

FREE DVD

SystemRescueCd 6.0.3
64 bit

Plant Monitoring
with a Raspberry Pi

IoT on the Cheap
Hack your smart plugs and
make them talk to Linux

LINUX
MAGAZINE



Double-Sided DVD
INSIDE!

LINUX

MAGAZINE

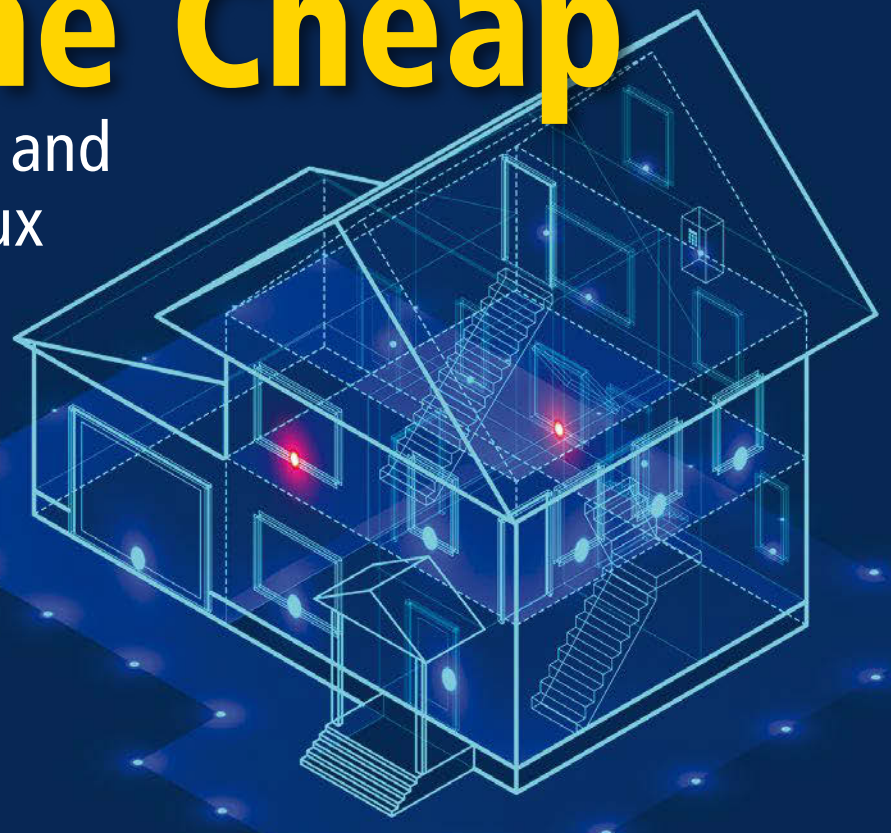
AUGUST 2019

IoT on the Cheap

Hack your smart plugs and
make them talk to Linux

Find Performance
bottlenecks
with eBPF

Karoshi
The easy way to set up a
complex server system



Timer Tools
Schedule commands
and scripts with at,
cron, and anacron

chrony
Sync up with the *other*
time service daemon

Go Tricks
Search for photos
by GPS coordinates

LINUXVOICE

- Count Your Money with GnuCash
- Fanurio Time Management Tool
- maddog on Huawei and 5G



- FOSS Picks**
- KDE Partition Manager
 - Olivia Cloud Music Player
 - DOSBox-X
- Tutorials**
- Scripting: Add User Input
 - Preparing an Object for 3D Printing

Issue 225
Aug 2019
US\$ 15.99
CAN\$ 18.50

7 25274 58049 1

AN EXTREME REFRESH
WITH OCTA-CORE POWER!

DEDICATED ROOT SERVER EX62

High speed performance with the new
Intel Core i9-9900K octa-core processor

ENTERPRISE
HDDs or
NVME SSDs



Dedicated Root Server EX62

- ✓ Intel® Core™ i9-9900K Octa-Core incl. Hyper-Threading-Technology
- ✓ 64 GB DDR4 RAM
- ✓ 2 x 8 TB SATA Enterprise Hard Drive 7200 rpm
- ✓ 100 GB Backup Space
- ✓ Traffic unlimited
- ✓ Location Germany or Finland
- ✓ No minimum contract
- ✓ Setup Fee \$78

monthly from \$ **72.50**

Dedicated Root Server EX62-NVMe

- ✓ Intel® Core™ i9-9900 Octa-Core incl. Hyper-Threading-Technology
- ✓ 64 GB DDR4 RAM
- ✓ 2 x 1 TB NVMe SSD
- ✓ 100 GB Backup Space
- ✓ Traffic unlimited
- ✓ Location Germany or Finland
- ✓ No minimum contract
- ✓ Setup Fee \$78

monthly from \$ **72.50**

BORG REORG

Dear Reader,

Microsoft just announced that its built-in Linux kernel is available for testing. The Linux kernel that will be integrated with Windows 10 is tuned to work with the Windows Subsystem for Linux compatibility suite. Predictably, a new round of alarms went up around the Linux community – just as they did when Microsoft announced that they would soon be shipping this new Linux back in May. Microsoft distributing Linux? Is this a trick or some kind of nefarious subterfuge? Weren't they the ones who said Linux is a cancer?

The company now says they are quite friendly and happy with Linux, but their past behavior was so outrageous that many who remember the bad old days are still a bit spooked by their change of heart.

Microsoft used to prowl the headlines with a very intimidating aura of invincibility. The popular notion was that, whenever they wanted something, they got it. Starting with taking down IBM for dominance in the PC space, through the browser wars, through the office software wars, and on into countless other skirmishes, the Redmond raiders were known for using the power of their market position to overwhelm and destroy opposition, conjuring up comparisons with the Star Trek Borg and their menacing motto "resistance is futile." This image of indomitability was a tool in their arsenal that played quite well with users, venture capitalists, and government regulators and became a sort of self-fulfilling prophecy, which made matters worse.

When they declared Linux was their enemy, it was all very freaky for a while as they played through their various monopolistic power tricks, such as false information, lawsuits, and half-baked "patent indemnity" gambits. Many are wondering if they have truly given up on all the funny business.

But does it even matter anymore? Even if you don't trust that Microsoft has changed, you would have to agree that whatever it is they were trying to do back then doesn't seem to be working very well anymore.

Microsoft's recent record as a monopolist isn't exactly flattering. In fact, the whole reason they are working with Linux now is because their grand design to destroy Linux *totally failed*. Their grand plan to shoehorn themselves into the mobile phone space didn't just fail but *failed spectacularly*.

Microsoft is a player in today's web economy, but they are not even the biggest or most intimidating player. Their Bing search engine is a very distant number two in search

engines with around 10 percent of the market share. Their Azure cloud appears stable but isn't exactly knocking down all the top competitors. (By the way, Azure is populated with lots of Linux systems at this point.)

I'm not sure you could say Microsoft has actually *re-formed* in some cosmic sense, but I think the people who are afraid of them co-opting and subjugating Linux are giving them way too much credit.

Microsoft is not a power player in the Linux space, because, well, they don't want to be – and they won't want to be until they finally give up on selling Windows. And the tools they have used for control in the past (marketing hype, intellectual property threats) are not especially effective in the Linux community.

They'll hang around for a while. Maybe they will fit in and stay, or maybe they will get voted off the island. Somehow I don't see them as the last one standing.

Joe

Joe Casad,
Editor in Chief



LINUX MAGAZINE

WHAT'S INSIDE

This month we show you how to hack a smart plug, reflash the firmware, and control it from a nearby Linux computer. If you love the concept of the Internet of Things but you don't like spending a lot of money on proprietary gadgets, read on for our low-tech high-tech solution. Also in this month's issue:

- **eBPF** – explore the performance and monitoring tools available through the enhanced BPF in-kernel virtual machine (page 30).
- **Karoshi** – an innovative Linux distro with the emphasis on simplifying configuration (page 36).

Look in MakerSpace for a look at the Rasp Pi as a function generator, and check out LinuxVoice for articles on GnuCash and the Fanurio time management tool.

SERVICE

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

NEWS

08 News

- Chromium-Based Browsers Will Ignore Google's Ad-Blocking Ban
- Zorin OS 15 Released
- Ubuntu to Package Proprietary Nvidia Driver
- openSUSE Leap 15.1 Released
- Red Hat Enterprise Linux 8 Available
- Google Brings Linux to Chromebook
- New Class of CPU Flaws Affects Almost Every Intel Processor Since 2011

12 Kernel News

- Cleaning Up Dependencies on Older GCC Versions
- Protecting RAM from Hostile Eyes
- Verifying Return Addresses

IN-DEPTH

22 chrony

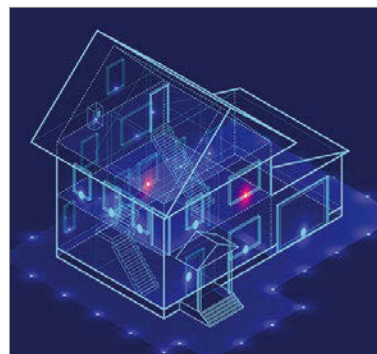
The chrony implementation of the network time protocol provides an alternative to the familiar ntp daemon.



COVER STORIES

14 Controlling a Smart Plug

You could spend hundreds of dollars on specialized IoT appliances and fixtures, or you could just hack a smart plug and talk to it with your Linux system.



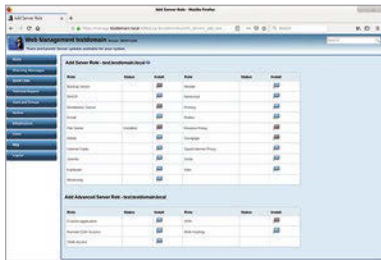
26 Command Line – at, cron, anacron

The at command and the related cron and anacron can help you efficiently schedule tasks, whether one-time events or jobs to be done repeatedly.

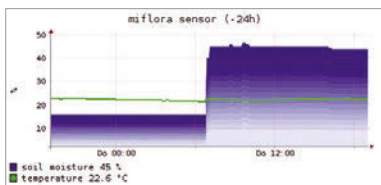


IN-DEPTH

- 30 **eBPF**
Use this in-kernel virtual machine to identify resource bottlenecks and optimize your installation.
- 36 **Karoshi**
Intranets with multiple servers and services require careful configuration. With Karoshi Linux, you can set up even complex structures in no time at all.



- 42 **Charly's Column – Mi Flora**
Columnist Charly Kühnast recently attached Mi Flora humidity sensors to his potted plants. At first, they only transmitted junk on Bluetooth, but armed with the right tools and a Rasp Pi, Charly now reaps a rich harvest of data.



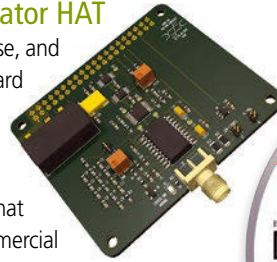
- 44 **Programming Snapshot – Go**
Every photo you take with your mobile phone stores the GPS location in the Exif data. A Go program was let loose on Mike Schilli's photo collection to locate shots taken within an area around a reference image.

