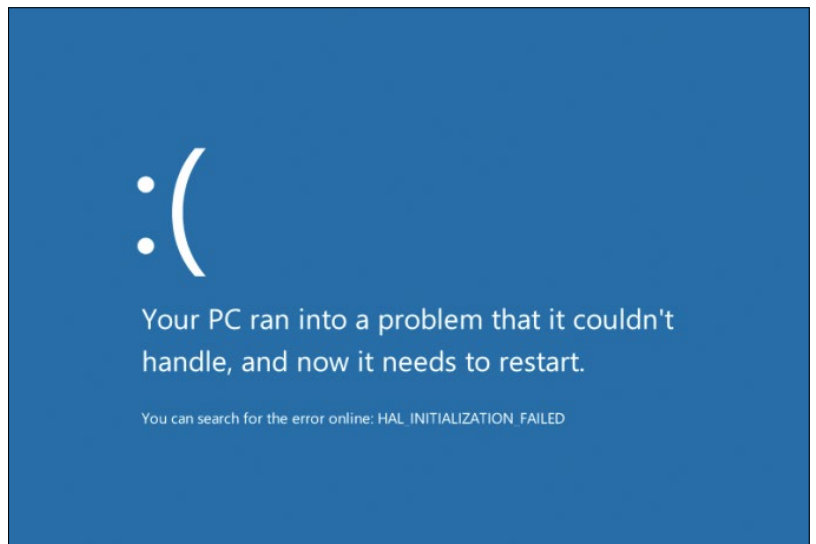




Figure 4: FOSS is not immune to security holes, as Heartbleed proved. But we can still argue that it's more secure than proprietary software.

So instead of focusing on the number of security problems discovered in one piece of software versus another, you can talk about how FOSS is better at preventing and handling such problems. As you know, with open source, many eyeballs can look through code and find security vulnerabilities before they are exploited. This is a potential benefit, as not every piece of FOSS has thousands of people working on it, but it often plays out in reality.

Similarly, you can talk about how FOSS projects are much more proactive when it comes to security. Find an example of a security hole being discovered and then being patched within hours. Compare this with an example from a proprietary software company that took days, weeks, or even months before releasing a patch.

You can extend this argument even further by talking about unknown vulnerabilities. Because we regular humans can't look at the Windows 10 source code, do we have any idea how many potential vulnerabilities there are in the operating system? Of course not. There could be some real whoppers in there being exploited by crackers and government agencies as we speak. And if those crackers are netting lots of info or money by breaking Windows boxes, they won't be in a rush to tell Microsoft...

Contrast this with Linux and FOSS. Yes, they are not perfect; yes, they have security issues as well. But with all the code in the open, such issues can be discovered much more quickly, and it's extremely hard for nefarious types to squeeze in broken code to be exploited later on. When you're advocating Linux, make your target know that you feel much, much more comfortable – and in control of your PC – by running FOSS.

Other Benefits

You can talk about other things as well, such as performance: Many Linux distros run like a charm on older hardware, negating the need to upgrade so often. The ability to deeply customize the OS could be a big win for many people as well. But, if you're tempted to talk about price, this is also where you have to be careful. Most people get Windows preinstalled with their PCs and don't realize that they've had to spend money on it.

Additionally, people can be rather mistrusting of things that are "free." Where's the catch? Do I have to pay later? Is it like a "freemium" service where all my data is uploaded to the Internet? On top of this, some people may still have the perception that GNU, Linux, and FOSS are some kind of "hippie commie" movement – which wasn't helped by Microsoft executives, in the early 2000s, saying Linux has "characteristics of communism" and is "against the American way."

FOSS and Support

There's a common misconception that there's no support available for FOSS. Sure, that may be the case for some random two-person SourceForge project that's at version 0.1.3, but many big-name FOSS apps and distros have commercial support options available. Most companies and large organizations don't switch to Linux without paying someone to provide support.

Now, this is a bit trickier for home users; they're unlikely to pay for Red Hat Enterprise Linux or Collabora Office (a supported enterprise version of LibreOffice). But when advocating Linux, if

this argument comes up, you can turn it around: Have you ever phoned Microsoft for support with Windows or Office? What sort of help did you get?

Chances are that your target has never done this nor interacted with Microsoft in any meaningful way. They've paid for "supported" software, but their options are limited. Look at the support page for Microsoft Office home users, for example [3] – it recommends visiting the Office community forums (Figure 5). Well, great! You've spent all this money, but Microsoft is asking its own users to provide support for each other.

Another Argument: You're Already Using Linux!

Here's another good way to advocate Linux and FOSS. Find out what hardware, software, and services your target is already using on a daily basis, and chances are FOSS is involved somewhere. Just make sure that your potential converttee is happy with that software before you tell them that it's Linux or FOSS, though. You don't want to start a conversation like this:

"Hey, are you using an Android phone?" "Yes, I've had it for six months now." "Oh cool. Did you know Android is based on Linux? You should try Linux on your PC as well." "Err, but I hate this phone. It's slow and buggy, and I want to get rid of it. Why on earth would I want this on my home PC?!"

So put out the feelers first, before making the connection. Similarly, find out what websites or services the person loves. If you discover that those services are built on FOSS, that's one avenue for advocacy as well – "Service X that you love is built using Linux and FOSS. If they trust it, you should give it a try, too!"

So explain to your potential converttee early on exactly why Linux is free. Discuss how it's developed, who works on it, and how it is funded (many projects like LibreOffice and Debian survive off donations). But, you can also state that there is plenty of money made in the Linux world as well – point to Red Hat, a company that makes megabucks from providing support and other add-on services for FOSS. (See the "FOSS and Support" box for more info.) Show how Google has tens of thousands of machines running Linux and contributes back.

What You Can Do

So, you have your advocacy target in mind, you've thought about what they want, and you've put together your arguments. What's next? Well, you can simply demonstrate your own FOSS workflow. Don't just hand over a USB key or Live CD – show Linux running on your own machine first. Put a guest account on your laptop and let your advocacy target play around with it. Try to use a familiar desktop environment; it doesn't have to look exactly like Windows, but it should be accessible and obvious.

You can also introduce FOSS one app at a time, before encouraging your target to try Linux. Perhaps the best example for this is Firefox: If they're using Internet Explorer, Edge

or Chrome, install Firefox and suggest they try browsing with it for a few days. Firefox has its quirks, but it really is better than the competition in many ways and your potential converttee could fall in love with it – especially if you demonstrate some useful extensions. (See the "Another Argument: You're Already Using Linux!" box for other ideas.)

Then you could try other software such as LibreOffice and Gimp. This might be a harder sell if your target is already using Microsoft Office and Photoshop, but if they're pinching pennies you may be in luck. Once that person is happily using a bunch of FOSS programs, it's a lot easier to suggest trying a Linux distro, where all of these apps are preinstalled (and often more neatly integrated than under Windows).

Above all, make it clear to your potential converttee that he/she doesn't have to commit to anything. You can try Linux and many FOSS apps and if you don't like them, don't use them! FOSS is awesome because it doesn't require market share or money (although those are nice). FOSS doesn't try to lock up your data, make you pay money, or bombard you with ads – it's there because people love to work on it. Get that message across, and you'll be onto a winner. Good luck – and let us know how your advocacy efforts get on! ■■■

Info

- [1] Free software:
<https://www.gnu.org/philosophy/free-sw.en.html>
- [2] OpenSSL's Heartbleed:
<https://en.wikipedia.org/wiki/Heartbleed>
- [3] Microsoft Office support:
<https://support.office.com/home/contact>

Figure 5: Microsoft tells its Office home users to use "community forums" for support – so what are people paying for?

Microsoft

Office Products Templates Support

Apps Install Subscription Training Admin

Contact Office Support

Office for home

Find answers in the [Office community forums](#)

Need to talk to a person?

[Answer Desk](#) is ready to help you with whatever you need.

FAQ

TensorFlow

Welcome our new artificial intelligence overlords by tinkering with their gray matter.

BY BEN EVERARD

Q Let's start at the beginning. What are tensors, and where should they flow?

A Without getting too bogged down in mathematics, tensor is just a fancy word for multidimensional array. These arrays flow through a series of steps to arrive at some calculation. Basically, TensorFlow is a programming tool for pushing big bits of data through a predefined series of steps.

Q Ah, that's interesting, but it sounds like an awfully complicated way of programming. Why would I want to go to all the trouble of flowing tensors?

A TensorFlow isn't really a programming language. It's a toolset for performing intensive mathematical computation. Although it's fairly general purpose, it was created initially for creating neural networks.

Q A what-network?

A A neural network is a computational model that's inspired by the way our brains work. Our brains are made up of billions of neurons. Each one on its own is quite simple: It collects values from other neurons, applies a weighting to each value, and combines them through a function to create an output. This output is

then sent to all the neurons that are connected to it (which typically aren't the same ones that it got the input values from).

Q I sort of understand; is there a way to get a better idea of what's going on?

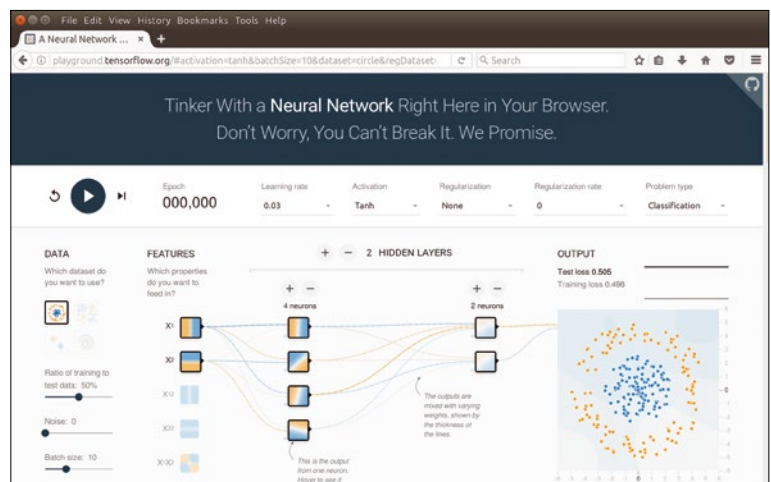
A Yes! The best beginner's visualization for this that we've come across is the TensorFlow Playground [1], which allows you to run a simple neural network in your browser (Figure 1). The playground has a simple task: Given a pattern of dots where some are orange and some are blue, the neural network has to correctly work out which dots are which color by looking at where they are on the graph. You can change the pattern you're using in the data section on the left-hand side. Some of the patterns are quite simple; others are more complicated (particularly the swirl).

In the playground, the neural network consists of a set of inputs, one or more hidden layers, and an output. In each layer, there are one or more neu-

rons that combine the output from all the neurons at the previous level and assign a weight to them. (You can see the weighting of a particular link by the boldness of the line connecting two neurons.) In each epoch, data flows through the network, and each time the network correctly classifies a datapoint, the connections that were correct are strengthened.

Click on the play arrow to start training your network, and you should see it gradually get better at classifying the orange and blue dots. Playing with the various settings to see how well different configurations can classify dots can get addictive as it's interesting to see how the different settings can make the machine more or less intelligent.

Figure 1: Neural networks can be confusing, but TensorFlow Playground makes it easy to see what's going on.



Q That's fun, but what an earth could that be used for?

A This type of problem is known as a classification problem. Given a bit of data, you have to classify it as a particular type. In this case, given a dot, you have to classify it as either orange or blue, which is quite simple, but more complex cases could be recognizing what objects appear in an image or what words appear in a piece of audio. Essentially, any time you need to recognize something, you have a classification problem that potentially could be done in this way.

Q So all you need to do is set up your neural network, feed in some data, and you can recognize stuff? That sounds awesome.

A On one level, that's correct, but there's quite a lot involved in the phrase "set up your neural network." For the simple dots classification, most networks manage to be fairly accurate, but when the problems become more complex, the particular parameters, setups, and configurations become vitally important to the accuracy of the classification.

Q Ah, OK. My machine struggles a bit with the few layers needed to classify the spiral in the playground. How on earth am I supposed to run more complex networks?

A The web browser version isn't full TensorFlow, and it isn't anywhere near as efficient as running it natively. That said, neural networks can be quite processor intensive. There's a version of TensorFlow that runs on Nvidia GPUs, and you can distribute your programs across many machines.

TensorFlow originated at Google, and the search giant has developed special chips designed to run TensorFlow programs as efficiently as possible.

Q Developed by Google you say? Does that mean they've fed my search history through a network to try and work out what sort of person searches for obscure Linux error messages and wild mushroom identification features?

A We couldn't say for sure. It's certainly possible that they use some form of artificial intelligence to try to target advertisements more effectively, but they're also probably moving into the more general area of classification, because the better you can classify something, the easier you can search for it.

Through their subsidiary DeepMind, Google is also pushing into a whole host of other areas including a controversial partnership with Britain's National Health Service (NHS). In 2016, DeepMind started a collaboration with the Royal Free London NHS Trust to work with kidney injuries. Unfortunately, neither the NHS Trust nor DeepMind actually got consent from the patients before running the data through their artificial intelligence systems.

Q Ugh. This is why we can't have anything nice.

A The field of artificial intelligence does pose some difficult questions for privacy. After all, doctor's notes are routinely stored on computers, and there are various standards that they have to meet. Does the fact that the computer has some potential level of intelligence affect this? Is there a point where computers are sufficiently intelligent that we have to consider our privacy from them not just their operators?

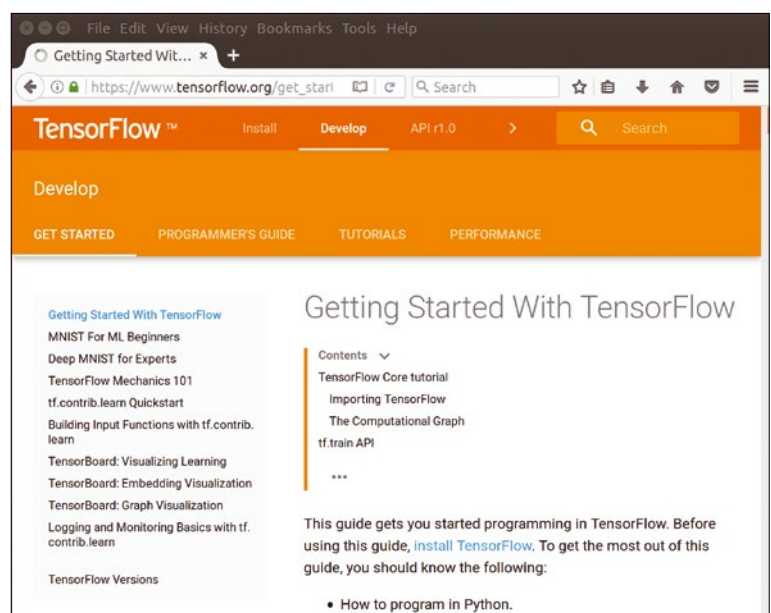
Q Invasion of privacy and philosophical questions aside, this all sounds really interesting. I suppose that you need to learn some obscure programming language known only to PhD students before you can get started with TensorFlow?

A Nope. Python is by far the most popular language for working with TensorFlow, but there are also bindings for C++, Java, and Go. The mechanics of using TensorFlow are actually quite straightforward once you've got your head around the flow graphs. However, understanding what you need to set up to get the artificial intelligence you want can be quite challenging. That said, provided you start slowly and build up your understanding, TensorFlow is within the abilities of the average geek. The getting started guide [2] will, as the name suggests, help you get started (Figure 2). ■■■

Info

- [1] TensorFlow Playground: <http://playground.tensorflow.org>
- [2] TensorFlow Getting Started Guide: https://www.tensorflow.org/get_started/get_started

Figure 2: The TensorFlow website can guide you through learning how to build your own artificial brain.





Valentine Sinitsyn works in a cloud infrastructure team and teaches students completely unrelated subjects. He also has a KDE develop account he never really used.

CORE TECHNOLOGY

Learn what's going on in your network, using Linux and its arsenal of packet capture tools. BY VALENTINE SINITSYN

Network Sniffers

We are always told that eavesdropping is bad. In social relations, that's probably true, but in computing (especially networking) where this activity is known as sniffing, it's an indispensable debugging technique. What goes in the wire is an ultimate answer to "What you've thrown at my service?" and "How did I reply?" Debugging aside, network sniffers may collect statistics or perform security monitoring. Linux comes with many tools of this kind, both GUI and terminal based. In this Core Tech, we'll discover perhaps the most popular one (or just my favorite).

Although sniffing is a legitimate technique, it is still largely prohibited in corporate environments. Many ISPs deem it illegal, too, so be careful when experimenting. A virtual machine based dedicated test lab is the safest option. Anyway, employ common sense and don't sniff traffic that could be sensitive, even if your housemates or colleagues are careless enough to send it unencrypted.

Back to the Beginning

Any network sniffer relies on the operating system's ability to forward it to all packets the network card receives, regardless of which process (or even host) they really target. The exact way of doing this is platform-specific. In Linux, packet sockets are the standard mechanism [1].

The `AF_PACKET` address family operates on raw Layer 2 packets (or frames). An "address" of this family (`struct sockaddr_ll`) tells the kernel from which interface to sniff packets, and in which Layer 2 protocols you are interested. Packet sockets come into play early, shortly after the network card receives a frame, and well before the data goes through the Linux networking stack. Capturing from the wire and crafting raw Layer 2 datagrams is a sensitive operation, so only processes with `CAP_NET_RAW` capability can do it. We've discussed

capabilities in the last issue of *Linux Magazine* [2], but typically it just means you need to be `root`.

A rare network sniffer uses packet sockets directly. Most rely on `libpcap` [3], a library that abstracts away platform specifics and adds some bonus features. For example, it can save captured packets in so-called `.pcap` (packet capture, you guessed it) files and later you can read, or "replay" a `.pcap` file. This way, you (or someone acting on your behalf) can capture traffic on one box to analyze it on another. The `libpcap` library is designed so that from the application point of view reading a `.pcap` file is almost indistinguishable from capturing content live, so replay is usually doable with any sniffer. Because `.pcap` files are, well, just files, reading from them does not typically require root privileges.

Another problem `libpcap` solves elegantly for you is getting only data you want. Modern networks may run at whopping 40Gbit/sec or more, yet you might be interested in that small VoIP session your boss is complaining about. In other words, sniffing is closely intermingled with filtering.

A technology called Berkeley Packet Filters (BPF) was conceived long ago just for these purposes, and `libpcap` implements a high-level filtering language it compiles into BPF bytecode. This doesn't sound particularly intriguing until you realize that you can safely put arbitrary BPF bytecode in the kernel: It can never hang or crash (by design). Moreover, Linux compiles BPF bytecode into native CPU instructions, so these filters are also fast. On the rare occasion when `libpcap` finds it impossible to execute a filter in the kernel, it resorts to userspace emulation, which is a bit slower. BPF is a really powerful concept, and it was recently extended to include tasks that have nothing to do with network filtering, such as performance profiling. This is a whole set of opportunities, but we'll leave them for another time (drop us a line if you are interested in seeing more).

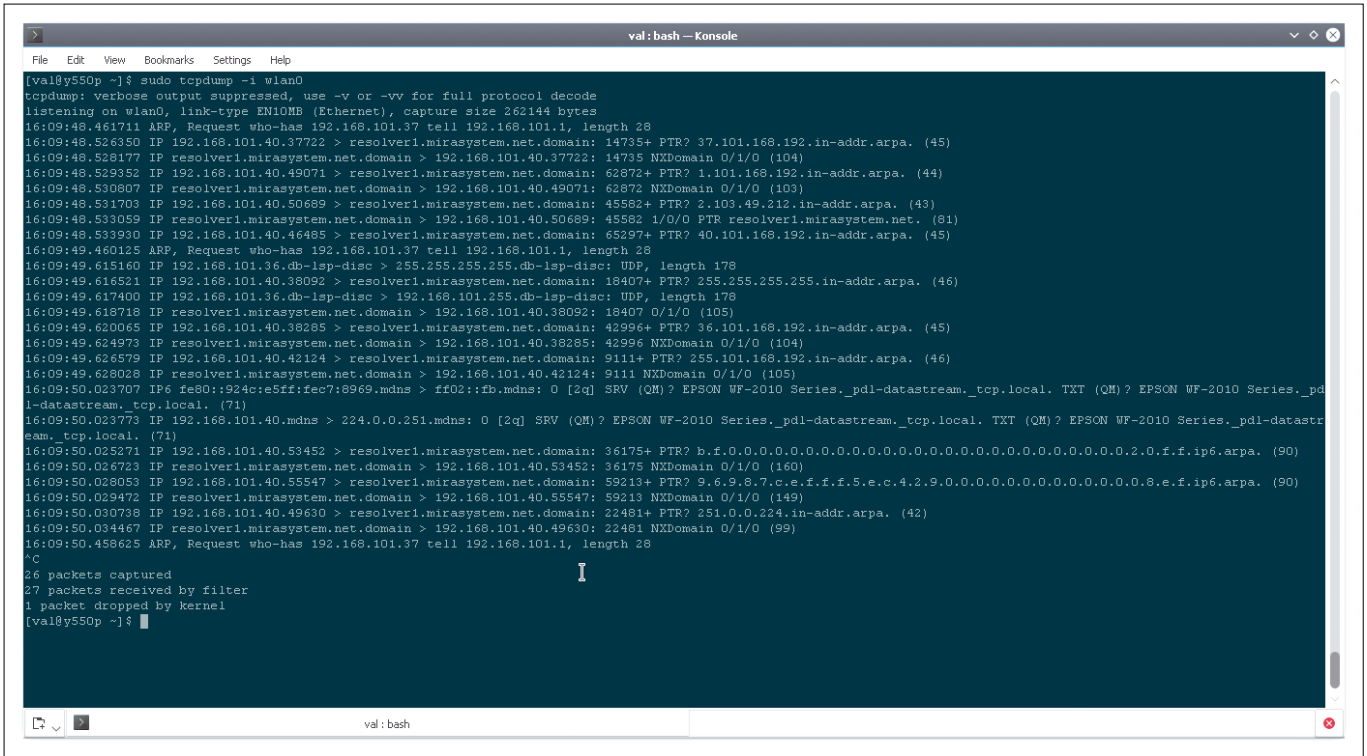


Figure 1: The tcpdump tool, dumping flows from a wireless network. You may see reverse DNS requests and some mDNS exchange with the local printer.

Dumping TCP (and Whatever Else)

If you'd want me to vote for a single network sniffer in Linux, I'd go for tcpdump (Figure 1). It may not be as good at decoding protocols as some commercial ones (although I never felt it was limited), and it has less eye candy than Wireshark (Figure 2). However, it seems to be installed on every Linux box I have access to, so it wouldn't be an exaggeration to call tcpdump a de facto standard.

The tcpdump sniffer is a text-mode command-line tool. You pass it some options and read the dump from your terminal window or put it in a file. Despite the name, tcpdump is not about TCP. It understands about 150 network protocols, spanning the majority of OSI Model layers: from Ethernet and ARP through IPv4/IPv6, TCP, UDP, SCTP, and friends to DHCP, HTTP, and BGP. For something really exotic or new, you can always get a raw hex dump.

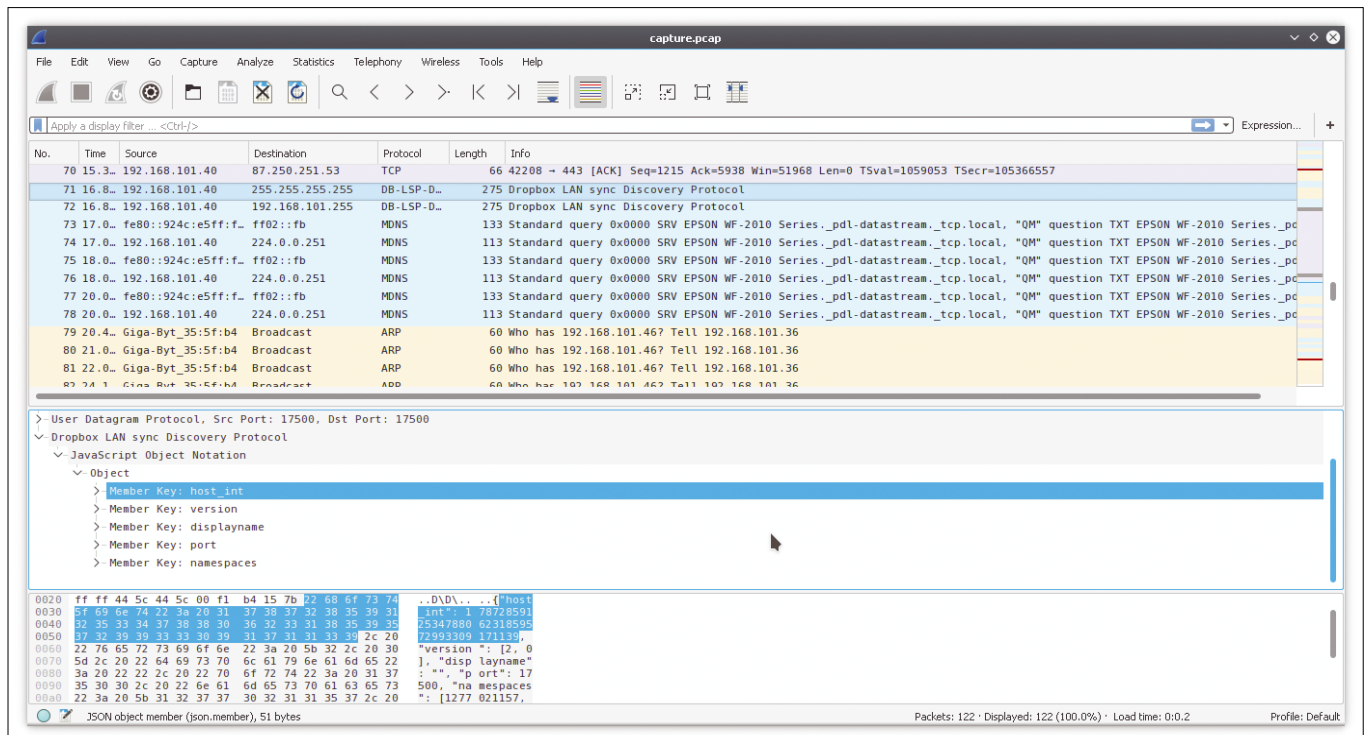


Figure 2: Wireshark is a popular GUI network sniffer for Linux. Its cousin, tshark, does almost the same from the command line.

Before anything else, you should tell `tcpdump` from which network interface to capture packets. You do this with the `-i` switch, and it is also possible to monitor all NICs in the system with `-i any`. Moreover, `tcpdump` can list network interfaces it can capture from:

```
$ tcpdump --list-interfaces
1.wlan0 [Up, Running]
...
```

Note that the interface must be up (`ip link set up dev wlan0`) to be usable in `tcpdump`. As I mentioned previously, live packet capture implies root permissions, so don't forget to prefix your commands with `sudo`. I do this in Figure 1, where you can see some flows from my home wireless network. By default, a `tcpdump` session lasts forever until you type `Ctrl+C`, but you may instruct the tool to receive a predefined number of packets (`-c`). It also puts the device from which you capture in so-called "promiscuous" mode, where the device receives frames destined to other hosts. Usually, that's what you want, but if you are interested in local traffic only, consider adding `-p` to disable promiscuous mode.