- 48 **SystemRescueCd**
The SystemRescueCd live system contains numerous tools that you can use to recover deleted files or a defective system.



MAKERSPACE

- 52 **Function Generator HAT**
A touch display, a case, and a custom add-on board transform the humble Rasp Pi into a high-performance function generator that rivals expensive commercial offerings.
- 58 **Open Hardware – Librem One**
Ahead of the Librem 5 phone release, Purism releases a suite of communications applications called Librem One, including Chat, Mail, and more. But how accessible are these apps for average users?
- 62 **Go on the Rasp Pi**
We show you how to create a Go web app that controls Raspberry Pi I/O.

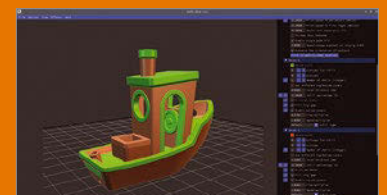


LINUXVOICE

- 65 **Welcome**
This month in Linux Voice.
- 67 **Doghouse – 5G**
The recent uproar over Huawei has more to do with intellectual property and 5G than a fear of malware.
- 68 **GnuCash 3.5**
Put your personal finances in order or manage your small business with GnuCash 3.5.
- 78 **FOSSPicks**
This month we explore Olivia, DIN Is Noise, Kaidan, termshark, Worldview, Snipes, and more!
- 84 **Tutorials – Shell Scripts**
Letting your scripts ask complex questions and give user feedback makes them more effective.
- 90 **Tutorials – 3D Printing**
Paul Brown looks at how to prepare your design for 3D printing.



- 72 **Fanurio Time Management**
If you need to accurately measure time spent on individual projects and tasks, you need something more sophisticated than yellow sticky notes.



On the DVD



**TWO TERRIFIC
DISTROS**
**DOUBLE-SIDED
DVD!**

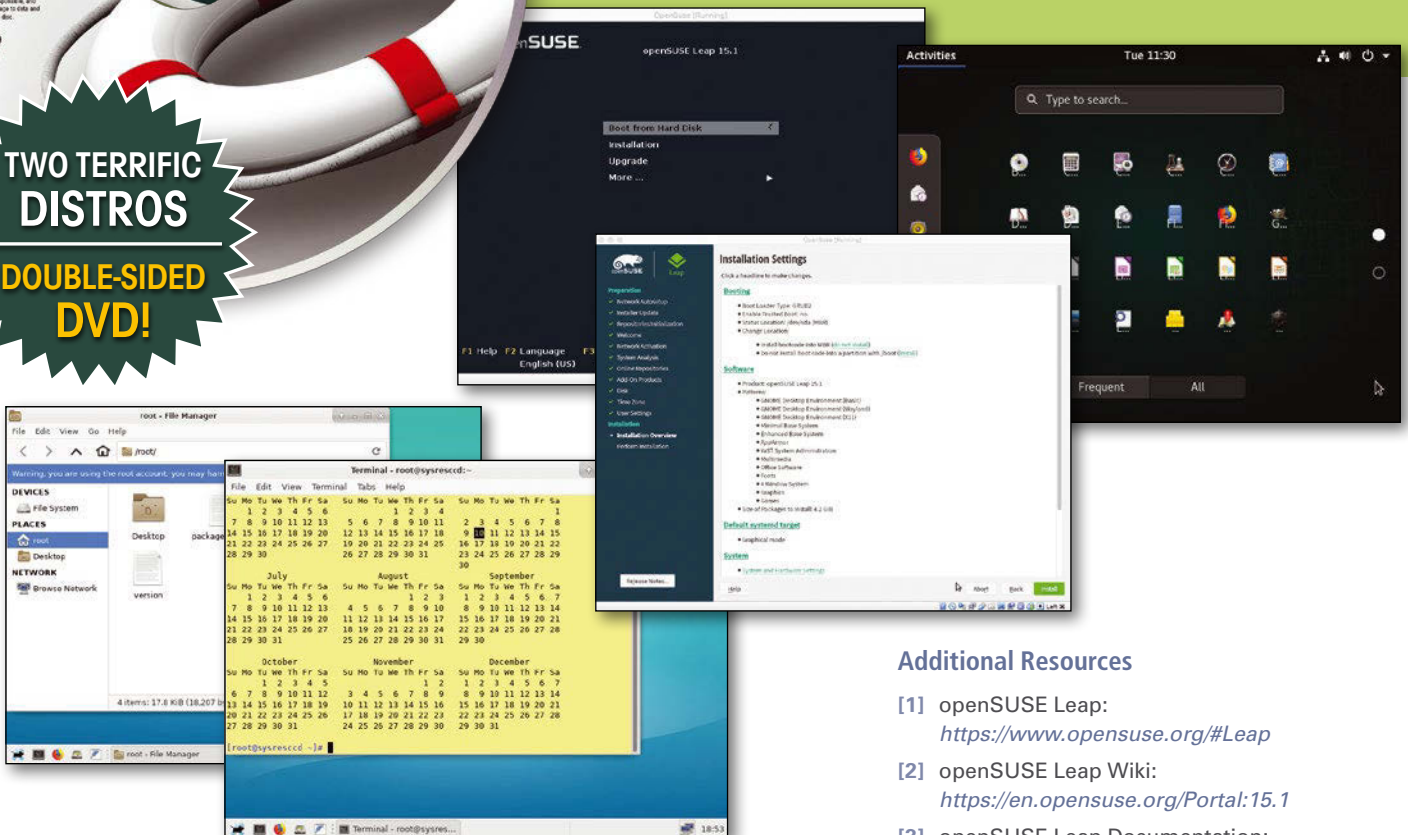
openSUSE Leap 15.1

OpenSUSE is a community-based Linux project sponsored by SUSE. The openSUSE Leap edition is a regular release version with tools and new technologies that will one day appear in SUSE Enterprise Linux.

According to the openSUSE developers, the latest release offers "continuity and stability." OpenSUSE 15.1 comes with a major update to the graphics stack and better support for GPU virtualization. The YaST configuration tool offers an improved partition utility and a new interface for firewall configuration.

SystemRescueCd

SystemRescueCd is a Live Linux edition used for repairing and rescuing computer systems. Along with a selection of powerful repair and diagnostic tools, you'll find tools for creating and editing partitions, as well as a helpful collection of administration and configuration utilities.



Additional Resources

- [1] openSUSE Leap:
<https://www.opensuse.org/#Leap>
- [2] openSUSE Leap Wiki:
<https://en.opensuse.org/Portal:15.1>
- [3] openSUSE Leap Documentation:
<https://doc.opensuse.org/>
- [4] SystemRescueCd:
<http://www.system-rescue-cd.org/>
- [5] SystemRescueCd Manual:
<http://www.system-rescue-cd.org/manual/>

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

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



openSUSE®

Asia Summit 2019

Bali, Indonesia



Information Technology Department
Faculty of Engineering
Udayana University
Bali, Indonesia
October 5—6, 2019



openSUSE 

Embedded System 

Containers 

Open Source Software 

Cloud 

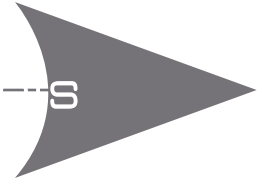
...and more 



<https://events.opensuse.org/conferences/summitasia19>

NEWS

Updates on tech



THIS MONTH'S NEWS

- 08** • Chromium-Based Browsers Will Ignore Google's Ad-Blocking Ban
- Zorin OS 15 Released
- 09** • Ubuntu to Package Proprietary Nvidia Driver
- openSUSE Leap 15.1 Released
- Red Hat Enterprise Linux 8 Available
- More Online
- 10** • Google Brings Linux to Chromebook
- New Class of CPU Flaws Affects Almost Every Intel Processor Since 2011

Chromium-Based Browsers Will Ignore Google's Ad-Blocking Ban

Commercial web browsers including Brave, Opera, and Vivaldi won't be disabling ad blocker extensions as desired by Google. These browsers are based on the same open source codebase that is used with Google Chrome. Google maintains an open source project called Chromium as the base of its Chrome browsers.

According to ZDNet, "At the end of May, Google made a new announcement in which it said that the old technology that ad blockers were relying on would only be available for Chrome enterprise users, but not for regular users."

Chromium has become standard for web browsers; except for Safari and Firefox, almost everyone is using Chromium – including Microsoft Edge. Chromium derivatives Brave, Opera, and Vivaldi have said they will not follow Google into crippling ad blocking.

Zorin OS 15 Released

Artyom Zorin, chief executive officer of Zorin OS, has announced the release of Zorin OS 15, the latest version of the Linux-based distribution.

Zorin OS 15 is based on Ubuntu 18.04.2 LTS with the Hardware Enablement (HWE) stack. The LTS base enables Zorin OS developers to focus on making incremental improvements to the distro without having to chase the always moving target, since new versions of Ubuntu are released every six months.



According to a Zorin blog, speed has been a top focus in Zorin OS 15, so the desktop runs dramatically smoother on a wide range of hardware, old and new. "With the introduction of Gnome Shell 3.30 and the Linux kernel 4.18, performance optimizations have been made at every level of the operating system," said the blog post.

Zorin OS also packs Nvidia drivers in the ISO, so users don't need Internet in order to install such drivers.

Zorin OS includes the adaptive desktop, which changes the background to adapt to the brightness of the environment. The new Zorin Auto Theme feature automatically switches the desktop theme into dark mode at sunset and back to light mode after sunrise.

Zorin is targeted at Windows and macOS users who want to switch to desktop Linux.

Zorin OS is available in different editions: <https://zorinos.com/download/>



Ubuntu to Package Proprietary Nvidia Driver

According to reports, Ubuntu developers are planning to add the proprietary Nvidia drivers to the ISO of the next Ubuntu release (19.10).

However, these drivers will not be activated/enabled by default.

The reason for backing these drivers is simple. As mentioned in the Launchpad bug report, “On Ubuntu desktop, without a network connection, the user can elect to install 3rd party drivers (which states that it’ll install graphics driver) but even if the user selects this option, Nvidia proprietary drivers won’t be installed because they are not on the pool of the ISO.”

With drivers backed into the ISO, users can install these drivers without Internet. To ensure that there won’t be any licensing issues, Will Cooke of Canonical said that they have worked with Nvidia to ensure that they are allowed to distribute the drivers on the ISO. Depending on user feedback, Canonical might also backport this to earlier Ubuntu releases.

Canonical will continue to offer open source nouveau drivers as the default driver for Nvidia cards.



Image © belchonock, 123RF.com

openSUSE Leap 15.1 Released

The openSUSE community has announced the release of openSUSE Leap 15.1, the stable release of the Linux-based distribution. OpenSUSE Leap 15.1 is targeted at IT professionals.

The release updates the graphics stack for the distribution, bringing significant improvements. Graphics hardware support released with the 4.19 Linux kernel was backported for Leap 15.1, which uses the 4.12 Linux kernel and supports additional graphics drivers for Graphics Processing Unit (GPU), along with improved support for the AMD Vega chipset. The release also adds a few popular WiFi drivers for more modern wireless chipsets.

OpenSUSE Leap 15.1 also comes with improvements to the YaST administration tool, including a better 4k display (HiDPI) experience. HiDPI displays are now auto-detected, and the UI is auto-scaled, giving the installer a beautifully crisp interface.

Some of the improvements to YaST have made for better management of services. FirewallD can be managed in text mode. There is a new user interface to manage FirewallD, including AutoYaST support/advancements. System administrators will have better control with Salt formulas in the *yast2-configuration-management* module, and management of SSH keys per user will make sys admins tasks much more pleasant.

OpenSUSE Leap 15.1 is available for free download: <https://software.opensuse.org/distributions/leap>

Red Hat Enterprise Linux 8 Available

At the Red Hat Summit 2019 the company announced the availability of Red Hat Enterprise Linux (RHEL) 8.

According to Red Hat (<https://www.redhat.com/en/about/press-releases/red-hat-enterprise-linux-8-every-enterprise-every-cloud-every-workload>), RHEL 8 is redesigned for the hybrid cloud era and built to support the workloads and operations that vary from enterprise data centers to multiple public clouds.

One of the goals of modern tech companies is to simplify tech consumption. RHEL 8 abstracts many of the deep complexities



MORE ONLINE

Linux Magazine

www.linux-magazine.com

Linux Administration Focus

<http://www.linux-magazine.com/tags/view/administration>

Hardening Linux for Production Use

Ken Hess

To protect your production server from attacks, employ these common security tools to help safeguard your system.

Turbocharge Your Network with Zeroshell

Mayank Sharma

Turn an old unused computer into a state-of-the-art router.

ADMIN HPC

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

In the Loop • Jeff Layton

Diving deeper into OpenMP loop directives for parallel code.

ADMIN Online

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

Plundering treasures with Gitrob • Chris Binnie

Automate the search for passwords, secret keys, tokens, and other authentication credentials on GitHub.

Simplify Integration of S3 Storage with Local Resources • Thomas Drilling

The AWS hybrid storage service, known as the Storage Gateway, provides local applications with a seamless connection to Amazon S3 storage. We explain the different gateway types and guide you through their setup.

Shifting Drupal to Amazon’s Cloud

Vidya Mane

If your Drupal installation performs sluggishly or experiences repeated failures, migrating to a public cloud like AWS can solve your problems.



of granular sys admin tasks behind the RHEL web console. The console provides an intuitive, consistent graphical interface for managing and monitoring RHEL system, from the health of virtual machines to overall system performance. To further improve ease of use, RHEL supports in-place upgrades, providing a more streamlined, efficient, and timely path for users to convert RHEL 7 instances to RHEL 8 systems.

RHEL 8 also includes RHEL System Roles, which automate many of the more complex tasks around managing and configuring Linux in production. Powered by Red Hat Ansible Automation, System Roles are preconfigured Ansible modules that enable ready-made automated workflows for handling common, complex sys admin tasks. This automation makes it easier for new systems administrators to adopt Linux protocols and helps to eliminate human error.

Google Brings Linux to Chromebook

Linux-based Chromebooks are not capable of natively running Linux apps and utilities. Last year, Google launched project Crostini to allow Linux apps – primarily command-line tools and utilities – to run natively on Chrome OS using containerization.

According to some media reports, at the Google I/O summit this year, Google announced that “all Chromebooks launched in 2019 will be Linux-ready right out of the box.” It means that all new Chromebooks will have Crostini enabled by default.

“Crostini is the umbrella term for making Linux application support easy to use and integrating well with Chrome OS. It largely focuses on getting you a terminal with a container and easy access to install whatever developer-focused tools you might want. It’s the default first-party experience,” said the Project Crostini page.

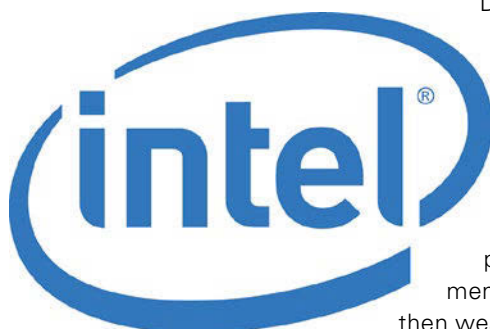
Both Google and Microsoft are trying to lure developers towards their platforms, and they see a benefit in providing Linux command-line utilities that many developers/sys admins need to test, build, and run their cloud-native applications.



New Class of CPU Flaws Affects Almost Every Intel Processor Since 2011

Researchers have released info on a new class of flaws in Intel CPUs that affects every processor made since 2011.

“ZombieLoad attack affects a wide range of desktops, laptops, and cloud computers with Intel processor generations released from 2011 onwards. It can be used to read data that is recently accessed or accessed in parallel on the same processor core,” wrote the Hacker News (<https://thehackernews.com/2019/05/intel-processor-vulnerabilities.html>).



“Dubbed Microarchitectural Data Sampling (MDS) attacks, the newest class of vulnerabilities consist of four different flaws, which, unlike existing attacks that leak data stored in CPU caches, can leak arbitrary in-flight data from CPU-internal buffers, such as Line Fill Buffers, Load Ports, or Store Buffers,” wrote the Hacker News.

Hackers can exploit these vulnerabilities to leak privileged information data from an area of the memory that hardware safeguards deem off-limit and then weaponize it.

Shop the Shop

shop.linuxnewmedia.com

DISCOVER LibreOffice



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

Create your own:

- Word processing docs
- Spreadsheets
- Presentations
- Databases

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

Order online:

shop.linuxnewmedia.com/specials

For Windows, macOS, and Linux users!

Zack's Kernel News



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

By Zack Brown

Author

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

Cleaning Up Dependencies on Older GCC Versions

When Linus Torvalds recently agreed to raise the GCC minimum supported version number to 4.8, it meant that any older system that still used older build tools, including older versions of GCC, wouldn't be able to compile new kernels without upgrading that software. The justification for his decision was partly that the various kernel ports had come to depend on later GCC versions and partly that the various Linux distributions had started shipping with later GCC versions as well. To Linus, this meant that regular users would almost certainly not be inconvenienced by the change; while kernel developers – well, Linus didn't mind inconveniencing them so much.

But it wasn't entirely an inconvenience to developers, as Steven Rostedt recently demonstrated. Now that GCC 4.8 was the new minimum, the kernel no longer had to support older versions of GCC that lacked some of the modern new features. The way this generally works is that the kernel build system checks which version of GCC is installed and then compiles certain kernel features that are specifically coded for that GCC version. This way, by hook or by crook, all kernel features get implemented, even if they have to work around deficiencies in an older compiler. When the older compilers aren't supported anymore, all of that targeted kernel code can simply be torn out by the roots, without anyone making a fuss.

Steven had noticed that function tracing had traditionally used two possible implementations: one using only GCC's `-pg` flag, and one using the `-pg` flag in conjunction with `-mfentry`. Everyone understood that `-pg` plus `-mfentry` was better, but the kernel build system had always had to support using only the `-pg` flag by itself, because older versions of GCC didn't support the `-mfentry` flag. In order to make sure function tracing was supported in all kernels, there needed to be multiple implementations: some using the excellent `-pg` plus `-mfentry`, and others using the less excellent `-pg`. Now that

the minimum GCC version was 4.8, it meant that all supported GCC versions now implemented the `-mfentry` flag, so the `-pg` solution was no longer needed.

Steven posted a patch to remove it.

There was wide applause for this violent and merciless act. Peter Zijlstra and Jiri Kosina both shouted with glee, while Linus smiled contentedly.

It's common with this sort of adjustment that developers will start to look deeper into the given piece of code to identify yet more areas that can be simplified or removed. In this case, as Linus pointed out, there were a bunch of conditionals that no longer had more than one choice, now that the second choice had been taken away. So it would now be possible to get rid of all those conditionals and simplify the code still further.

This type of patch is very popular with Linus and his core group of developers – anything that shrinks the code base, or that shrinks the size of the compiled kernel, is already a patch they like, before they've even looked at what it does. After that, they may object for other reasons. But just tell people your patch removes X hundreds of lines of code, and they start nosing around interestedly, hoping to find an excuse to apply this patch to the tree.

Protecting RAM from Hostile Eyes

Security issues are always weird. They're always something nobody has thought of yet. And then lo and behold, there it is, like a rabbit in a hat. Where did that come from? Or else you're implementing what seems to be a perfectly good feature, like a macro language for your closed source word processor product; what could ever go wrong with that? And lo and behold, you create an entirely new universe of attacks against perfectly innocent users.

In the Linux world, Matthew Garrett saw a sliver of danger lurking in the crack of time between when a piece of RAM is freed by a piece of software and when it gets wiped clean by the kernel. In that moment, a hostile attacker could potentially reboot the system into a dif-

ferent OS, read the RAM, and see the data belonging to that piece of software. If that data was secret and sensitive, there could be a problem.

There are already mechanisms to prevent sensitive data being sent into swap and to make sure RAM gets thoroughly overwritten when the program is done with it. But there was always that sliver of time between release and overwrite, when a reboot might leave the data exposed.

Now, it was also possible to instruct UEFI firmware to wipe all RAM before booting to another OS. But as Matthew pointed out, this would actually multiply the time it took to boot the system by quite a bit, producing a truly unacceptable slowdown.

A better approach, he felt, was for Linux itself to wipe all RAM before shutting down. Assuming the attacker didn't simply turn off the power to induce a cold boot, Linux could then shut down gracefully, deleting all data and leaving the attacker with nothing to target. There was real value in this, Matthew said. Consider if the OOM killer suddenly killed a process, without giving it any time to exit cleanly and delete its own data. The operating system itself would have to be the last line of defense. Matthew posted a patch to implement his idea.

There was a bunch of support for this. However, Christoph Lameter pointed out that Matthew's patch only wiped RAM a little bit earlier than it would otherwise be wiped, given that RAM was always scrubbed clean before being allocated to another process. If anything, he said, Matthew's patch would reduce the width of the dangerous sliver of time, but it wouldn't eliminate it completely. And since Matthew's patch extended the behaviors of system calls and created whole new kernel behaviors, it seemed to Christoph that this slight reduction of the attack surface was not worth the added complexity.

It's unclear whether Christoph's objection might stop the patch from going into the kernel. Clearly his point is valid. But there was also a fair amount of initial support for the patch, and it seems like a common sense precaution in some ways. Ultimately, I believe, if the security hole remains open, it may make Linus Torvalds less willing to accept a patch that adds any sort of complexity to the kernel. But to close the hole entirely may prove to be impossible.

Verifying Return Addresses

Some security fixes look cool but are nonstarters for other reasons. For example, Mike Rapoport from IBM wanted to strengthen Linux address space isolation, which on its own merits could be very useful. Address space isolation is where a given process can only "see" a certain range of RAM addresses, and anything outside of that range is completely cut off from potential attack.

Mike's approach, he explained, was to have the kernel confirm every address that popped off the stack. This way, the return addresses of any function calls made in an address space, would be guaranteed to bring execution back to that calling function. Without Mike's patch, a hostile user might be able to insert a fake return address that didn't exist in the kernel's predefined symbol table. Then when that address came up off the stack, control would be handed over to a hostile piece of code, possibly with root access. Mike's patch stopped that dead in its tracks.