You see that `tcpdump` prints a human-readable summary for each Layer 3 datagram it receives. Add `-e` to include Layer 2 header information, such as MAC addresses (and the respective NIC manufacturers). You can also make `tcpdump` dump more data with `-v`, `-vv`, or even `-vvv`, where `v` stands for verbosity, and the more `vs` you have, the more verbose `tcpdump` is. (This is a recurring pattern in its CLI.) A maximum verbosity level ap-

Figure 3: `tcpdump` can be quite verbose. Here, you get a full protocol decode and a complete hexdump of two Dropbox LAN sync discovery packets.

pears in Figure 3. I usually run `tcpdump` in this mode myself and also add `-n` to disable host/port/whatever else names from being resolved. Without it, `8.8.8.8` would become `google-public-dns-a.google.com`, and `80` would read `http`. This is useful most of the time, but when I do network troubleshooting, I prefer clear pictures.

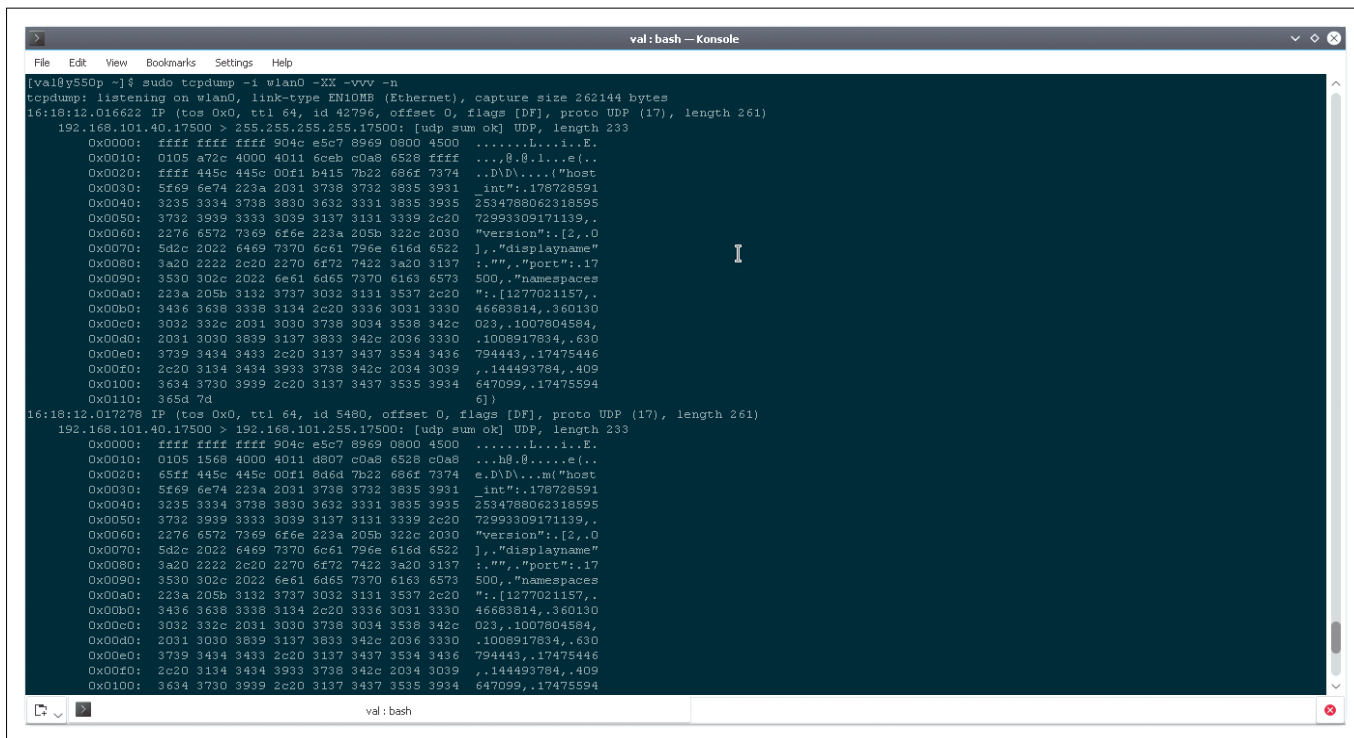
Additionally, `tcpdump` does a great job of decoding network protocols, but if it prints some packets as garbage, you may want to look at raw bytes to see what's wrong. You may see that the packet is indeed well-formed, with just a few extra bytes at the beginning. `tcpdump` facilitates this workflow with `-x` and `-X` switches. Both print raw hex dumps, but the latter also adds ASCII as most hex editors do (Figure 3). A corresponding double-`x` version (`-xx` and `-XX`) includes Layer 2 headers in the dump.

So far, `tcpdump` is dumping everything it sees in the wire on your terminal. It is also completely possible to store packets in `.pcap` files:

```
$ sudo tcpdump -i wlan0 -vn -w first.pcap
tcpdump: listening on wlan0, link-type EN10MB (Ethernet), capture size 262144 bytes
Got 77
```

The counter indicates how many packets were captured. If you are going to grab a lot of data, `tcpdump` can spread it across multiple files that act as a "ring buffer." Consult `tcpdump(1)` man page [4] for details. (See also the "PF_RING" box.)

To read a `.pcap` file, use `tcpdump -r` (recall it doesn't require root permissions). Note that `.pcap`



stores not only packets but also timestamps, so expect it to run as long as your capture session lasted. You can also open a `.pcap` in another sniffer, as shown in Figure 2.

Sometimes, you may see `tcpdump` complaining about checksum errors in packets coming from your local Linux box. It is completely normal: Many modern network cards compute checksums in silicon (this is called “checksum offloading”), so the kernel doesn’t bother filling the respective fields. Packet sockets operate before the packet is handed to the NIC, so checksum fields may contain incorrect values. If these complaints annoy you, disable checksum calculation in `tcpdump` with `-k`.

Filters Galore

If you tried the preceding examples on a busy network, you may have noticed that they produce too much data to be really useful. As I mentioned earlier, `libpcap` (and `tcpdump`, for this reason) provides a high-level filtering language that compiles into BPF. The `pcap-filter(7)` man page [6] describes the formal syntax, so you can go through some typical examples there.

Filters consists of primitives joined by logical operations, such as `&&/and` or `||/or`. Some filters span a single primitive. Conventional relation and Boolean operators are also supported. At the next level, primitives consist of an ID, which is the value you filter for (say, an IP address or a port) and some qualifiers. The latter tells if the ID is a host address, a port, or whatever; whether it refers to the source or the destination; and which protocols you are after. If omitted, the type of ID is assumed to be `host`, the direction is `src` or `dst` (i.e., any), and any protocol where ID makes sense would match. For example, `port 80` is the same as `(tcp or udp) port 80`, and `host 1.2.3.4` matches `ip, arp, or rarp`, but not `ip6`.

Most of the time, you would filter by host: `host 192.168.0.2` does the job. You are free to use a DNS name instead, and `tcpdump` will resolve it at compile time. Such a filter grabs both ingress and egress traffic for the host. If you are interested only in packets where `192.168.0.2` is the source or the destination (or both), just prepend the filter with `src`, `dst`, or `src and dst`, respectively. The latter is quite rare. It is also a good idea to filter out ARP – which you are not normally interested in – to reduce the clutter: `host 192.168.0.2 and not arp`.

You may also be interested in traffic for a particular service, such as a web server. Here you need two primitives: one saying that the server’s IP address must be involved as a source or a destination, and another saying you want TCP traffic to or from port 80. In `tcpdump`’s parlance, this reads `host 192.168.0.1 and tcp port 80`. For trickier protocols

PF_RING

Packet sockets are standard in Linux but not particularly fast. This is usually not an issue for debugging but may cause problems in network monitoring scenarios. If the capture mechanism isn’t fast enough, you’ll be losing events you were to monitor. Not good.

PF_RING [5] is one approach to making packet capture in Linux faster. It doesn’t probably come with your distribution kernel, but you can build it yourself or download a pre-packaged version for Ubuntu/Debian or Red Hat/CentOS. PF_RING is as a kernel module with its own userspace API, but it also provides the custom `libpcap`, which you can use as a drop-in replacement. PF_RING plugs into the Linux kernel’s NAPI subsystem to poll packets from the network card and pass them to userspace via a set of ring buffers. This helps to distribute packets better on multicore CPUs, resulting in higher throughput (and CPU usage). The authors claim that a PF_RING-aware `libpcap` is 12-34 percent faster than a vanilla one, depending on the NIC driver. A zero-copy variant, PF_RING ZC bypasses the kernel almost entirely, but it’s a proprietary software (PF_RING is GPLv2). This “drawback” aside, it claims to be able to operate at line rates, that is, capture packets at 10Gbit/sec on a 10G network.

like BitTorrent, you may want to specify multiple ports with `port range`.

You can also filter by network addresses (`192.168.0.0/24`) rather than hosts. You may set limits on the packet size with `len`. For advanced cases, you may access individual bytes at known offsets using array index notation. The man page has a good example: `ip[6:2] & 0x1fff = 0` drops IPv4 fragments other than the first (or the only) one. This filter takes a 16-bit half word at the offset

Listing 1: A BPF Assembly Code for “src 1.2.3.4” Filter

```
01 (000) ldh      [14]
02 (001) jeq     #0x800          jt 2   jf 4
03 (002) ld      [28]
04 (003) jeq     #0x1020304      jt 8   jf 9
05 (004) jeq     #0x806          jt 6   jf 5
06 (005) jeq     #0x8035        jt 6   jf 9
07 (006) ld      [30]
08 (007) jeq     #0x1020304      jt 8   jf 9
09 (008) ret     #262144
10 (009) ret     #0
```

6 in IPv4 header and masks the highest three bits (or flags) to get a fragment offset value. If it's zero, the packet is either non-fragmented (most likely) or the first fragment in a series. To tell the former from the latter, you can check the MF bit within masked flags.

For those of you wanting to know how all of this works under the hood, `tcpdump` can print a low-level BPF representation of your filter (Figure 3). Three modes are available. With `tcpdump -d`, you get a human-readable BPF disassembly, `-dd` dumps a C code snippet to use in your programs, and `-ddd` produces raw bytes you can paste into whatever BPF interpreter you want.

Look at Listing 1 for a BPF disassembly of a simple filter: `src 1.2.3.4`. I got this output with `sudo tcpdump -d src 1.2.3.4`. Note that while the compiler itself is unprivileged code, `tcpdump` still needs `CAP_NET_RAW` and complains of insufficient permissions to capture otherwise. This is minor but annoying.

First, the code loads a half-word at offset 14. It's EtherType, and it stores Layer 3 protocol the Ethernet frame encapsulates. If it's 0x800, or IPv4, a 32-byte word at offset 28 is loaded and checked against 0x1020304 (1.2.3.4 in network byte order). That's where the Source Address field in IPv4 is: Ethernet header is 16 bytes long, and Source Address is at offset 12 within the IPv4 header. Other branching instructions check for ARP (EtherType

0x806) and RARP (0x8035). We discussed these protocols in a previous issue [7]. The program returns a number of bytes to take from the packet. If the condition matches, it's 256K or the whole packet (in IPv4, it is never more than 64K; typically 1 to 10K). If not, it's zero, and the packet is discarded. ■■■

Info

- [1] packet(7) man page: <https://linux.die.net/man/7/packet>
- [2] "Core Technology" by Valentine Sinitzyn, *Linux Magazine*, Issue 198, pg. 76
- [3] lipcap library home: <http://www.tcpdump.org/pcap.html>
- [4] tcpdump man page: http://www.tcpdump.org/tcpdump_man.html
- [5] PF_RING: http://www.ntop.org/products/packet-capture/pf_ring/
- [6] pcap-filter(7) man page: <https://linux.die.net/man/7/pcap-filter>
- [7] "Core Technology" by Valentine Sinitzyn, *Linux Voice*, Issue 31, pg. 94
- [8] Scapy homepage: <http://www.secdev.org/projects/scapy/>
- [9] "Core Technology" by Valentine Sinitzyn, *Linux Voice*, Issue 28, pg. 94

Command of the Month: Scapy

`tcpdump` and friends do a decent job, but they are mere observers. On the other hand, Scapy [8] can not only capture and decode packets but also forge them and send them over the wire.

We devoted an entire Core Tech to Scapy more than a year ago [9], but it never hurts to refresh the basics. Scapy is a Python framework for dissecting network protocols, and an associated interactive tool, `scapy`, built on top of it. Each protocol is represented as a Python class, and you combine them with `/` to build protocol stacks or read their properties to get protocol fields you want. Knowing Python is a bonus for using Scapy (and many other tools, in fact), but not a must. Indeed, do you need it to read the following?

```
>>> Ether() / IP(dst='1.2.3.4')
<Ether type=0x800 |IP dst=1.2.3.4 |>>
```

That's self-explanatory. Scapy could be a sniffer if you tell it to read packets from the network or a `.pcap` file. Or it could be a fuzzer if you use it to forge random packets and throw them at the target. It can even build beautiful packet visualizations for you, as seen in Figure 4. It doesn't matter whether you are a network professional or just a hobbyist: Having Scapy under your belt may save you some day. ■■■

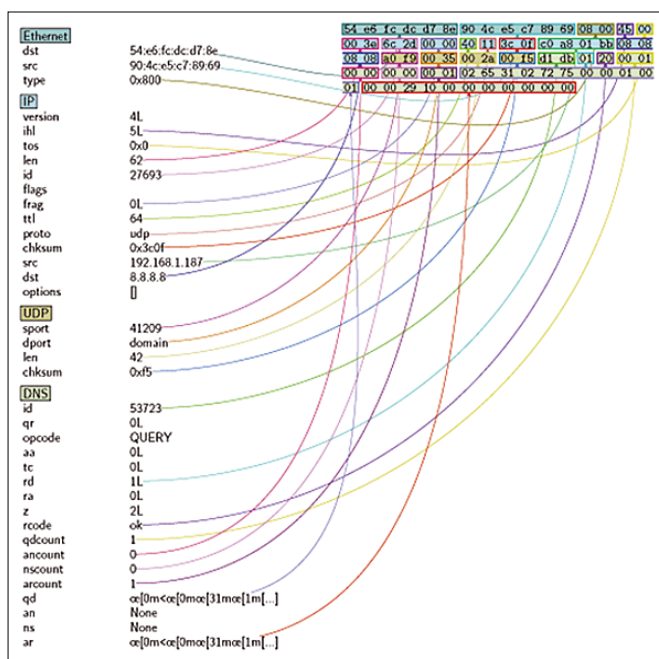


Figure 4: Nearly all sniffers are network protocol decoders, but Scapy also shows you where each value came from.



What?!

Archives
come with my
digital subscription?



Archives + Current!

Sign up for a digital subscription to get the latest issues of Linux Magazine, PLUS access to archive articles.

shop.linuxnewmedia.com/digisub

That's a lot of articles!

FOSSPicks

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



Graham tears himself away from updating Arch Linux to search for the best new free software. **BY GRAHAM MORRISON**

Audio analyzer

Sonic Visualiser 3

One piece of measuring and analysis hardware is available on nearly every computer, whether that's a smartphone, a Raspberry Pi, or a big box PC: it's the audio interface. Most of us only become aware of the audio interface when PulseAudio stops playing back our favorite music or when a video chat doesn't detect the microphone. But audio signals are their own fascinating world and one that doesn't need to be restricted to music or even what you might consider audio. Sonic Visualiser is a portal into this magical world.

As its name suggests, Sonic Visualiser doesn't depend on the

subjective and variable world of hearing. Instead, it's all about how audio appears and how it can be measured. Initially, the waveform view will seem familiar to anyone who's played with Audacity or Ardour – it's a waveform with time on the horizontal axis and amplitude on the vertical axis. It shows when a sound gets louder and quieter and is often the most useful view for editing audio. But there are almost no editing tools in Sonic Visualiser. Click on what looks like a pair of compasses, for example, and you can measure the variation between two points on the waveform, with Y showing the sampled voltage and X showing the time – just like an oscilloscope.

But the power really comes from changing what you're looking at. As in Gimp, this can be done with layers, but it can also be done by adding new panes below the original. Each is capable of rendering different representations of the same audio, complete with a totally different set of values to measure. Add a spectrogram, and the color is used to show amplitude and the vertical is used to show frequency. A melodic range spectrogram and a peak frequency spectrogram interpret the frequencies as notes, potentially visualizing a melody. A final view shows a slice through one moment in time, giving you the powers of an audio dendrologist, with the frequencies split from left to right above a piano keyboard, and variable amplitude for these frequencies on the vertical. There's even a note layer that takes this a step further, showing the contents of an audio file as if they were MIDI notes.

All this sounds(!) more technical than it really is, and the best way to understand how Sonic Visualiser works, and what it's capable of, is simply to play with it. You can change the rendering palette for each chart, the scale that's used, and the sampling window. The UI updates smoothly, and the application always looks fantastic. Even better, though, is the plugin system that lets you add some serious functionality from external sources. There's quite a selection to choose from, including a collection of audio algorithms from the BBC's Research and Development department, harmony and chord extraction plugins, beat trackers, tempo trackers, and audio transcription plugins. You can even use more traditional audio plugins to change the sound and analyze the output, which is an excellent way of reverse engineering their functionality. But Sonic Visualiser can obviously do so much more. It's an audio geek's playground, and it all works brilliantly.



1 Layers and panels. One view can lay on top of another and different panes can be shown side by side. **2 Regions.** Regions can be marked and even labeled. **3 Editing.** Measure audio characteristics and even perform simple copy and paste. **4 Appearance.** Colors and scales can be changed per layer and per panel. **5 Shortcuts.** Almost everything has a keyboard shortcut, and there's a reference window to help remember them. **6 Pitch tracking.** Many plugins can detect the pitch and note values of a frequency. **7 Waveform.** The traditional waveform view is also available, alongside a spectrum view. **8 Converter.** Convert sampling frequencies to pitch and beats.

Project Website
<http://www.sonicvisualiser.org/>

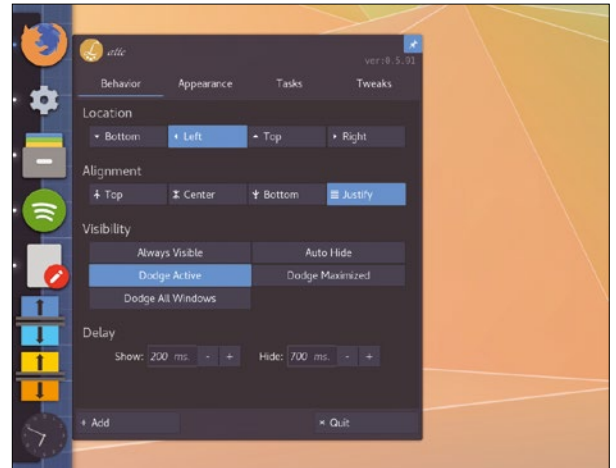
KDE panel

Latte Dock 0.5.91

One of the best things about KDE – and there are many – is the functionality contained within the panel. It works well horizontally stretched across the entire display, or as a small panel across half. It works well vertically, as well as floating in the middle. You can have more than one, and each one can be configured to do as much or as little as you want. But the default KDE panel can still feel a little utilitarian, partly because it's nothing like the panel in OS X. This is why there are perhaps so many panel replacements that look more like Apple's similar dock. Until now, there hasn't been a decent option for KDE users.

Latte Dock is the best alternative I've seen for KDE. Even in this early state, it's perfectly usable.

And unlike most panel replacements, it's got a comprehensive set of configuration options that mimic much of what KDE's default panel does. By default, it will appear in the middle of your screen's bottom border when your pointer gets close. The icons for running applications appear and enlarge as you roll over these in beautifully animated transitions. The same happens when you open the configuration panel, which allows you to change the location of the panel and its alignment. You can place it anywhere and get the icons to center exactly as you wish. You can also control the transitions and zoom levels, as well as enable or disable the panel background and running application highlight modes. On a high DPI desktop, it looks absolutely fantastic. The



Drag and drop applications, and even Plasmoids, into this gorgeous panel replacement.

SVG icons scale perfectly, and replacing the old KDE panel with this is a serious temptation, despite it not fully supporting functional applets like monitoring tools or desktop pagers.

Project Website

<https://github.com/psifidotos/Latte-Dock>

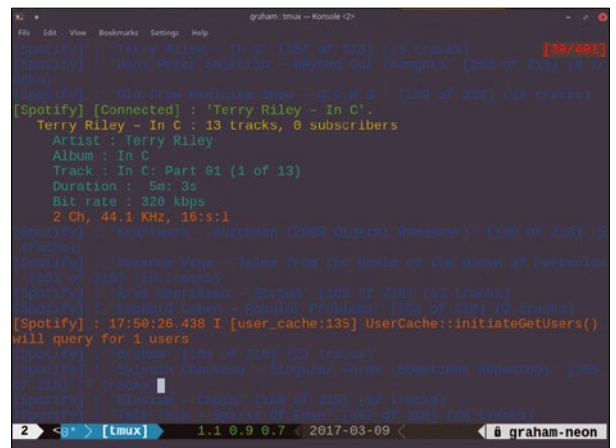
Cloud music player

Tizonia 0.7.0

Online streaming music services, even the ones you pay for, can be unrivaled portals to both new music and old. But you're often bound to the user interfaces provided by those services. Spotify doesn't do a bad job with its Linux client, but Google Music's woeful web interface doesn't even let you arrange your collections in any meaningful way, let alone let you arrange albums as you want them arranged. What's needed is a simple Linux tool that plays the music you want in the way you want – preferably from the command line.

This is what Tizonia does. It connects to one of its supported online streaming services and usually allows you to play specific playlists and

songs, although features change slightly depending on the specific service you're using. It currently supports SoundCloud, Spotify, Google Music, Dirble, YouTube, and SHOUTcast, alongside local audio files, and credentials can be fed either through the command or a comprehensive set of configuration files. With credentials added, you then pass an argument to tell Tizonia which service you want to use, followed by the ID or name of the song, album, or playlist. Spotify currently only has playlist support, but if you've saved an album as a playlist you could play it by typing `tizonia --spotify-playlist 'Terry Riley -- In C'`. You will then hear your chosen music from the command line. This is of course useful on the desktop if



Rather than using `ffmpeg`, `libav`, `gststreamer`, or `libvlc`, Tizonia's playback is based on the OpenMAX IL 1.2 framework.

you don't want to run a huge audio application or access a slow web portal, but it's even more useful on a small low-powered device perhaps connected to an amplifier. Simply start playback and use any MPRISv2-compatible remote control to handle playback without going back to the terminal.

Project Website

<http://www.tizonia.org/>

Mac data recovery

HFS+ Rescue 3.3

HFS+ Rescue – or `hfsrescue` as it is on the command line – is an absolutely remarkable tool that saves Apple Mac hard drive data from disaster. This may seem like an odd thing to write about in a magazine that deals predominantly with Linux, but it's not. Many of us install Linux on Apple hardware and keep the old Apple operating system around for one reason or another – OS X is still excellent for audio software, for example, and some designers still refuse to give up on the Adobe hegemony. HFS+ is the journaled filesystem used on these machines, and just like any other filesystem, it can go wrong.

As this is Apple's ecosystem, native tools that run on OS X are

no longer permitted on Apple's App Store (because they require permissions to modify the underlying filesystem structure, of course!), and almost every other option is proprietary and expensive. Fortunately, Linux and open source are here to save the day. HFS+ Rescue is a really well-engineered tool that you can run against an HFS+ formatted partition that will attempt to rebuild the entire contents of the drive, including recently deleted files, on your Linux drive. It does this via six different steps that need to be run separately. Step 1 will scan for files; step 2 cleans up the database; step 3 restores those files; step 4 restores the directory structure; step 5 moves those files into the correct directories; and step 6 cleans up.

```

root ~ # cd ~/Documents/podcast_rescue; hfsrescue -s1 /dev/sdb2
hfsrescue 3.3-rc1 2017/03/03 By Elmar Hanlhofer https://www.plop.at
Start: 2017/03/10 09:39:23
Signature:                0x2b48, H+
LastMountedVersion:      HFSJ, last mount by Mac OS X.
FileCount:                1081352
DirCount:                 202881
BlockSize:               4096
TotalBlocks:              73151175
AllocationFile StartBlock: 1
ExtentsOverflowFile StartBlock: 13968
CatalogFile StartBlock:  562576
Total size:                279 GB
100.00% scanned (279.05 GB). Found: 272618 directories, 1155584 files.
End: 2017/03/10 10:01:10
Elapsed time: 21.78 minutes.
Done.
=====
Next step: STEP 2, cleanup file database.
Next command: hfsrescue -s2

```

Charge huge consultancy fees to your Apple-owning friends by saving their corrupt hard drives with open source.

You'll end up with everything hopefully in a local copy of your Mac drive, with non-unique or undeleted files in a separate folder. And the most important thing is that it works. I tested this on a 250GB SSD with a Mac data partition, and the entire drive was restored to my home folder in around four hours.

Project Website

<https://www.plop.at/en/hfsrescue/index.html>

Byte converter

bcal

This is another brilliantly useful tool that performs one simple job extremely well – `bcal` converts between storage formats and performs various calculations. This is important because there's plenty of confusion over what certain numbers represent and how they're valued. GiB is a good example, as this is often used as a replacement for GB representing gigabytes, which itself gets confused for all kinds of values for data across a network. Typing `dca1 50 gib` will give the canonical answer – 52428800 KiB, 51200 MiB, 50 GiB, or 4.8828125000e-02 TiB, as represented by the IEC standard (base 2). The SI standard (base 10) is also output, alongside decimal and hexadecimal address values

plus both LBA offsets. These details are obviously going to be handy for anyone working with storage, but it's also useful for anyone worried about space or who wants to work out exactly what they're paying for with their latest SSD purchase.

Additionally, `bcal` can be used for all kinds of representation and conversion. You can show the binary, decimal, and hex representations of a number, for example, using the `-c` argument followed by the number. You can convert between logical block addressing (LBA) and

```

graham ~ # master -> build > bcal > ./bcal 50 gib
UNIT CONVERSION
53687091200 B
IEC standard (base 2)
52428800 KiB
51200 MiB
50 GiB
4.8828125000e-02 TiB
SI standard (base 10)
5.3687091200e+07 kB
5.3687091200e+04 MB
5.3687091200e+01 GB
5.3687091200e-02 TB
ADDRESS
dec: 53687091200
hex: 0xc80000000
LBA:OFFSET
sector size: 0x200
dec: 104857600:0
hex: 0x640000:0x0
graham ~ # master -> build > bcal >

```

Never get your kilobytes, megabytes, gigabytes, and terabytes mixed up again.

cylinder-head-sector (CHS) with `-f`, and you can work out storage capacity by adding `-s`, followed by a sector size, such as `bcal 0xaabbcc kb -s 4096`. Because the values are output all at once, the tool makes it easy to see what the differences are and why there's so much confusion around their use.