Unfortunately, as Mike himself admitted, confirming that every single return address had a corresponding entry in the symbol table was going to be a high-intensity operation, happening many times per second on a normal system. He saw no way to accomplish this without the kernel taking some kind of generalized speed hit.

Not only that, but as Jiri Kosina said, Mike's code was actually incompatible with other security projects that attempted to solve similar problems. So between the speed hit and the project conflicts, the otherwise useful aspects of Mike's patch were not sufficient to gain any interested supporters among the kernel developers.

It's not such a rare outcome – someone works many hours to prove that a given feature is possible, only to find that it doesn't work with another obscure project that seems to have more favor or that the feature itself has some inescapable drawback like a speed hit or something else that dooms it.

At the same time, in many cases, a failed project points the way towards a better approach that later on succeeds. It's almost never the case that a project is simply bad and useless, especially if a developer thought it was important enough to invest so much time. ■■■



Controlling a cheap smart plug from Linux

Liberation

You could spend hundreds of dollars on specialized IoT appliances and fixtures, or you could just hack a smart plug and talk to it with your Linux system.

By Răzvan T. Coloja

Have you ever wondered if you could use Linux to power your entire IoT house? Make it prepare your coffee in the morning – even if your coffee machine is older than you and does not have a CPU? Or maybe start the PC at work with the push of a button while you sit down at home to enjoy breakfast?

Homeowners spend hundreds of dollars on high-tech light bulbs and appliances that tie in with IoT networks. But what if you like your old appliances? What if you are concerned about the security and privacy issues of a full-blown IoT infrastructure, or maybe you just want to keep it simple while maximizing your flexibility for future customization.

The easiest and most foolproof version of an IoT network is a simple smart plug – a small device that plugs into an outlet and lets you control when the power flows. You can plug any electrical device into the smart plug and turn it on or off remotely – even if the device itself was never intended for IoT use.

This article describes how to control an inexpensive smart plug from a Linux system. Once you get control over the plug, you can use all the tools and powers of Linux to automate your electrical devices.

The solution described in this article requires a smart plug for each device, Linux, and an FTDI converter to program the smart plug. I'll use the Sonoff S20 smart plug from ITEAD as an example for this article. The Sonoff S20 [1] is a cheap Chinese smart plug that you can control via an Android application. I'll

Author

Răzvan T. Coloja is a Romanian psychologist who has worked both as an administrator since 1997 and as the editor-in-chief of the now-defunct Romanian IT magazine *MyLINUX*. He was the executive editor of the IT-centric magazine *MyCOMPUTER* for three years, and he has worked as an editor for *CONNECT* magazine. He has published Linux articles in international print magazines and online.

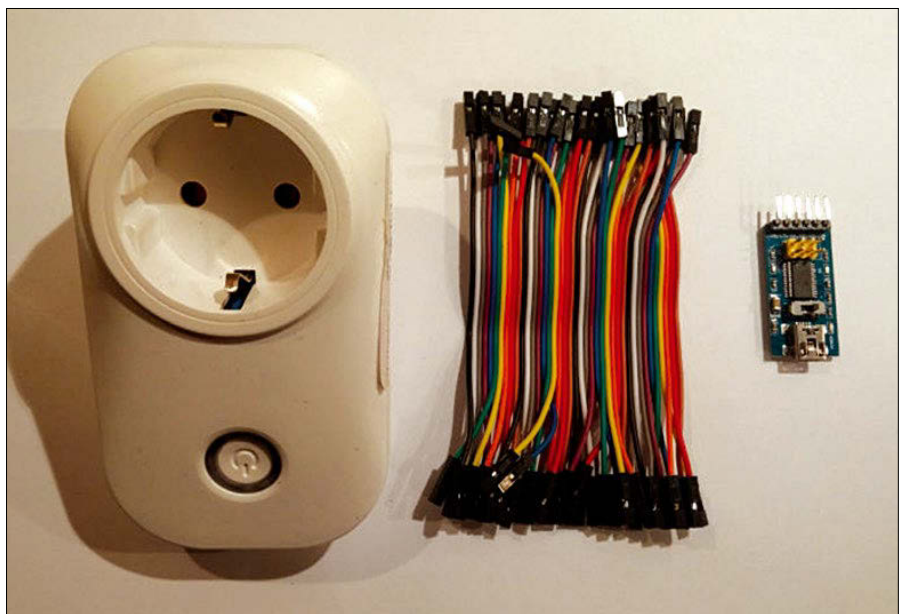
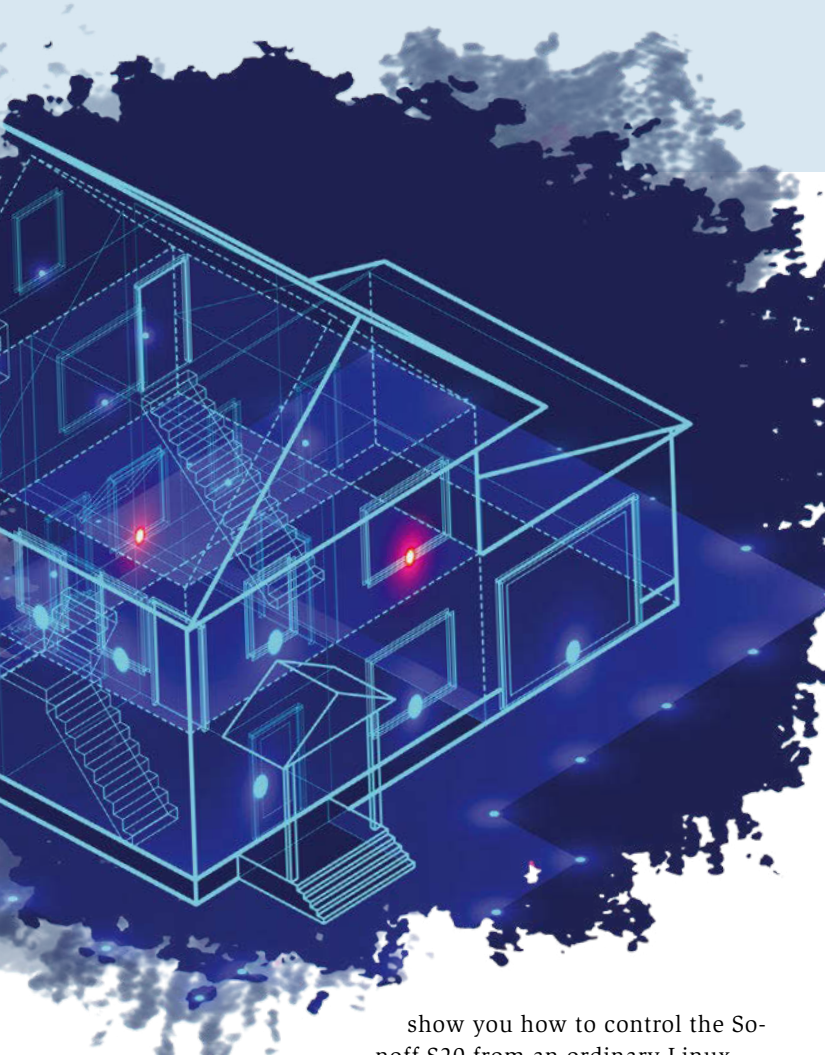
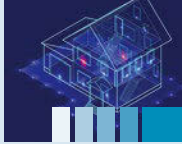


Figure 1: The Sonoff S20 smart plug is available for both European and American-style outlets. This project also requires colored cables and the 3.3V FTDI FT232RL USB to TTL converter.



show you how to control the Sonoff S20 from an ordinary Linux computer.

For this solution, you'll need one FT232RL USB to TTL FTDI Controller [2] that can handle 3.3V. This part costs about US\$4 online, and you only need one. You also need four differently-colored cables to connect the converter to the smart plug (Figure 1). The FT232RL USB to TTL FTDI controller has a switch that makes it output either 5V or 3.3V. Leave it on 3.3V; otherwise, you risk frying your smart plug or even your laptop.

The Concept

The procedure outlined in this article takes a little more trouble than most people are used to going to for a \$12 smart plug, but hacks such as this one have a long history with the Linux community. This is a proof of concept, of course. A similar approach might work with a different smart plug, but you'll need to adapt the steps as necessary.

The basic idea is, you flash new firmware onto the smart plug that supports the open MQTT protocol [3], which is

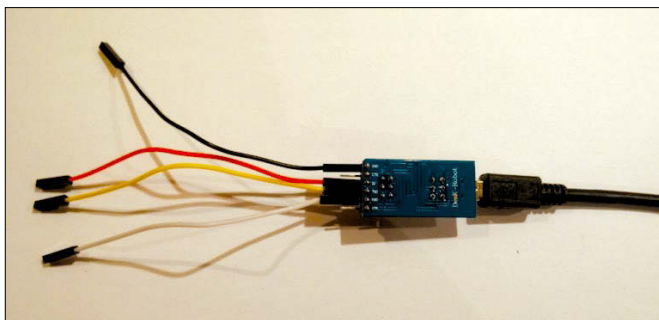


Figure 2: Black goes to *GND*, orange to the *5V* pin, yellow to *TXD*, and white to *RXD*.

often used for IoT communication. Then you install the open source Mosquitto message broker [4], which speaks MQTT, onto your Linux computer. Mosquitto will manage communication between Linux and the smart plug. You can then enter a command for the smart plug directly into your web browser as a specialized URL. You can also use the Bash `curl` command to send an HTTP-based command to the smart plug. I'll show you how to create rule-sets that turn on the power at a specific time or after a specific interval. The ability to access the plug through Bash also means you can integrate the smart plug into scripts and add the scripts to auto-launch utilities such as cron.

Preparation

Insert a USB cable in the converter's USB slot and connect the four cables to four of the connector's six pins. One cable should connect to the pin marked as *GND*, one to *5V*, one to *TXD*, and the last one to *RXD* (Figure 2).

Inside the Smart Plug

Now you need to unscrew the Sonoff S20 smart plug. There is only one screw underneath the label. All you have to do after

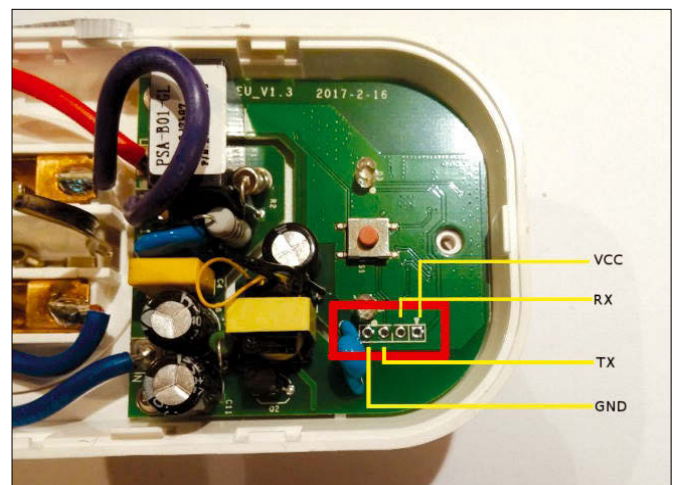


Figure 3: Layout.