Project Website

<https://github.com/jarun/bcal>

Another brilliantly useful tool that performs one simple job extremely well!

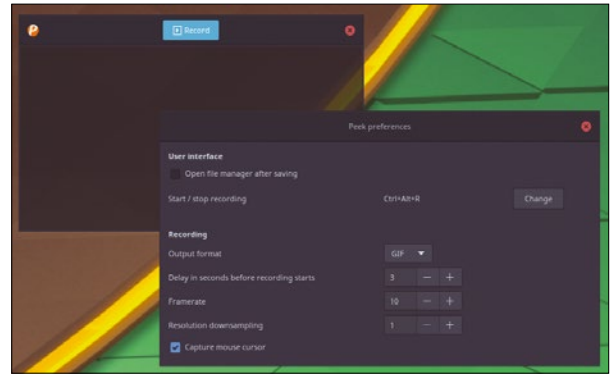
Screen capture

Peek 1.0

You can't seem to scroll down a news site in peace these days without being bombarded by animated GIFs. These little low-res movies often start without prompting and run on short loops until you either close the tab or relent and watch the full thing. But animated GIFs are an ancient technology, and they can't be all bad if they've lasted this long. They're small, resource light, and supported by almost every platform, from the Commodore Amiga to the iPhone 7, which is what makes their ubiquity part annoyance and part a handy addition to static websites – when used cautiously. And that doesn't mean as a vehicle for memes and the emergence of a meme

dialect that will one day supplant traditional languages.

In other places, however, these little low-quality movies are genuinely useful, and one example is as a way to quickly show something happening on screen. This could be because you're writing documentation for a piece of software or sharing a tutorial on how something is accomplished. Or, it could simply be the way you share discoveries on your website. Either way, creating short animated GIFs of your mouse or keyboard interacting with something is often much easier to understand than writing 200 words to do the same thing – which is exactly what Peek does so well. It opens a small window that is used to grab a section of your screen at a predefined frame rate.



Sadly, this isn't a new version of the peek and poke commands, but a screen capture to GIF utility for your cat videos.

It's simple, easy to use, and generates an animated GIF instantly. The latest version will even generate WebM and MP4 files too, which makes Peek far more useful as a general purpose screen grabber. If you need a quick and easy tool to grab your screen, much like animated GIFs themselves, Peek is perfectly suited to the task.

Project Website

<https://github.com/phw/peek>

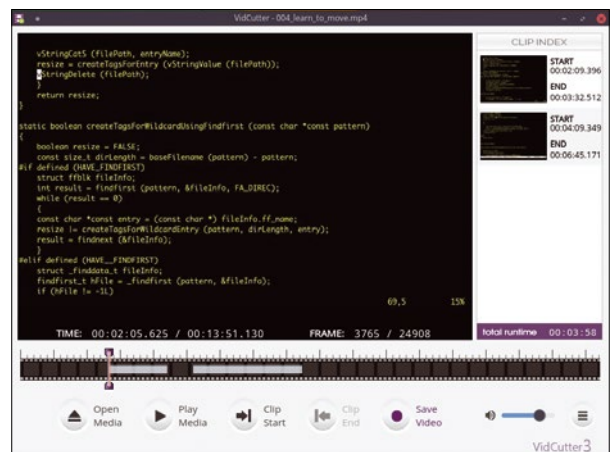
Video splitter

VidCutter 3

If you create lots of videos, such as screencasts of your desktop (see above), you often need to cut bits out of them. This isn't just to turn them into punchy, meme-friendly snippets of your gerbil olympics; it's an essential part of both the recording and editing process. Editing is obviously the central process in all video editors, but its significance can overburden applications with similar processes. Often, all you want is the video equivalent to selecting a piece of text and pressing delete. VidCutter aims to do just that – make practical edits quick and easy – and it mostly succeeds.

The UI is refreshingly simple. Most videos will load without conversion, thanks to the libmpv back end, and you can

immediately start playing with the edits. The basic process mimics that of fully fledged editors while restricting options to the minimum. You go through a file creating clips by selecting the start point and end point of each clip. The mouse wheel or left/right cursor keys can be used to step a single frame forward or backward, which is excellent, while up and down keys will skip five seconds. Return can be used to quickly create a start or end marker for a clip, and a thumbnail of each clip appears in the right after being defined. You can reorder the clips by dragging and dropping them within this panel, and when you've finished making clips and dragging them into a sequence for your final video,



Using Qt, VidCutter remains lightweight and easy to use and can even output EDL files for further editing with other applications.

clicking Save will output the final version to a new file. There are no effects or audio processing, which is why VidCutter is so effective. You often don't need those facilities for instant uploads and quick funny clips.

Project Website

<https://github.com/ozmartian/vidcutter>

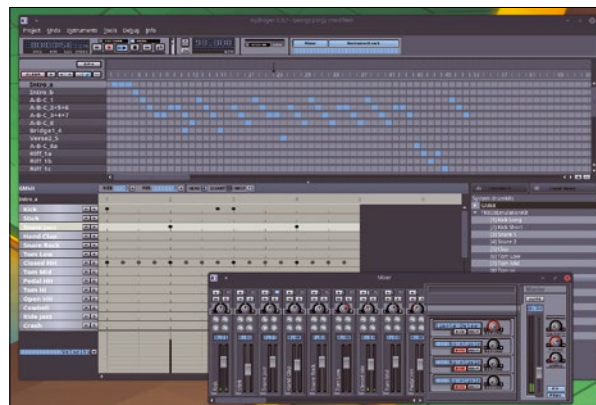
Drum machine

Hydrogen 0.9.7

Drum machines are awesome – even if you’re not into making music, and even if you have no sense of rhythm. That’s because they’re instantly addictive and can produce excellent results with just a few random presses (see modern pop music for references). Early software versions, such as Firebird’s MicroRhythm, were even sold alongside cassette games for the Commodore 64 back in 1986. Over 30 years later, Hydrogen is performing a similar role, albeit with open source and on our favorite operating system, alongside OS X and Windows, if you must. Hydrogen is a software drum machine that lets you load bundles of sound samples, usually recorded from a vast array of original drum machines, and trigger those samples from a grid. Even hardware

drum machines have worked this way from the beginning, letting the user program their own sounds and patterns, which can then be sequenced together into a song.

Hydrogen does all this and more. It attempts to use Jack for reroutable audio and for synchronization, but will also work without Jack if you’d rather quickly create some beats. You can then load samples and create instruments by layering these sounds while editing parameters such as pitch and gain. Layers allow you to change the sound dynamically as the intensity of a hit increases or to change the sound completely, if that’s what you’re after. Another important feature is that every instrument has its own volume control, and there are four separate sends for effects. This means you can create Phil Collins’ trademark



The latest version of Hydrogen is the first major release in more than two years.

gated snare sound without compromising your Daft Punk bass drum hit, and then record them onto separate tracks in Ardour if needed. Whether you dive into the details or simply load some example songs and press play, Hydrogen is great fun and instantly addictive, just like those early pieces of hardware and software.

Project Website
<http://www.hydrogen-music.org>

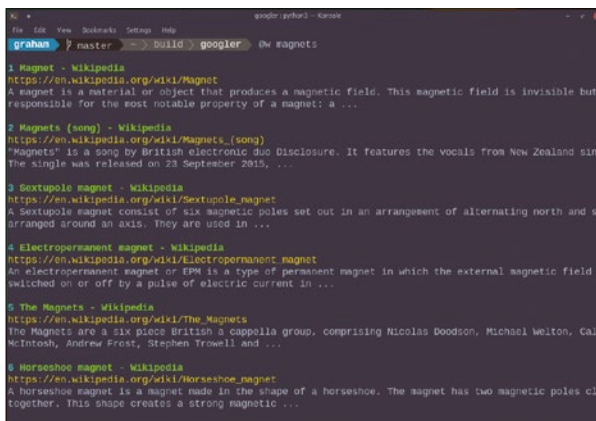
CLI Internet search

Googler 3

Terminal-based web browsers, such as Lynx, are great for testing sites or for accessing and downloading information when you don’t have access to a graphical display. But there isn’t (yet?) a web browser for the command line that makes you want to ditch Chromium or Firefox. Googler, however, is a great step in the right direction. It’s a command-line interface for searching Google and delivering results directly into the terminal. This is brilliant for so many reasons. Often, you only need the search results, and these appear almost instantly unencumbered by advertising. You can then move between the results using the cursor keys, and pressing return with this latest version of Googler will open that link in your default web browser. You can even

set the top result to automatically open without a keypress, which may be useful for serendipity or scripting purposes. You can make advanced specific searches, too, based on age or result location, for example, or even file type if you want to search for music files or animated GIFs. Google’s web interface already does all this, of course, but it’s quicker and easier from the command line and you’re less likely to be sidetracked by YouTube.

The best thing about this new release is that it includes a file that can be “sourced” to set up a huge list of shortcuts to popular domain names. These are used to transparently trigger Googler with parameters configured to search a specific domain. After typing `source googler_at`, for example,



It’s Google, but not as you know it. And Googler is not affiliated with Google in any way.

you can then type `@w magnets` to search Wikipedia for articles on magnets. The results appear just as if you’ve used Googler natively. The `googler_at` file currently contains hundreds of sites, and you quickly get used to these super-short search terms to find what you’re looking for without distracting yourself by opening a graphical browser with its many temptations.

Project Website
<https://github.com/jarun/googler>

Shoot things

Project: Starfighter 1.7

Fight across four planetary systems in 26 levels of shoot-em-up mayhem. If there's one gaming genre that polygons just can't kill, it's the 2D scrolling shooter. There's even a retro-themed sub-genre dedicated to their survival, and the popularity of the Uridium-like reboot, *Hyper Sentinel*, proves there's a lot of shared nostalgia for quick reflexes, chiptunes, and lasers. Project: Starfighter is one such game. It's name even seems to reference one of the first films to use CGI and feature an arcade game – *The Last Starfighter*, 1984 – and the game itself feels very much like a game of that era. The story has you fighting the WEAPCO mega corporation after stealing a ship from the company. The background scrolls in all four directions just like *Time Pilot* – up, down, left and right, rather

than just horizontally, and you need to use your Ctrl-controlled lasers to take out things that attack you. You have energy, rather than dying instantly, and this adds more strategy to the play style. Your enemies are configured similarly. It's great fun and has the typical intensity of a game like this.

The game was originally developed in 2003 and wasn't even open source originally. It was only in 2015 when onpon4 – also known for *Pacewar* and *ReTux* – picked up the game and released it as GPL, making it freely distributable. To do this, all the original graphics had to be replaced due to licensing issues, and while the replacement images work perfectly, they lack the pixelated charm of some other retro shooters. This 1.7 release is a major update that adds difficulty levels, better camera handling, and



Fight across four planetary systems in 26 levels of shoot-em-up mayhem.

enemy balance. And regardless of the graphics, the gameplay is well tuned after all these years, and it's definitely addictive. The excellent music and sound are also worth a listen, and because the entire project is now open source (thanks onpon4!), they'd make an excellent foundation for new levels if you wanted to get into gaming.

Project Website
<http://starfighter.nongnu.org>

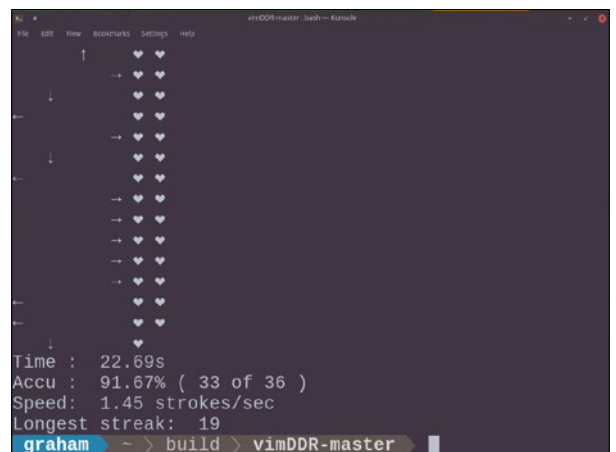
Vim trainer

vimDDR

Back in the mid-1990s, I had an office job that involved scanning lots of barcodes and entering the details from thousands of "assets" into a database. The database front end was nothing more than a simple DOS script, and I spent many hours, days, and weeks tapping away and getting nowhere. As a distraction that might also teach me how to touch type, I wrote a small game in the QBASIC version that came bundled with the PC. This game simply timed how long it took me to type all the letters on the keyboard in a specific random order. The best thing about this game was that while I was playing it, I appeared to be working amazingly productively! All my co-workers were in awe of the rapid keyboard sounds that came from my cubi-

cle, thinking I was rattling through the asset database at an unrivaled rate. In reality, I was trying to beat my high score.

VimDDR reminds me very much of that early QBASIC game I wrote because you can't really get a simpler game. Its 118 lines of Python wouldn't even look out of place printed in a 1980s computer magazine, ready for game-starved readers to type in the themselves. Run the script and a line appears showing a single unicode arrow and three hearts. The hearts are your lives, and the arrow indicates which vim direction key you need



118 lines of Python can be surprisingly useful and addictive.

to press. This is real vim, which means you're not allowed to use the cursor keys! It's a classic finger trainer for h, j, k, and l – the gateway keys to vim's heady greatness. Master these instead of those cheap arrow keys, and you'll position yourself for vim's many and varied editing shortcuts. And it works!

Project Website
<https://github.com/blitzkraft/vimddr>

Master these instead of those cheap arrow keys, and you'll position yourself for vim's many and varied editing shortcuts.

First Steps in Server Security

Fear not the barbarians of cyberspace, and follow our guide to shoring up your digital defenses.