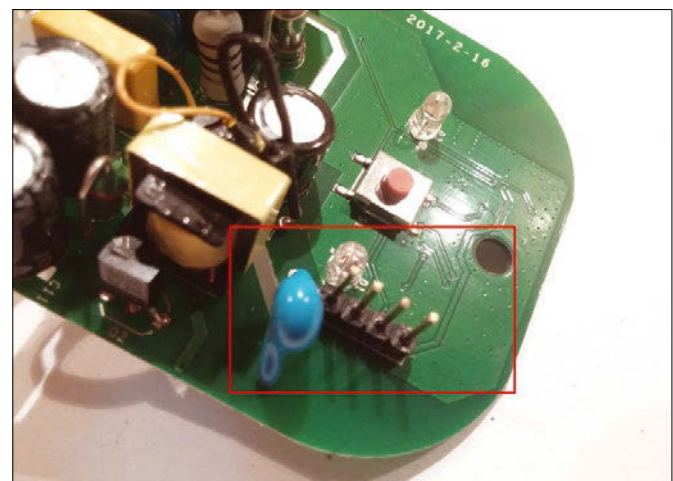


Figure 4: Solder four pins to the board.

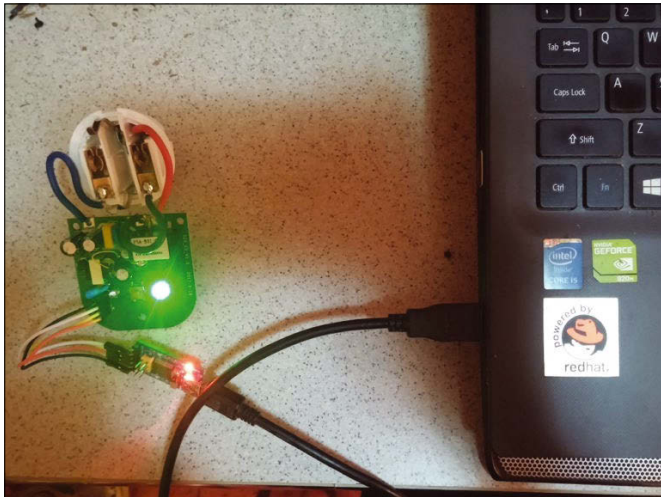


Figure 5: The smart plug's LED glows green when first connected to a laptop's USB port.

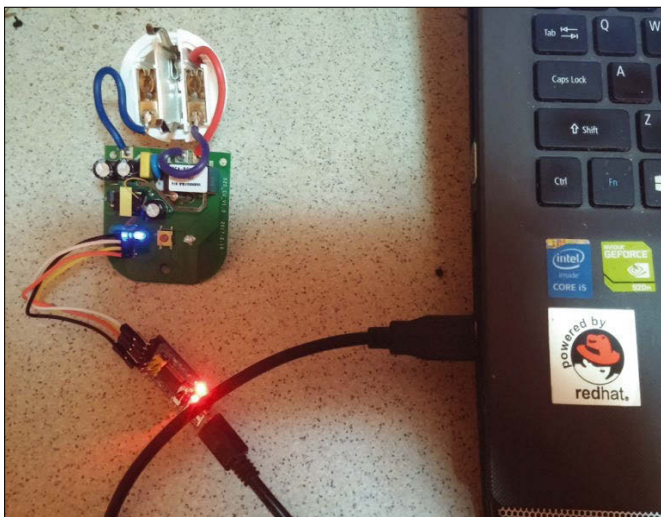


Figure 6: You know you have entered programming mode if the LED glows blue.

you remove the screw is pry open the plastic enclosure by inserting a butter knife or a screwdriver along the edges and gently separating them. Inside the casing, you will find a board pinned by two screws that need to be taken out so you can get access to the underside of the board. Right next to the smart plug's power button you will see four holes (Figure 3). These holes are important; you will need to connect them to the four free ends of the four differently-colored cables. For this you need four pins or four pieces of copper wire thin enough to fit through the holes. You need to solder these pins onto the board so the four cable ends can be fitted into these pins (Figure 4).

Flashing the Firmware

Connect the *GND* cable coming from the controller to the *GND* pin on the board, the *5V* cable to *VCC*, the *RDX* to *TDX*, and the *TDX* to *RDX*.

Now power up your Linux laptop or PC and connect the USB cable to it. The FTDI controller's LED should glow red, and the LED on the smart plug should

glow green (Figure 5). The next step is to put the Sonoff S20 into programming mode. Press the smart plug's power button (it only has one button) and, while holding it down, take out the USB cable from the laptop. While still holding down the button, wait for three seconds, insert the USB cable back into the USB port, and release the power button after another two seconds have passed. If all went well, the smart plug's LED light should glow blue (Figure 6). If not, repeat the steps until it does.

If you do an `lsusb` in a console emulator, the Linux kernel should detect the FTDI controller (Figure 7).

To check what USB port the smart plug is connected to use

```
$ dmesg | grep tty
```

The Sonoff S20 comes with an ESP8266 chip [5]. The ESP8266 includes WiFi support and a full TCP/IP stack to receive commands for controlling the device. To get control over the smart plug, I'll replace the firmware on the ESP8266 with the ESPEasy open source firmware [6], which is available for free download. You'll need to download the `esptool.py` Python script to flash a new firmware image on the Sonoff S20. Be sure to make the script executable:

```
$ wget -c https://raw.githubusercontent.com/letscontrolit/ESPEasy/mega/test/esptool.py
$ chmod +x esptool.py
```

You also need to install Python and the Pip package manager for Python packages. If you use a Debian-based distribution, you can install Python and Pip with:

```
$ sudo apt install python pip
```

Once Python and Pip are in place, you need to install `pyserial` using Pip:

```
$ pip install pyserial
```

Download the 2.0.0-dev12 version of the ESPEasy firmware with:

```
$ wget -c https://github.com/letscontrolit/ESPEasy/releases/download/v2.0.0-dev12/ESPEasy_v2.0.0-dev12.zip
```

Unpack the firmware from the ZIP file, and you should get a firmware file named `ESPEasy_v2.0.0-dev12_normal_1024.bin`. Assuming the USB port of the FTDI controller is connected to `/dev/ttyUSB0` and that you remained in the same directory as



Figure 7: Output of the `lsusb` command.



```

cypress@aspire:~/Desktop$ sudo ./esptool.py --port /dev/ttyUSB0 --baud 115200 write_flash 0x0000 ESPEasy_v2.0.0-dev12_normal_1024.bin -fs 8m
esptool.py v1.3
Connecting...
Running Cesanta flasher stub...
Flash params set to 0x0020
Writing 569344 @ 0x0... [ 49504 (26 %)]
    
```

Figure 8: Flashing the new firmware.

the firmware file, you can now reflash the smart plug with the following command (Figure 8):

```

$ sudo ./esptool.py --port /dev/ttyUSB0 --baud 115200 write_flash 0x0000 ESPEasy_v2.0.0-dev12_normal_1024.bin -fs 8m
    
```

Make sure you use 115200 as a baud rate. The flashing will be done in about a minute, and after that, you can disconnect the USB cable and reconnect it so that the Sonoff S20 will exit programming mode.

The new firmware has been flashed. If you check your surrounding WiFi networks, you will find one called *ESP_Easy_0*. This network's WiFi password is *configesp*; you will have to connect to

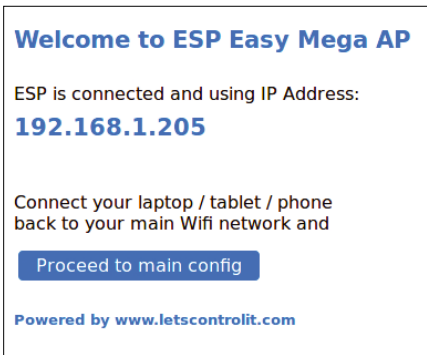


Figure 9: The welcome message of the web page for the new firmware.

it with the help of a web browser to be able to configure the smart plug. The smart plug's IP address has changed to 192.168.1.205, so be sure to navigate to this IP address. A web page with a message should greet you (Figure 9). Now you have to disconnect from this WiFi

network, connect to your own WiFi network, and click the *Proceed to main config* button. This button will allow you to pick your WiFi network and enter your WiFi network password. You can change your smart plug's IP configuration to match the netmask of your own local area network (Figure 10) or use DHCP to give the Sonoff S20 a dynamic IP address (Figure 11).

Configuration

The new firmware you flashed has Telnet features, can handle both 2.4 and 5GHz WiFi networks simultaneously, has a logging system, has an information panel with statistics, and even has a place to set custom commands to be executed by the smart plug's power button. It can send you email notifications, update the current firmware through a web interface so you won't need to use an FTDI controller again, and – most importantly – it has support for the MQTT IoT protocol, which you will use to remotely control the Sonoff S20 through Linux.

You'll need to set up openHAB and an openHAB client like Mosquitto. Mosquitto is an open source MQTT broker maintained by the Eclipse foundation. To install Mosquitto on your Linux computer, use:

```

$ sudo apt install mosquitto mosquitto-clients
    
```

You can now set up the configuration for the power button with

```

# mosquitto_sub -h my.mqtt.server -t "#" -v /sonoff-s20/Relay/State 1
    
```

and

```

# mosquitto_sub -h my.mqtt.server -t "#" -v /sonoff-s20/Relay/State 0
    
```

Using Linux and Mosquitto, you can start and stop the smart plug from the command line with:

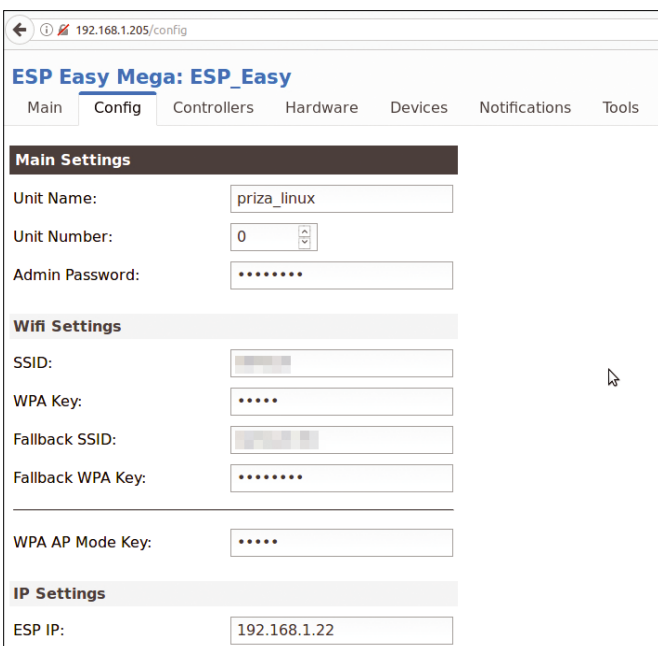


Figure 10: Configure the name of the smart plug and set its IP address.