BY BEN EVERARD

So, you just got your first server. Maybe it's a VPS you're renting, or maybe it's running off your home Internet connection. The main point is that it's connected directly to a public IP address, so you can access it from anywhere. That means you can share things with other people or access your data on the go, but it also means that you've entered the domain of hackers, distributed denial-of-service (DDoS) attackers, and other nefarious folk who roam the Internet looking for weak links in security. You need to batten down the virtual hatches and bar the digital door: You're now responsible for securing your own domain. Let's look at the first things you need to do to ensure you stay safe (see "Picking a Distro" for more information).

If you're renting your server, then the chances are that you're first interaction with it is via Secure Shell (SSH). This protocol allows you to create an encrypted shell session to your server and generally use it as though it were a local server. The power that comes with SSH is the reason it's the most common target for attackers. If you have a server on a public IP address, people will attempt to log in via SSH, and usually this starts happening almost instantly.

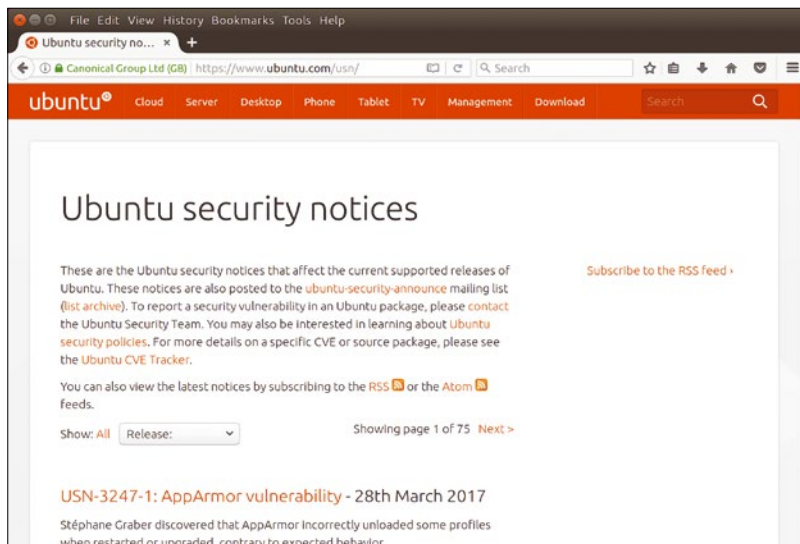
When it comes to server security, SSH is the place to start. The first rule is never, not even temporarily, use a default password to log in with. We're looking

particularly at people using Raspberry Pis, but it goes with anything. If you have to login via password, it should be a randomly generated long string of characters, but we won't delve too far into password rules, because we'll move swiftly on to the second rule of SSH: Disable password login as soon as possible.

Sometimes, when first setting up a server, you have to login via password as it's the only option the server host gives you. However, this should only ever be a temporary solution, because passwords just aren't secure enough to guard this most powerful entry point to your server. As soon as you can, you should shift to disabling passwords and only log in with keys.

When you go to a secure website, your web browser authenticates the website using a certificate and then sets up an encrypted channel to

Figure 1: All good server distros have security announcements as an RSS feed or email list (ideally both). You should keep an eye on this so you know when to update.



Picking a Distro

Generally, when picking a distro for a server, you want to consider the stability and longevity of the distro, particularly with regards to how long you get security updates (Figure 1). If you want to constantly run the latest and greatest software, then you need to pick a distro that supports this, but also be aware that this means that you will have to do more work to update the server as core software versions become unsupported.

On the other hand, if you just want basic software (web server, SQL database server, etc.), then you may wish to plump for a distro that's stable and supported for a long time. It may not always have the latest software, but keeping up to date should mostly be a simple case of grabbing the latest security fixes from the package repository and not having to reconfigure software or reinstall the distro. There's no right answer to this question, but our approach is to choose a stable, long-supported distro (such as CentOS) unless there's a good reason not to (Figure 2).

communicate over. That way, you can be sure that you're really communicating with `www.linux-voice.com` and that no one can listen in on the conversation. All that took place without any passwords being used; instead, cryptographic keys were used to validate the identity of the servers. You can use the exact same cryptographic techniques to log into your new server.

There are a few advantages to using keys rather than passwords to authenticate:

- Multiple different keys can access the same account so that you can let other people access the machine without giving away your secrets (and likewise you can remove their access without having to change passwords).
- You can use the same keys on multiple machines without having to let other people know the passwords to the machines.
- You can be sure that no one breaks your security by setting up a weak password.
- Keys are easier because you don't have to remember (or even type) a password.

Setting up keys is easy. First, you need to generate a public/private key pair. The public key is what the server knows and uses to validate your identity. Any machine with the private key can authenticate against the public key and log in to the server, so guard the private key. The command to generate a key pair is:

```
ssh-keygen -t rsa
```

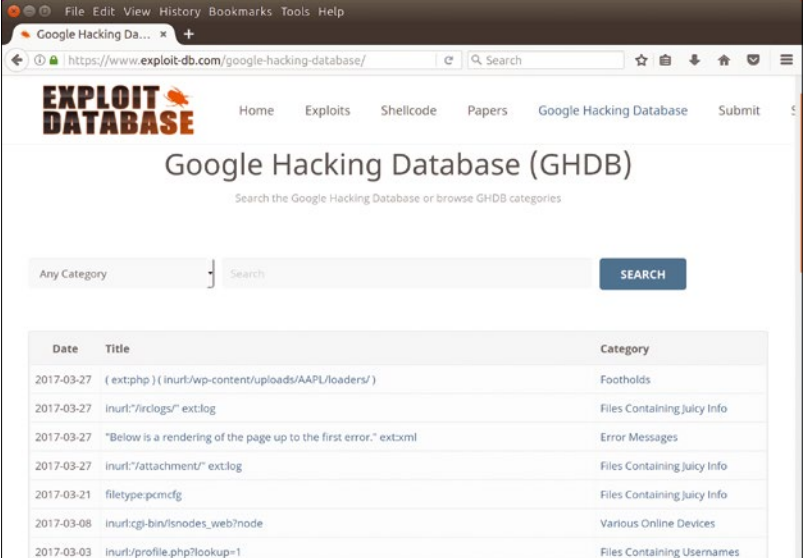
You'll get a few options: You can just accept the defaults for where to store the key (in your `~/.ssh` folder), and if you want, you can add a password to protect the key even if the file itself is compromised.

You then need to copy the public key onto the remote server. This can be as simple as copying the public key from `~/.ssh/id_rsa.pub` (or wherever you told the previous command to save the files) to the `~/.ssh/authorized_keys` file of the user you want to log in as on the remote host. It's best not to overwrite the `authorized_keys` file but to copy the data from the public key to the end of the `authorized_keys` file. Once this is done, you should be able to `ssh` in without using a password.

Once you've verified that you can log in without a password, you can then block all passworded SSH logins. This is done by editing `/etc/ssh/sshd_config` and ensuring that the following lines are in there (without a preceding `#`):

```
ChallengeResponseAuthentication no
PasswordAuthentication no
UsePAM no
```

Once you've put these in, restart your SSH server with:



Date	Title	Category
2017-03-27	{ ext:php } { inurl:/wp-content/uploads/AAPL/loaders/ }	Footholds
2017-03-27	inurl:"/irclogs/" ext:log	Files Containing Juicy Info
2017-03-27	"Below is a rendering of the page up to the first error." ext:xml	Error Messages
2017-03-27	inurl:"/attachment/" ext:log	Files Containing Juicy Info
2017-03-21	filetype:pcmcfg	Files Containing Juicy Info
2017-03-08	inurl:/cgi-bin/snodes_web/node	Various Online Devices
2017-03-03	inurl:/profile.php?lookup=1	Files Containing Usernames

Figure 2: Be careful what you put on your server, because Google's always watching. The Google Hacking Database [1] is a list of searches that find improperly secured servers.

```
Service sshd restart
```

Now you've blocked SSH passwords, which is one of the biggest security vulnerabilities you can have.

Ports of Entry

Your computer allows connections from the outside world via numbered ports. You bind software on your machine to a port number; then anything trying to connect to that port number on that ma-

Move SSH Port

Since SSH is the most popular target for attackers, some people advocate moving it from the default port (22) onto another port. (Typically one numbered above 1000). The advantage of this is that anyone scanning for open SSH ports (which is typically how attackers find machines to target) will simply assume that you don't have SSH running and move along. Anyone actually targeting your machine will probably still find the SSH port.

Another way of adding a bit more security is using Fail2ban. This is a service that watches your log files looking for IP addresses that repeatedly try to log in with invalid credentials and blocks them.

In our opinion, neither using a different SSH port nor using Fail2ban is necessary if you're using SSH keys rather than passwords, and if you're not, neither provides a sufficient level of security. So, for personal servers, we're not convinced that they're needed. However, if you're protecting a high-value server, you may wish to consider additional security measures. In that case, you should be looking for more specific security advice and not a basic guide to securing your first server.

chine gets forwarded to the appropriate bit of software. In this context, ports are virtual and don't exist in the real world: They have nothing to do with the physical connections on the back of your server.

For example, we've already looked at SSH. By default, the SSH software on your server listens on port 22, and your SSH client automatically tries to connect to port 22 (see the "Move SSH Port" box for more information). The two link up and everything works. Understanding what is listening on which ports is key to understanding the security of your machine. Every bit of software listening on a port is a thing that an attacker could have access to.

You can see all the network connections on your machine with the following command (Figure 3):

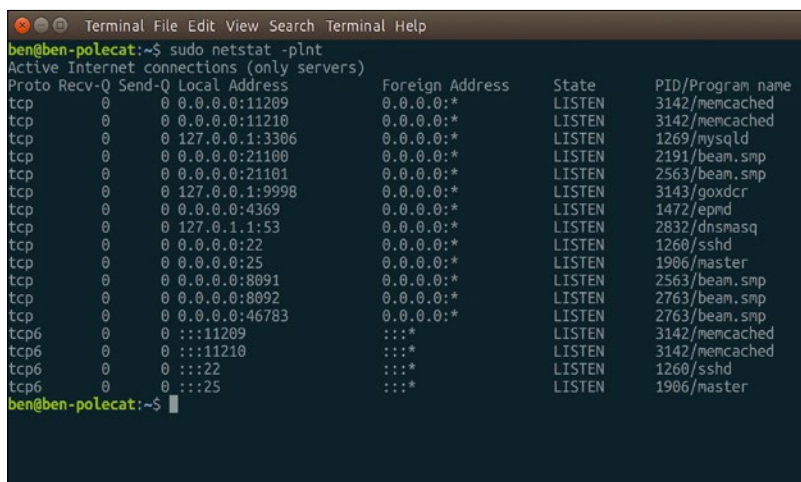
```
netstat -plnt
```

You'll need to run this command as sudo to get the full output. The two key columns here are Local Address and PID/Program Name. The Local Address tells you which IP/port combination the software is listening on and the PID/Program name tells you what it is.

The Local Address column contains two bits of information: the IP address and the port number. The IP address could be localhost (127.0.0.1), it could be all IP addresses (0.0.0.0), or it could be a specific IP address assigned to the machine. The software is only available to machines that can reach that IP address, so anything listening on 127.0.0.1 can only be reached by from the server itself so isn't a significant security consideration. On the other hand, anything listening on 0.0.0.0 is accessible to everyone.

At this point, you need to decide what it is you want to be accessible. More services mean more functionality, but also more things for attackers to target. There's no right answer here,

Figure 3: Netstat is one of the most important pieces of security software, because it allows you to keep track of what software is on what port.



but it really comes down to deciding how much you want to run on your server versus how much time you have to keep your machine secure.

Before we look too closely at this, we need to look at another piece of the network security puzzle: iptables. When a packet of data arrives at your machine, the kernel evaluates it against a set of rules before deciding whether to let it in. These rules are usually created by a bit of software called iptables. You can see the currently running iptables rules with the command:

```
iptables -nL
```

If you've just set up the machine, then this list could be empty, or it could have some default rules created by the OS. Iptables gives you quite fine-grained control over what external IP addresses should be able to access which ports on your machine.

There are whole books written on iptables, and we can only cover the basics here. By default, you have three lists of rules: one for input, one for output, and one for forwarding. For a basic server setup, you can ignore the output and forwarding lists and only deal with the rule on packets coming into the server. Iptables is configured using the file `/etc/sysconfig/iptables`. But it can be easier to make changes to the running iptables interface and then save those changes to disk.

There are two ways of adding rules to Iptables, via insert and append. Insert places a rule at a particular point in the list, whereas append adds it to the end.

You'll want to define rules to set up, that is, which IP addresses can access what. Exactly what goes here depends on what you want to be possible. If you want port 80 to be available to every IP address, you could add this rule by inserting it at the first position in the list with:

```
iptables -I INPUT 1 --dport 80 -j ACCEPT
```

If you want a port to be available to a specific IP address (e.g., one assigned to another server you run), in this case 10.0.0.1, you could add a rule such as:

```
iptables -I INPUT 1 -s 10.0.0.1 --dport 80 -j ACCEPT
```

If you omit the `--dport 80`, then that IP address will have access to all the ports on the machine.

If you want a port to be available to a specific range of IP addresses (e.g., just the local network on 192.168.0.1 to 192.168.0.255), you could use a rule such as this:

```
iptables -I INPUT 1 -m iprange ↗
--src-range 192.168.0.0-192.168.0.255 ↗
-j ACCEPT
```

Once you've defined all the rules you want, you need to end with the following rule, which tells iptables to drop all the packets that don't match one of these rules. This time, we'll use the append option so that this rule is at the end of the list. Because rules are evaluated sequentially, anything after this rule will never be reached.

```
iptables -A INPUT -j DROP
```

If you're connecting to the server via SSH, don't forget to make sure the SSH port (typically 22) is opened in the iptables config before adding the above rule. All of these rules will be applied as you type them in, but unless you use the `save` command:

```
Service iptables save
```

they will be lost when you restart the machine.

Proxy Moxy

So far, we've locked down our machine by SSH and by the ports that are open. However, much of what we want to install on our servers these days comes with a web interface. Sometimes this web interface is the whole of the app, other times it's just an admin panel. Not all of the web interfaces have good security built in, so let's look at how to put a layer of protection in front of a web service with a reverse proxy.

There are a few ways of doing this, but the easiest way is with NGINX (pronounced engine-X). Essentially, this just listens on a port (typically 80) and then forwards any web requests onto other software (which you'll have to configure to listen on other ports). You can configure it to perform authentication before passing any request on, or to apply HTTPS encryption to the connection between the web browser and the server. So, before you can deploy NGINX, you have to consider what ports you want to use for what.

Typically, you should use port 80 for NGINX itself, but other software you're using might already be configured to use this port, so first use `netstat -pInT` to see what (if anything) is on port 80, and reconfigure it to use a different port (this should be straightforward and highlighted in the documentation or the config file). Once you've done this, install NGINX through your package manager. You'll also need a tool for generating password files (`htpasswd`). This is usually in the Apache tools package, which is called `apache2-utils` in Debian and Ubuntu or `httpd-tools` in Red Hat, CentOS, or Scientific Linux.

Listing 1: Add a New Server Block

```
01 server {
02     listen 0.0.0.0:80;
03
04     location / {
05         proxy_pass http://localhost: 8080/;
06         auth_basic "Restricted";
07         auth_basic_user_file /etc/nginx/.htpasswd;
08     }
09 }
```

`htpasswd` creates a password file that just contains one or more username/password hash pairs. If a user enters a password that matches the hash, they are allowed to view the site; if not, they're denied. To configure this with NGINX, first create the username/password pair with:

```
htpasswd -c /etc/nginx/.htpasswd ↗
<username>
```

You then need to configure NGINX to reverse proxy the other web page using this authentication. Essentially, what you're doing here is linking one port with another. In the file `/etc/nginx/nginx.conf`, you should see various sections that define different parts of the way NGINX works. There are many ways to fine-tune this server if you need to handle huge numbers of requests, but we don't need that. You just need to add a new server block to the configuration file with something like what is shown in Listing 1 (which links port 80 to port 8080).

Once you've saved the file, restart NGINX with:

```
Service nginx restart
```

And you should now have a password protected version of the software running on port 8080 on port 80. Don't forget to block port 8080 in iptables to everything other than localhost; otherwise, this hasn't added any security.

Security doesn't have to be hard, but you do have to pay attention to it. By just keeping an eye on what software is listening on which ports, you should stay out of trouble online. Always remember to keep your software up to date, especially any software that runs on a public port. Do this and you can benefit from all the advantages of an Internet-facing server without having security problems. ■■■

Info

[1] Google Hacking Database: <https://www.exploit-db.com/google-hacking-database/>

TkInter – Graphical Python Apps in Minutes

Expand your Python knowledge and write GUI apps with a smattering of code, thanks to the TkInter toolkit.

BY MIKE SAUNDERS

We Linux Voicers have, over the years, argued for more consistency on the Linux desktop. We appreciate that choice and diversity is good, and there are reasons why the Gtk and Qt graphical toolkits exist. Indeed, competition between them can be a good thing. But the downside to this – especially if you’re running a mixture of Gtk and Qt apps on your desktop – is inconsistency in the user experience. It’s not a big deal in the grand scheme of things, and apps using different toolkits can still get along nicely, but sometimes you notice some rough edges.

Other Toolkit Options

As we mentioned, TkInter is just one option for creating graphical apps in Python. It has the benefit of being readily available on pretty much every operating system that runs Python, but it’s not the most feature-rich toolkit. If you want to write more advanced apps, there are some options worth considering (although you’ll have to pore through quite a bit of documentation).

PyQt [1] is one of the best-known Python toolkits and is available for Python 2 and Python 3. It’s a well-maintained project with extensive documentation, and a commercial version with extra features and support options is available as well.

If you’re more of a Gtk/Gnome fan, then check out PyGObject [2]. This module gives you access to oodles of Gnome features and functions and has become the successor project to the now-dormant PyGtk.

Then there’s Kivy [3] for developing multi-touch apps that can run on iOS and Android, and many more. The Python wiki has a detailed list of toolkits [4], showing which platforms they run on, and the most recent update (so you can quickly tell if a toolkit is being actively developed or may be suffering from bitrot).

With this in mind, we’d normally be reticent to bring another toolkit into the mix (see the “Other Toolkit Options” box). But that’s what we’re going to do in this tutorial, with TkInter. This lets you create graphical applications in Python – providing you with buttons, menus, text entry boxes, and other widgets. But, why are we devoting these pages to TkInter, when Python interfaces to Gtk and Qt already exist?

Well, TkInter has some significant advantages. First, it’s Python’s de facto GUI toolkit and is usually installed by default alongside the language. Whereas PyQt and PyGtk can be quite fiddly to install – especially on other platforms like Windows and Mac OS – TkInter is always there. You can write cross-platform apps and only require your end users to install Python. Second, it’s quite easy to use. If you have a command-line Python script and want to add a graphical layer on top of it, TkInter lets you do this with relatively few lines of code.

TkInter isn’t the prettiest toolkit, nor is it absolutely loaded with features. But it’s well worth learning for the reasons mentioned above, so we’ll show you how to get started with it and work with the various widgets it provides. Whether you’re a Python guru or just getting started with the language, you’ll learn some valuable skills for the future. In the immortal words of Mario, let’s-a go!

TkInter Basics

Many distros install TkInter alongside Python by default, but you may need to install it separately. If you see an error message when running the following code, search your package manager for *python3-tk* or *python3-tkinter* and install that package.

Here’s our first TkInter program – save this code into a plain text file called `test.py` (Listing 1) and run it in Python 3 (e.g., `python3 test.py`).

Let’s go through this step by step. On the first line, we tell Python to import the TkInter module, and specifically all of the methods and features that it provides. (If you’re quite new to Python, you don’t need to be concerned with the specifics here – just think of this line as making TkInter available to the rest of your program).

Listing 1: test.py

```
01 from tkinter import *
02
03 root_win = Tk()
04
05 my_label = Label(root_win, text =
    "Hello, world!")
06 my_label.pack()
07
08 root_win.mainloop()
```

On the next line, we create our first widget. In TkInter, as indeed with most toolkits, widgets are simply GUI elements – buttons, checkboxes, sliders, text entry boxes, and so forth. Even windows are widgets. So, on this line of code, we tell TkInter to create a new “root” window widget, which we call `root_win`. The root window is like the root directory in your Linux installation – it’s the base of everything else. So, we need to create a root window before we can do anything else.

Now, a root window on its own isn’t much use, so let’s add something to it: a text label. This is a non-interactive widget that simply displays some text inside the window. To achieve this, we use the `Label()` method from TkInter with two parameters: the window in which it should be placed, and the text to be displayed. You can see in our program that we chose to place the label inside `root_win` and use “Hello, world!” as the text.

So far so good – but now we need to tell TkInter to fit or “pack” our newly created label inside the root window. That’s what we do with `my_label.pack()` – we don’t need any other parameters here.

Finally, we start TkInter’s main loop, which displays all of the widgets we’ve created and processes events relating to them. Because our program isn’t interactive, the main loop doesn’t

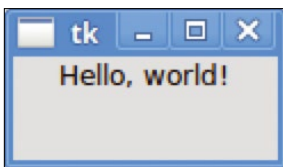


Figure 1: Our first program – it doesn’t do much, but it shows that TkInter is installed and working correctly.

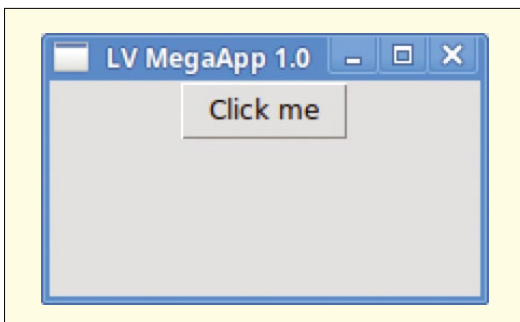


Figure 2: Now we’re going interactive! Clicking the button generates text in the terminal from which the app was launched.

Listing 2: Additional Features

```
01 from tkinter import *
02
03 def do_click():
04     print("Clicked")
05
06 root_win = Tk()
07 root_win.title("LV MegaApp 1.0")
08 root_win.geometry(200x100)
09
10 my_button = Button(root_win, text = "Click me", command = do_click)
11 my_button.pack()
12
13 root_win.mainloop()
```

do much here – it simply shows the window (like in Figure 1) and waits for the user to close it (via the window manager).

Adding Buttons and Callbacks

Let’s now make an interactive program with some additional features (Listing 2). In the previous example, the window was given a generic title (“tk”) and automatically sized to fit the label we created. However, we have control of these things, so let’s customize them and add a button to perform an action.

In this program, we import TkInter like we did previously, but then we define a function called `do_click`. All this does is print a “Clicked” message in the terminal, but later in the code we’re going to connect this function with a button. In other words, when the user clicks a button, “Clicked” is displayed in the terminal window.

After the function, we set up a root window as usual – but this time we perform two extra operations on it. First of all, we give the window a custom title, and then we set the window’s size in pixels: 200 wide by 100 high. Note that the user can still resize the window using his/her window manager, but we’ve provided its default dimensions.

Next up, we use TkInter’s aptly named “Button” method to create a clickable button. This takes three parameters: the window in which it should be placed, the text that is displayed on the button, and the command that should be executed when the button is clicked. In our case, we tell TkInter to run the `do_click()` function we created earlier.

Finally, we pack that button into the root window and start the main loop – handing control over to TkInter. Et voilà, you’ll see a window like in Figure 2. Try clicking the button a few times, and watch the output in the terminal where you launched your program. This is still a very simple app, but we now know how to make programs interactive. Not bad for just a few lines of code!

A Complete Program

You can do quite a bit with buttons alone, especially if your Tkinter app is just a front end to a command-line script, but there are many other widgets you can incorporate into your programs as well. For the next program, we'll create a graphical tool for calculating the area of a circle based on its radius (using pi). For this, we'll need a text entry box in which the user can type the radius, along with a button to trigger the calculation. Then we'll need a text label that's updated with the newly calculated area. As an extra bonus, we'll throw in a menu as well.

Listing 3 shows the code – have a look and see if you can work out what's going on. Then, we'll examine it in detail.

Some parts of this will be familiar to you, but we've introduced a bunch of new things as well. Along with the Tkinter module, we also import the `sys` and `math` modules as we'll need them later. Next, we define a couple of functions that will be linked to menu items. The first one, `about_dialog`, simply pops up a Tkinter message box with two parameters: the box title, and the text to be displayed. Note that in the second parameter, you can include newline (`\n`) characters if you want to display a larger amount of text.

Our second function, `exit_app`, will be called when the user clicks Exit in the menu. This simply calls the `exit` function in Python's `sys` module with a return code of zero – that is, the program exited correctly. Following this, we have a third function that does the work of calculating the area of a circle based on its radius; we'll come back to this in a moment.

Then, the main code begins: We create a root window with a title and default horizontal and vertical size, as we did previously. Next, we create a main menu and attach it to the root window – but by default it's blank. To add a specific menu, like File, we have to incorporate it into the main menu, and then we can add individual menu items. This is exactly what we do in the last two lines of this code block: We create About and Exit items under the File menu, connecting them to the two functions we defined early in the program.

Next, we have a chunk of code beginning with this: `radius = StringVar()`. Essentially, we need to use two variables in our program, one for the radius (which the user can modify) and one for the area (which is calculated from the radius). Now, we could use vanilla Python variables for this, but we'd stumble across a problem: When the variables change, like when the user enters a new radius, the program won't update automatically. Tkinter doesn't track these variables and update the display accordingly.

Therefore, we need to use Tkinter's own `StringVar` routine to create special string variables that can be used in widgets – and those widgets are updated on the screen when the values in them change. So, we create a Tkinter string variable called `radius`, setting it to zero, and another called `area`.

Organizing your Widgets

Then the fun part starts: adding widgets to the window. This time we append `.grid()` onto each

Listing 3: Create a Graphical Tool

```
01 import math, sys
02 from tkinter import *
03 from tkinter import messagebox
04
05 def about_dialog():
06     messagebox.showinfo("About", "Version 1.0")
07
08 def exit_app():
09     sys.exit(0)
10
11 def calc_area(*args):
12     area_result = (float(radius.get()) ** 2) * math.pi
13     area.set(round(area_result, 2))
14
15 root_win = Tk()
16 root_win.title("Area calculator")
17 root_win.geometry("200x100")
18
19 main_menu = Menu(root_win)
20 root_win.config(menu = main_menu)
21 file_menu = Menu(main_menu)
22 main_menu.add_cascade(label="File", menu = file_menu)
23 file_menu.add_command(label="About", command = about_dialog)
24 file_menu.add_command(label="Exit", command = exit_app)
25
26 radius = StringVar()
27 radius.set("0")
28 area = StringVar()
29
30 area_label = Label(root_win, text = "Area:").grid(column =
31     1, row = 1)
32 area_value = Label(root_win, textvariable = area).
33     grid(column = 2, row = 1)
34 radius_label = Label(root_win, text = "Radius:").grid(column
35     = 1, row = 2)
36 radius_entry = Entry(root_win, width = 7, textvariable =
37     radius).grid(column = 2, row = 2)
38 calc_button = Button(root_win, text="Calculate", command =
39     calc_area).grid(column = 2, row = 3)
40 root_win.mainloop()
```

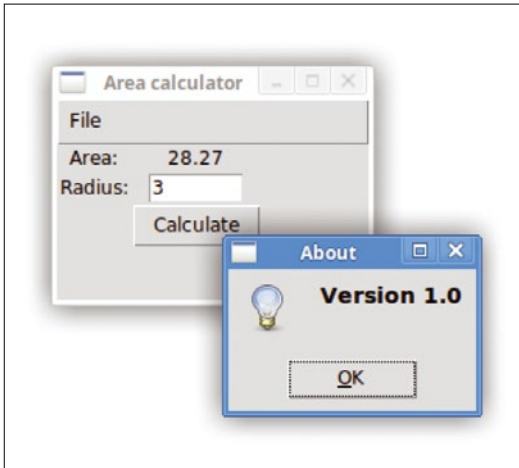


Figure 3: Here's a fully working app with a text entry box, labels, buttons, and menu options.

widget creation, specifying the row and column inside the root window where the widget should appear. This gives us some basic control of the layout inside the window (think of it like a grid) and also gives us the benefit that we don't need to "pack" each widget after creating it – that happens automatically when we specify the position.