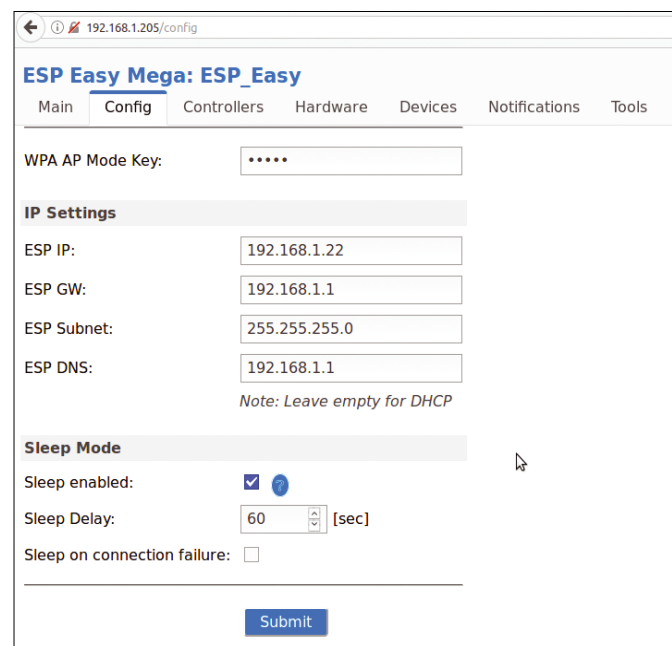


Figure 11: You can use DHCP instead of configuring a static address.



ESP Easy Mega: priza_linux

Main Config Controllers Hardware **Devices** Notifications Tools

Task Settings

Device:

Name:

Enabled:

Sensor

Internal PullUp:

Inversed Logic:

Note: Will go into effect on next input change.

1st GPIO:

Switch Type:

Switch Button Type:

Send Boot state:

Data Acquisition

Send to Controller IDX:

Delay: [sec] (Optional for this Device)

Values

Value	Name
1	State

Figure 12: Button configuration.

ESP Easy Mega: priza_linux

Main Config Controllers Hardware **Devices** Notifications Tools

Task Settings

Device:

Name:

Enabled:

Sensor

Internal PullUp:

Inversed Logic:

Note: Will go into effect on next input change.

1st GPIO:

Switch Type:

Switch Button Type:

Send Boot state:

Data Acquisition

Send to Controller IDX:

Delay: [sec] (Optional for this Device)

Values

Value	Name
1	State

Figure 13: Relay configuration.

```
# mosquitto_pub -h 192.168.1.22 -t "/sonoff-s20/gpio/12" -m "1"
```

and

```
# mosquitto_pub -h 192.168.1.22 -t "/sonoff-s20/gpio/12" -m "0"
```

where 0 stands for *Power Off* and 1 for *Power On*, assuming the IP address of the smart plug is 192.168.1.22.

After you configure the smart plug using the built-in web server, click on the *Devices* tab. Under *Task Settings*, choose *Switch input* as the device and name it *Button* (Figure 12).

ESP Easy Mega: priza_linux

Main Config Controllers Hardware **Devices** Notifications Tools

<	>	Task	Enabled	Device	Name	Port	Ctr (IDX)	GPIO	Values
<input type="button" value="Edit"/>	1	<input checked="" type="checkbox"/>	Switch input	Button				GPIO-0	State: <input type="text" value="0"/>
<input type="button" value="Edit"/>	2	<input checked="" type="checkbox"/>	Switch input	Relay			0 (0)	GPIO-12	State: <input type="text" value="0"/>
<input type="button" value="Edit"/>	3								
<input type="button" value="Edit"/>	4								

Figure 14: In the end, you need to have these two tasks present under the *Devices* tab.

ESP Easy Mega: priza_linux

Main Config Controllers Hardware **Devices** Notifications **Tools**

Advanced Settings

Rules:

Controller Settings

MQTT Retain Msg:

Message Delay: [ms]

NTP Settings

Use NTP:

NTP Hostname:

Timezone Offset: [minutes]

DST:

Log Settings

Syslog IP:

Syslog Level:

Figure 15: Make sure MQTT support is active.

ESP Easy Mega: priza_linux

Main Config Controllers Hardware **Devices** **Rules** Notifications Tools

Rules

Edit:

```
On Button#State=1 do
  if [Relay#State]=0
    gpio,12,1
  else
    gpio,12,0
  endif
enddo
```

Current size: 96 characters (Max 2048)

Figure 16: A ruleset example.

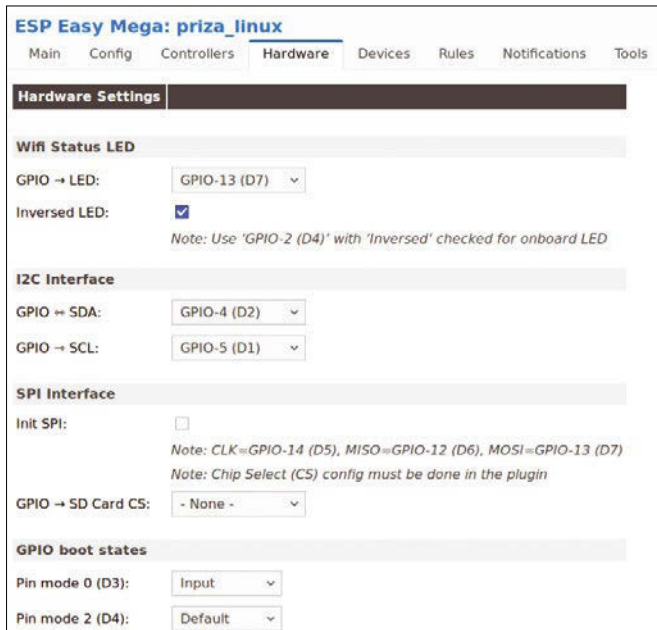


Figure 17: Make sure you make the necessary changes under the *Hardware* tab.

Listing 1: PowerButton Configuration

```
01 On Button#State=1 do
02 if [Relay#State]=0
03 gpio,12,1
04 else
05 gpio,12,0
06 endif
07 endon
```

Listing 2: Start and Stop Functions

```
01 On start do
02 gpio,12,1 //start
03 endon
04
05 On stop do
06 gpio,12,0 //stop
07 endon
```

Listing 3: Making Coffee for Seven Minutes

```
01 On startmakingcoffee do
02 gpio,12,1 // power on the coffee machine
03 timerSet,1,420 // 60*7=420 seconds
04 endon
05
06 On stopmakingcoffee do
07 timerSet,1,0 // you can force-stop the plug before the 7 minutes pass
08 endon
09
10 On Rules#Timer=1 do
11 gpio,12,0 // stop powering the coffee machine
12 endon
```

Listing 4: The Internal Scheduler

```
01 On Clock#Time=All,05:45 do // Power-On at 5:45 in the morning
02 gpio,14,0
03 endon
```

Listing 5: Power-On on a Day of the Week

```
01 On Clock#Time=Sun,08:30 do // Only on Sunday at 8:30 in the morning
02 gpio,14,0
03 endon
```

Check the box next to the *Enabled* field, check the one next to *Internal PullUp*, and underneath that select *GPIO-0 (D3)* as the 1st GPIO. Choose *Switch Type* as *Switch* and *Switch Button Type* as *Normal Switch*; then, under the first value name, enter *State*. Click the *Submit* button to save these settings. Now do the exact same thing for *GPIO-12 (D6)*, but this time also check the *Send to Controller* checkbox and name the new device *Relay* instead of *Button* (Figure 13).

Under the *Devices* tab, you now should have two devices: one named *Button* with a GPIO value of *GPIO-0* and another named *Switch* with a GPIO value of *GPIO-12* (Figure 14).

Now you need to create a ruleset for the power button. Click on the *Tools* tab and check the checkbox next to *Rules* (Figure 15). A new menu will appear, in which you can insert your own rules that define the behavior of the power button. Enter the contents of Listing 1 into this editable field (Figure 16).

Under the *Notifications* tab, you can configure your email address so that whenever the smart plug does something, you will be notified through an email. Finally, under the *Hardware* tab, you need to make the following changes: Set *GPIO-LED* to *GPIO-13 (D7)* and check the *Inverse LED* checkbox.

Under *I2C Interface* set *GPIO-SDA* to *GPIO-4 (D2)* and *GPIO-SCL* to *GPIO-5 (D1)* (Figure 17).

To actually control the Sonoff S20, create a new ruleset with the code in Listing 2.

Control the Smart Plug from a Web Browser

Save the new ruleset, and from this moment on, you can open up a browser tab and launch

```
http://192.168.1.22/control?cmd=event,start
```

to start the smart plug and

```
http://192.168.1.22/control?cmd=event,stop
```

to power it off remotely. Any device connected to it will thus power on or power off along with the smart plug. To start the plug and leave it powering whatever it has been connected to for exactly seven minutes, you can use the ruleset in Listing 3. Seven minutes is plenty of time for your coffee machine to get ready, make coffee, and power off when it's done.

Using curl from the CLI

But you don't have to use a web browser to give commands to the Sonoff S20. You can use the Bash `curl` command. To power on the smart plug and let it automatically shut down after seven minutes, use:



```
$ curl http://192.168.1.22/control?cmd=event,startmakingcoffee
```

The Timer Function

The smart plug has an internal scheduler. It can be used to program the Sonoff S20 to start and stop at given times. Make a ruleset like the one in Listing 4 to start the plug at 05:45 each morning.

Or you can make it power on depending on the day of the week at predefined hours (Listing 5).

You can even make it do things between certain hours during the day, as shown in Listing 6.

Conclusion

With a \$4 FTDI controller, four cables, and a bit of soldering, you can turn a Sonoff S20 smart plug into a remote-access power-handling device. Using `curl` or a web server with predefined HTTP links or JPEG buttons, you can control

your entire house, from lights to the fridge to anything that needs power from a wall socket. If you set up a VPN, you can SSH into your OpenWrt-powered router, and from there, pass commands via WiFi LAN to any of your smart plugs.

You can start the air conditioning at your log cabin during hot Summer days while on route, turn on your workplace PC before you enter the office building, warm up tea, or wake up to a hot pot of freshly-made coffee in the morning. Program it to do certain things at certain times of the day or week, plug it in, and forget about it. As long as there's power from the power grid and you can access the WiFi network, you don't need an Android app to control your house. The living room TV could power off at midnight if you fall asleep in front of it – even if it's not a Smart TV. Turn your vintage radio into an alarm clock, or water the lawn periodically without a dedicated hose controller – all with a few smart plugs and Linux. ■■■

Listing 6: Power-On at a Time

```
01 On Pir#Switch=1 do
02 If %systime% < 16:00:00 // start at 16:00
03 Gpio,16,1
04 Endif If %systime% > 16:33:00 // stop at 16:33
05 Gpio,16,1
06 Endif
07 Endon
```

Info

- [1] Sonoff S20: <https://www.itead.cc/smart-socket.html>
- [2] FT232RL USB to TTL FTDI Controller: <https://octopart.com/ft232rl-ftdi-2151617>
- [3] MQTT: <http://mqtt.org/>
- [4] Mosquitto: <https://mosquitto.org/>
- [5] ESP8266: <https://en.wikipedia.org/wiki/ESP8266>
- [6] ESPEasy: <https://github.com/letscontrolit/ESPEasy>



IT Highlights at a Glance

The collage features several overlapping banners:

- ADMIN HPC**: A banner for the HPC Update, mentioning 'The New IT ADMIN' and 'HPC Update'.
- Linux Update**: A large blue banner titled 'EXPLORING THE WORLD OF LINUX'.
- Discover LibreOffice!**: A banner promoting the free office suite.
- ADMIN Update**: A banner with 'Hottest Links' and 'Highlights'.
- FEATURED ARTICLES**: A section with various article titles.
- FURTHER READING**: A section with additional links and resources.
- MOST READ**: A section highlighting popular content.
- 2018 Archives Available Now!**: A banner for past issues.