So, the first widget we create is a simple static label that says *Area:*. The next label, *area_value*, displays the contents of the special Tkinter *area* string variable we declared earlier. Underneath these two labels, we have another static label for *Radius:*, and then we put an *Entry* (text entry) widget alongside this, to the right. We set this to be seven characters in width and tell Tkinter that the value entered by the user should be stored in the *radius* string variable we created beforehand.

Finally, underneath all of this – on row three – we add a *Calculate* button, linking it to the *calc_area* function we defined early in the program. If you go back and look at that function now, it should be clear how it works. Using a temporary variable called *area_result*, we calculate the area from the radius. Then, on the second line of the function, we use our Tkinter *area* variable's *set* method to update it – providing it with the contents of *area_result*, rounded down to two decimal points (just to make the app look tidier). We need to use this *set* method to update the Tkinter string variable; we can't just change it like a normal variable.

The final line in the program runs the main loop as usual. Try it out – it should look like Figure 3. Enter a radius, click *Calculate*, and see the result. Then try the options in the File menu that we created. Once you're familiar with how the program works, try experimenting with the layout – moving the position of the *Calculate* button, rearranging the labels, and so forth. You can also play around with the menu items, adding another submenu and calls to other functions.

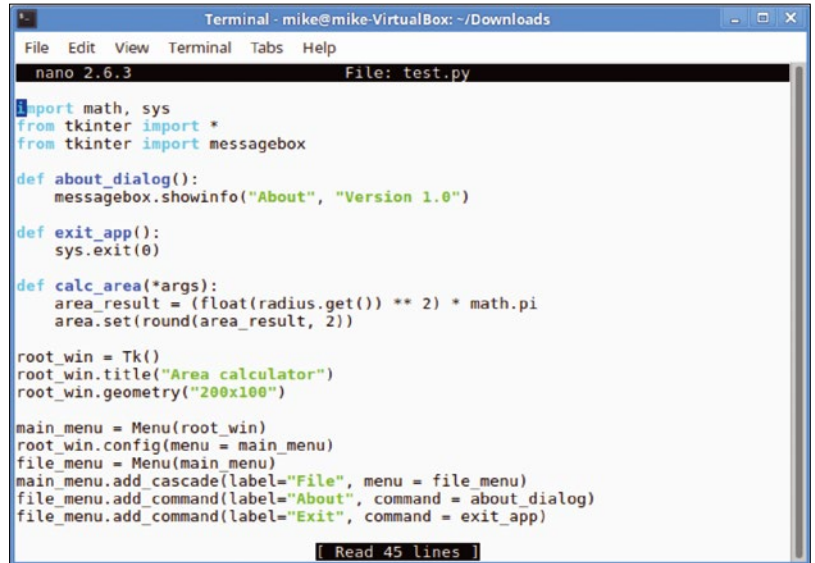


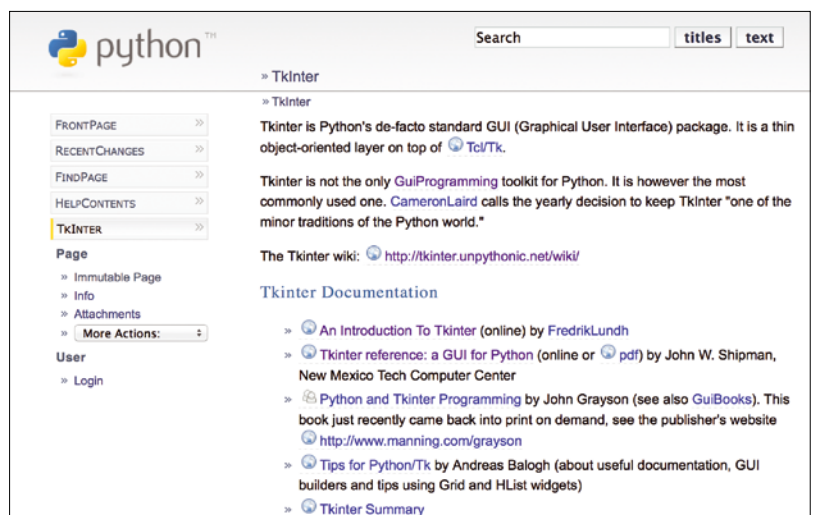
Figure 4: Syntax highlighting makes it much easier to work with code. Even Nano can highlight Python.

With the widgets we've used in this app, you can create some rather complex applications – but there's even more to explore. You can pop up file load and save dialogs, draw graphics on a canvas, and do much more as well (Figure 4). When you're fully familiar with all the code in our area calculator app, you can explore further features of Tkinter on the wiki (Figure 5) [5]. There you'll find links to reference guides and other tutorials. Happy hacking, and if you create an awesome app, let us know all about it! ■■■

Info

- [1] PyQt: <https://riverbankcomputing.com/software/pyqt/intro>
- [2] PyGObject: <https://wiki.gnome.org/action/show/Projects/PyGObject>
- [3] Kivy: <https://kivy.org>
- [4] List of toolkits: <https://wiki.python.org/moin/GuiProgramming>
- [5] Python wiki: <https://wiki.python.org/moin/Tkinter>

Figure 5: The Python wiki has heaps of resources for learning Tkinter in more detail.



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.

OSDC Berlin

Date: May 16–18, 2017

Location: Berlin, Germany

Website: <https://www.netways.de/events/osdc/overview/>

OSDC offers the opportunity to meet open source professionals and insiders and gather and share expertise over three days of presentations, hands-on workshops, and social networking. OSDC aims to “simplify complex IT infrastructures with open source” for experienced administrators and architects.

ISC High Performance

Date: June 18–22, 2017

Location: Frankfurt, Germany

Website: <http://isc-hpc.com/>

The 32nd conference in the ISC High Performance series focuses on HPC technological development, its application in scientific fields, and its adoption in commercial environments. The conference offers 12 HPC topics grouped under three categories: systems, applications, and emerging topics. You can also look forward to ISC tutorials, workshops, and the exhibition.

SUSECON 2017

Date: September 25–29, 2017

Location: Prague, Czech Republic

Website: <http://www.susecon.com>

SUSECON 2017 features exceptional technical content presented by SUSE employees, customers, partners, and community enthusiasts. SUSECON focuses on helping you find ways to build and define your future to provide a flexible and efficient infrastructure. The dynamic and entertaining keynote speeches inspire, inform, and invigorate.

EVENTS

Open Source Data Center Conference	May 16–18	Berlin, Germany	https://www.netways.de/events/osdc/
DevConf Panamá 2017	May 25–26	Panama City, Panama	https://www.devconfpanama.com/#/
openSUSE Conference OSDC 2017	May 26–28	Nuremberg, Germany	https://events.opensuse.org/conference/oSC17/
PyConWEB 2017	May 27–28	Munich, Germany	https://pyconweb.com/
ISC High Performance (ISC 2017)	June 18–22	Frankfurt, Germany	http://www.isc-hpc.com/
Deutsche Openstack Tage	June 26–28	Munich, Germany	https://openstack-tage.de/
AnDevCon	July 17–19	Washington, DC	http://www.andevcon.com/
InterDrone	September 6–8	Las Vegas, Nevada	http://www.interdrone.com/
Storage Developer Conference (SDC)	September 11–14	Santa Clara, California	http://www.snia.org/events/storage-developer
SUSECON 2017	September 25–29	Prague, Czech Republic	http://www.susecon.com/
SC17	November 12–17	Denver, Colorado	http://sc17.supercomputing.org/
Linux Presentation Day 2017.2	November 18	Europe-wide in numerous cities	http://www.linux-presentation-day.org/

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.



AUTHORS

Erik Bärwaldt	24, 26
Swapnil Bhartiya	8
Chris Binnie	42
Mario Blättermann	20, 58
Zack Brown	12
Bruce Byfield	54, 62
Joe Casad	3
Mark Crutch	65
Nate Drake	38
Ben Everard	65, 74, 88
Andrew Gregory	67
Jon "maddog" Hall	68
Klaus Knopper	34
Charly Kühnast	52
Vincent Mealing	65
Graham Morrison	82
Simon Phipps	66
Mike Saunders	70, 92
Mike Schilli	46
Tim Schürmann	16
Valentine Sinitsyn	76

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.

CONTACT INFO

Editor in Chief

Joe Casad, jcasad@linux-magazine.com

Managing Editor

Rita L Sooby, rsooby@linux-magazine.com

Localization & Translation

Ian Travis

News Editor

Swapnil Bhartiya

Copy Editors

Amber Ankerholz, Amy Pettle

Layout

Dena Friesen, Lori White

Cover Design

Dena Friesen

Cover Image

© Maxim Kazmin, 123RF.com

Advertising – North America

Ann Jesse, ajesse@linuxnewmedia.com
phone +1 785 841 8834

Advertising – Europe

Brian Osborn, bosborn@linuxnewmedia.com
phone +49 89 99 34 11 48

Publisher

Brian Osborn, bosborn@linuxnewmedia.com

Marketing Communications

Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
616 Kentucky St.
Lawrence, KS 66044 USA

Customer Service / Subscription

For USA and Canada:
Email: cs@linuxpromagazine.com
Phone: 1-866-247-2802
(Toll Free from the US and Canada)
Fax: 1-785-856-3084

For all other countries:
Email: subs@linux-magazine.com
Phone: +49 89 99 34 11 67

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 © 2017 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 Germany

Distributed by COMAG Specialist, Tavistock Road, West Drayton, Middlesex, UB7 7QE, United Kingdom
LINUX PRO MAGAZINE (ISSN 1752-9050) is published monthly by Linux New Media USA, LLC, 616 Kentucky St., Lawrence, KS, 66044, USA. Periodicals Postage paid at Lawrence, KS and additional mailing offices. Ride-Along Enclosed. POSTMASTER: Please send address changes to Linux Pro Magazine, 616 Kentucky St., Lawrence, KS 66044, USA.

Published monthly in Europe as Linux Magazine (ISSN 1471-5678) by: Sparkhaus Media GmbH, Zieblandstr. 1, 80799 Munich, Germany.

Issue 200 / July 2017

The Story of the GPL

**Look for our
Linux Voice
Archive DVD!**

Next month, we celebrate our 200th issue with a special look at the powerful GNU Public License (GPL) and how it changed the world.

Approximate	
UK / Europe	Jun 05
USA / Canada	Jun 30
Australia	Jul 31
On Sale Date	



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: www.linux-magazine.com/newsletter

Lead Image © Anton Balazh, 123RF.com

InterDrone

The International Drone Conference and Exposition

Discover the Future – at the World's Largest Commercial Drone Conference & Expo

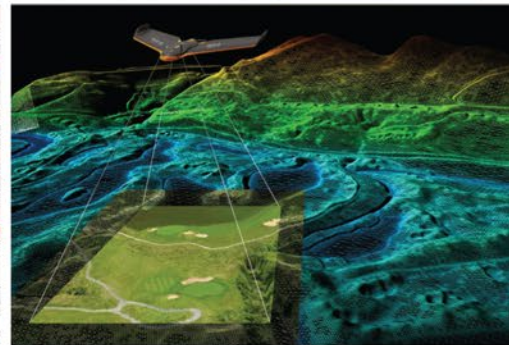


Image credit: Future Aerial Innovations



"If you want to see the state-of-the-art and expand your knowledge about the drone industry, InterDrone is the place to be."

—George Gorrill, Structural Engineer, Thomas Engineering Group

September 6-8, 2017

Las Vegas

www.InterDrone.com

Register Early for the Biggest Discount!





Lightning Fast

All Flash NVMe

S E R V E R S O L U T I O N S



BigTwin™
2U/4 DP Nodes, 6 NVMe/SAS3/SATA3 per Node



1U 10 NVMe Ultra
SYS-1028U-TN10RT+



2U 48 NVMe Simply Double
SSG-2028R-NR48N



2U 24 NVMe Ultra
SYS-2028U-TN24R4T+



2U 40/20 Dual Port NVMe
SSG-2028R-DN2R40L
SSG-2028R-DNR20L



7U 42 NVMe
Datacenter Blade®



- Supports Intel® Xeon® processor E5-2600 v4/v3 product families
- Over 60 SKUs Enabled
- Optimized Hot-Swap NVMe Designs
- Highest Quality
- The Leader in NVMe Server Development



Intel Inside®. Powerful Productivity Outside.

Learn More at www.supermicro.com

© Super Micro Computer, Inc. Specifications subject to change without notice. Intel, the Intel logo, Xeon, and Xeon Inside are trademarks or registered trademarks of Intel Corporation in the US and/or other countries.