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

Linux Update • ADMIN Update • ADMIN HPC

Keep your finger on the pulse of the IT industry.

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

Linux Update: <https://bit.ly/Linux-Update>

SDC19

Storage Developer Conference
September 23-26, 2019
Santa Clara, CA

SDC Keynotes include:

- Gilbert Bouzeid, eBay
- Skottie Miller, Dreamworks
- Brad Settlemyer, Los Alamos National Labs

View the full agenda at
www.storagedeveloper.org

Created BY Developers FOR Developers

Get the \$200 of on registration by
registering at www.snia.org/SDcode
and using code SDC19LINUXPRO.

www.storagedeveloper.org

SNIA[®]



Stay punctual with chrony

On Time

The chrony implementation of the network time protocol provides an alternative to the familiar NTP daemon. *By Chris Binnie*

Keeping your computer's clock set correctly is important; otherwise, a number of bad things can happen. For example, on a server, the logs can become inaccurate and confusing (and therefore insecure and ineffective). Also, when all files on your server or your desktop are modified or accessed, the time is noted. For day-to-day tasks, this timekeeping is moderately important, but for security it can be key, as you might imagine, if an attacker changes files on a system.

Networking can become disorganized, too. You may find with older implementations of secure web pages (SSL, Secure Sockets Layer) that the time on your desktop machine must have the correct time for the connections to work as expected.

Moreover, many anti-spam email systems treat time-warped email sent from

the future as unwanted spam by default. Some systems instantly and silently refuse to process the offending email in any way, shape, or form. Finally do not forget that scheduled tasks (e.g., downloading new packages) also get messed up if your system clock is skewed.

In this article, I show you a lightweight alternative to the long-residing champion of time synchronization `ntpd`, the network time protocol daemon. The excellent `ntpd` has been found on many a Unix-like operating system (OS) for decades. Another kid on the block, however, is called `chrony` [1], which has been around since dial-up modems were singing high-pitched cacophonies all over the planet. Rest assured, then, that it's a time-tested and trusted piece of software. I'll run through the basics of its installation and operation and how to secure it for servers and desktops alike.

What Time Is It?

The `chrony` package installed on my Debian derivative Ubuntu/Mint box offers the following additional packages: `libtomcrypt0`, `libtommath0`, and `time-limit`. Thankfully, they take up barely any disk space because the clever `chrony` is so lightweight and performant. At installation, the timely `chrony` managed to configure itself at lightning speed and then synchronize automatically with four remote time servers.

Two main binaries compose the `chrony` package: The daemon `chronyd`, which sits quietly running in the background, and `chronyc` (Listing 1), which is a command-line interface (CLI) program for adjusting an already running `chronyd`.

If you've used `ntpd` before, the output looks familiar and is relatively easy to understand: The network is connecting to four clocks and tells where in the pecking order (Stratum) they are for accuracy, with the times presented relative to your network time. In super-simple terms, the higher the stratum the more accurate the clock. A stratum 1 server, for example, might be connected to a reference clock or some other type of atomic clock or similarly reliable device. You will not see a stratum 0 device connected to a network by design, because such a reference clock would need a time server to do its networking for it.

Just typing `chronyc` without arguments will offer you a familiar CLI prompt, from which you can type a number of

Listing 1: `chronyc`

```
$ chronyc sources
```

```
210 Number of sources = 4
```

MS Name/IP address	Stratum	Poll	Reach	LastRx	Last sample
^- time.shf.uk.as44574.net	3	6	17	32	+5747us[+5747us] +/- 113ms
^* ntp3.wirehive.net	2	6	17	32	+33ms[+17ms] +/- 92ms
^- www.bhay.org	2	6	17	31	+11ms[+11ms] +/- 38ms
^- h37-220-20-12.host.redsta	2	6	17	31	+1181us[+1181us] +/- 72ms

native chrony commands (or you can enter them directly after the command, which I'll touch on in a moment). Table

1 shows the CLI commands available, as listed by `chronyc help`. As you can see, chrony is highly configurable.

You can run your commands of choice directly on the CLI or within the chrony shell. This flexibility could help with

Table 1: chronyc Commands

Command	Function
<code>accheck <address></code>	Check whether NTP access is allowed to <address>
<code>activity</code>	Check how many NTP sources are online/offline
<code>add peer <address> ...</code>	Add a new NTP peer
<code>add server <address> ...</code>	Add a new NTP server
<code>allow [<subnet-addr>]</code>	Allow NTP access to that subnet as a default
<code>allow all [<subnet-addr>]</code>	Allow NTP access to that subnet and all children
<code>burst <n-good>/<n-max> [<mask>/<masked-address>]</code>	Start a rapid set of measurements
<code>clients</code>	Report on clients that have accessed the server
<code>cmdaccheck <address></code>	Check whether command access is allowed to <address>
<code>cmdallow [<subnet-addr>]</code>	Allow command access to that subnet as a default
<code>cmdallow all [<subnet-addr>]</code>	Allow command access to that subnet and all children
<code>cmddeny [<subnet-addr>]</code>	Deny command access to that subnet as a default
<code>cmddeny all [<subnet-addr>]</code>	Deny command access to that subnet and all children
<code>cyclelogs</code>	Close and reopen logfiles
<code>delete <address></code>	Remove an NTP server or peer
<code>deny [<subnet-addr>]</code>	Deny NTP access to that subnet as a default
<code>deny all [<subnet-addr>]</code>	Deny NTP access to that subnet and all children
<code>dump</code>	Dump all measurements to save files
<code>local off</code>	Disable server capability for unsynchronized clock
<code>local stratum <stratum></code>	Enable server capability for unsynchronized clock
<code>makestep [<threshold> <updates>]</code>	Correct clock by stepping
<code>manual offlonreset</code>	Disable/enable/reset <code>settime</code> command and statistics
<code>manual list</code>	Show previous <code>settime</code> entries
<code>maxdelay <address> <new-max-delay></code>	Modify maximum round-trip valid sample delay for source
<code>maxdelayratio <address> <new-max-ratio></code>	Modify maximum round-trip delay ratio for source
<code>maxdelaydevratio <address> <new-max-ratio></code>	Modify maximum round-trip delay to standard deviation of round-trip delay ratio for source
<code>maxpoll <address> <new-maxpoll></code>	Modify maximum polling interval of source
<code>maxupdateskew <new-max-skew></code>	Modify maximum skew for a clock frequency update to be made
<code>minpoll <address> <new-minpoll></code>	Modify minimum polling interval of source
<code>minstratum <address> <new-min-stratum></code>	Modify minimum stratum of source
<code>offline [<mask>/<masked-address>]</code>	Set sources in subnet to offline status
<code>online [<mask>/<masked-address>]</code>	Set sources in subnet to online status
<code>password [<new-password>]</code>	Set command authentication password
<code>polltarget <address> <new-poll-target></code>	Modify poll target of source
<code>reselect</code>	Reselect synchronization source
<code>rtcdata</code>	Print current real-time clock (RTC) performance parameters
<code>settime <date/time></code>	Manually set the daemon time (e.g., Nov 21, 1997 16:30:05 or 16:30:05)
<code>smoothing</code>	Display current time smoothing state
<code>smoothtime resetlactivate</code>	Reset/activate time smoothing
<code>sources [-v]</code>	Display information about current sources
<code>sourcestats [-v]</code>	Display estimation information about current sources
<code>tracking</code>	Display system time information
<code>trimrtc</code>	Correct RTC relative to system clock
<code>waitsync [max-tries [max-correction [[max-skew]]]</code>	Wait until synchronized
<code>writerc</code>	Save RTC parameters to file
<code>authhash <name></code>	Set command authentication hash function
<code>dns -nl+n</code>	Disable/enable resolving IP addresses to hostnames
<code>dns -4 -6 -46</code>	Resolve hostnames only to IPv4/IPv6/both addresses
<code>timeout <milliseconds></code>	Set initial response timeout
<code>retries <n></code>	Set maximum number of retries
<code>exitlquit</code>	Leave the program
<code>help</code>	Generate this help

scripting. The tracking command, for example, offers a very useful summary of chrony's status (Figure 1) and can be achieved with either

```
$ chronyc tracking
```

or

```
$ chronyc
chronyc> tracking
```

Next, I'll look at the config file that runs the super-slick chrony.

You're Late

The main file to read is located (on my Debian derivative) at `/etc/chrony/chrony.conf`. In the `chrony` directory, you'll also find the `keys` file if you want to change authentication options. A number of useful comments can help when you're unsure about an option in the main file. In a moment, I'll look at three areas you might want to consider altering.

Additionally, first have a look at the excellent logging found, unsurprisingly, under `/var/log/chrony`. In that directory, you'll find the `statistics.log`, `measurements.log`, and `tracking.log` files. As you'd expect, the considerate chrony drops a `logrotate` profile in place to keep your system sane and juggles your logs automatically once a week by default. I'd recommend having a rummage in those three logfiles if you're tuning your time server sources. More on that in a moment.

Even a Stopped Clock ...

At least two or three worthwhile changes can help you take full advan-

```
chronyc> tracking
Reference ID      : 139.59.199.215 (139.59.199.215)
Stratum          : 3
Ref time (UTC)   : Mon Feb  4 10:24:09 2019
System time      : 0.000078427 seconds slow of NTP time
Last offset      : +0.543328822 seconds
RMS offset       : 0.543328822 seconds
Frequency        : 15.412 ppm fast
Residual freq    : +2073.618 ppm
Skew             : 24.519 ppm
Root delay       : 0.041239 seconds
Root dispersion  : 1.835499 seconds
Update interval  : 257.4 seconds
Leap status      : Normal
chronyc>
```

Figure 1: The `chrony tracking` command offers a useful summary.

tage of the trimmed-down `ntpd` model offered by `chrony` to help limit your attack surface. I'd recommend exploring these yourselves, because differing versions of `chrony` might do things slightly differently. For example, I think some defaults were changed in version 2.0.

Although `chrony` is a feature-filled time server, it's safe to say that not all the features will be required in most cases. As I alluded to at installation time, it was up and running in a few seconds.

You might simply want to keep your local clock synced, rather than allowing other machines to ask your machine the time. The venerable `chrony` allows you to bind to your `localhost` (`127.0.0.1`) with the `bindcmdaddress` option, which you can add to the config file mentioned above. Of course, you'll need to bounce your `chronyd` daemon after making a change (most likely with the `systemctl restart chronyd` command).

Adding the following option to your config file will ensure that the wider network to which your system is connected can't connect to `chronyc`:

```
bindcmdaddress 127.0.0.1
```

If you're using IPv6, make sure this option follows the above IPv4 option:

```
bindcmdaddress ::1
```

The well-written FAQ [2] provides much more information.

To disable all NTP client or peer connections completely, add

```
port 0
```

to the config file. This option stops requests ever even making it to the `chrony` daemon.

You can also consider this similar-looking option,

```
cmdport 0
```

with which you can switch off all access to `chronyc` (the CLI binary), outside of the local `chrony` user (which it runs under) and the machine's `root` user.

Just in Time

Amazon Web Services (AWS) began recommending `chrony` some time ago. The documentation [3] explains that AWS runs the Amazon Time Sync Service, to which it recommends connecting from Elastic Compute Cloud (EC2) instances. Sounding a little like a James Bond movie, the AWS docs go on to explain that the service "uses a fleet of satellite-connected and atomic reference clocks in each region to deliver accurate current time readings." In English, I suspect that means each continent's varying clusters of AWS data centers have access to atomic clocks for extra accuracy.

Not surprisingly, AWS appreciates the value to their customers of getting time syncing working correctly. As mentioned, it's a critical part of any production system's health. AWS docs confirm the `/etc/chrony.conf` path for that config file, so I'll hazard a guess that's the location of the file on RHEL derivatives, as well. AWS suggests adding the line

```
server 169.254.169.123 prefer iburst
```

to the config file. If you're using their in-house Amazon Linux 2 OS for your server instance, you can ignore that instruction because it already defaults to using the AWS time service.

The funny-looking IP address above is in the `LINKLOCAL-RFC3927-IANA-RESERVED` address range (`169.254.0.0/16`), which isn't routed out onto the Internet, so it is a fast, local way of syncing with the AWS internal systems. Note that it ends with `123`, which is the usual NTP port, making it easier for reference.

Only a Question of Time

The Amazon Time Sync Service topic segues perfectly into considerations about the

time server sources you can choose to use. I'll start by saying that if you're using AWS, you almost never have a reason not to trust their time service, but I'll then add that because timekeeping is such a critical service, you might consider adding some extra redundancy in the form of other NTP sources. Should you not be using AWS, you should definitely think about which sources you have in place for redundancy.

Before continuing further, I'm going to refer you back to the chrony FAQ page [2] and the section *How can I improve the accuracy of the system clock with NTP sources?*. There, you'll find information on tweaking the time servers to which you ultimately choose to sync.

In my experience with upstream time servers that are geographically close but not all in the same country, using the public servers listed on the NTP site [4] is a very effective approach. Note the warning on the site about denial-of-service attacks:

NTP users are strongly urged to take immediate action to ensure that their NTP daemons are not susceptible to being used in distributed denial-of-service (DDoS) attacks. Please also take this opportunity to defeat denial-of-service attacks by implementing Ingress and Egress filtering through BCP38.

A link is offered for BCP38 [5] that details the clever approaches for filtering out attacks (at the network Access Control List level) that plagued NTP for a while.

```
Papa ~ # host 0.pool.ntp.org
0.pool.ntp.org has address 195.195.221.100
0.pool.ntp.org has address 185.103.117.60
0.pool.ntp.org has address 178.62.24.228
0.pool.ntp.org has address 178.79.160.57
Papa ~ #
```

Figure 2: Four NTP servers under one DNS entry.

IPv4	IPv6
There are 1849 active servers in this zone.	There are 893 active servers in this zone.
1848 (+1) active 1 day ago	894 (-1) active 1 day ago
1842 (+7) active 7 days ago	889 (+4) active 7 days ago
1820 (+29) active 14 days ago	899 (-6) active 14 days ago
1817 (+32) active 60 days ago	879 (+14) active 60 days ago
1764 (+85) active 180 days ago	828 (+65) active 180 days ago
1925 (-76) active 1 year ago	796 (+97) active 1 year ago
1882 (-33) active 3 years ago	734 (+159) active 3 years ago
2039 (-190) active 6 years ago	549 (+344) active 6 years ago

Figure 3: Europe's pool looks pretty healthy, boasting lots of clocks.

Time Works Wonders

The NTP site has a very useful page dedicated to helping you choose time servers near you [6] and how round-robin is used to iterate through a list of servers presented to an NTP server by the Domain Name System (DNS).

Figure 2 shows that a typical NTP-style hostname actually points to multiple time servers, which allows lists of time servers to be gathered easily into "pools." I'm in Europe, so I can add the pool servers to my chrony config file that are geographically close [7],

```
server 0.europe.pool.ntp.org
server 1.europe.pool.ntp.org
server 2.europe.pool.ntp.org
server 3.europe.pool.ntp.org
```

by prepending `server` to each line. This setup offers 16 IPv4 time servers; additionally, `2.europe.pool.ntp.org` offers four IPv6 clocks with which to connect.

If you're worried that the number of community-volunteered clocks will reduce over time, in Europe alone, you can see that you should have no issues in that respect (Figure 3).

However, take note of the comment on the page for European servers: "In most cases, it's best to use `pool.ntp.org` to find an NTP server (or `0.pool.ntp.org`, `1.pool.ntp.org`, etc. if you need multiple server names). The system will try finding the closest available servers for you."

The clever (NTP) protocol that assists with timekeeping is innately designed to measure response times from servers

that are geographically disparate and then compensate against the inherent network latency.

If you want to get your hands

dirtier with NTP and `chronyd` (or `ntpd`, of course), then a word of advice would be to remember that DNS entries for the clock servers (e.g., `0.pool.ntp.org` through `3.pool.ntp.org`) point to a randomized set of servers that are updated each hour, which helps to distribute load. Bear in mind, however, that if you're working in a hospital and life depends on equipment running to the correct time, you're probably going to need a different pool of servers than someone who's watching YouTube at home. Business and domestic needs might be quite different.

The End of Time

In contrast to large and cumbersome applications, your machines will thank you for running small, lightweight pieces of software to help with load and, therefore, resource availability. The excellent `chrony` definitely ticks that box.

Moreover, being able to secure your time server installation quickly and easily is a very welcome feature. With AWS as a cloud provider and a number of OSs now opting for `chronyd` over `ntpd`, it's well worth getting to grips with `chrony` sooner rather than later. ■■■

Info

- [1] chrony: <https://chrony.tuxfamily.org>
- [2] chrony FAQ: <https://chrony.tuxfamily.org/faq.html>
- [3] Setting the time for a Linux instance: <https://docs.aws.amazon.com/AWSEC2/latest/UserGuide/set-time.html>
- [4] NTP home: <http://www.ntp.org>
- [5] BCP38: <http://bcp38.info>
- [6] NTP pool time servers: <http://support.ntp.org/bin/view/Servers/NTPPoolServers>
- [7] NTP pool servers in Europe: <https://www.pool.ntp.org/zone/europe>

Author

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

at, cron, and anacron

SCHEDULING COMMANDS AND SCRIPTS



The `at` command and the related `cron` and `anacron` can help you efficiently schedule tasks, whether one-time events or jobs to be done repeatedly. *By Bruce Byfield*

Scheduling tasks is as old as Unix. Usually, it is a concern for root users, handy for making sure that jobs like backups are done regularly. However, scheduling can also be useful for regular users, if only to broadcast a message to themselves to take a break. Either way, you have three scheduling systems for your needs: `at` [1] and its related commands, `cron` [2], and `anacron` [3]. The three overlap, but

Author

Bruce Byfield is a computer journalist and a freelance writer and editor specializing in free and open source software. In addition to his writing projects, he also teaches live and e-learning courses. In his spare time, Bruce writes about Northwest coast art. You can read more of his work at <http://brucebyfield.wordpress.com>.

each has its own peculiarities for configuration and scheduling.

The at Family of Commands

`at` and its associated commands are for scheduling of a one-time event. By default, the commands can only be run by root. However, other users can also be permitted to run `at` if you create a file called `/etc/at.allow`, adding names one per line. You can also add users to `/etc/at.deny` if there are users whom you specifically do not want to use the commands, including users that

exist for administrative purposes or users that might be created by intruders, such as `guest` (Figure 1).

To schedule a command to run once, enter the command followed by a time. The time must be precise down to minutes and can be precise down to seconds using the `MMDDhhmm[.ss]` format.

```
root@nanday:/etc# cat /etc/at.deny
alias
backup
bin
daemon
ftp
games
gnats
guest
```

Figure 1: `/etc/at.deny` specifies users who are not permitted to use `at`. A companion file, `/etc/at.allow`, can also be created.

Lead Image © rawpixel, 123RF.com

You can also specify the fixed times midnight, noon, or teatime (4pm) or a specific date followed by the time. Similarly, you can specify now, today, or tomorrow, optionally adding a time after now or today. For instance, 5pm + 1 week or noon August 2 would both be valid time entries. Should you enter a time that has already passed, such as 11am today when it is 4pm, the command will be run in about five minutes.

No matter how you enter the time, running the command will start the at command prompt. Enter as many commands as you want at the prompt, and press Ctrl + D when you are finished

(Figure 2). A message is sent to the user, who receives mail from root.

at itself has a limited set of options. You can suppress mail with -m or run a list of commands in a file using -f FILE. There are also aliases for the related commands, but most users will probably find it easier to remember the full names of the commands. Here they are with examples:

- atq: Lists scheduled jobs. If used by root, everybody's jobs are listed (Figure 3).
- atrm: Removes scheduled jobs, using their job number. No confirmation dialog appears, so check job numbers using atq first.

- batch: Used instead of at to run commands when the system load levels permit. Batch accepts no time parameters, since they are irrelevant. Up to 52 batches can be set at one time. The names for batches begin with lowercase a, and end with uppercase Z. Should you have trouble running the at family of commands, run service atd status. If at is not running, you can start it with service atd start.

Running cron Jobs

The main difference between at and cron is that at is for a command to be scheduled once, and cron is for a command to be used repeatedly. In addition, a cron entry requires more information than an at entry.

Like at, the use of cron is controlled by two files – in cron's case, /etc/cron.deny and /etc/cron.allow. Both the cron commands must be created using touch before any user except root can use cron. As with at's files, cron's files list users one per line. For added security, you may want to copy the contents of at.deny into cron.deny.

cron jobs to be run are stored in a file called a crontab. Non-root users can be given their own crontab by running crontab -u USER FILE, most likely using ~/.crontab.

To add a cron job, run crontab -e to manually enter a task. The first time a user runs crontab, they are asked to select a default editor to use (Figure 4) from among those installed on the system. They can then enter a cron job, one per line (Figure 5). Root's crontab is commented, while other users' are not – presumably because non-root users are rare.

A cron job has five fields:

```
MINUTES(0-59) HOURS(0-23)
DAY-OF-THE-MONTH(1-31)
MONTH(1-12) DAY-OF-THE-WEEK(0-6)
COMMAND
```

Notice that the numbering is inconsistent: While days and months have the expected range of numbers, minutes, hours, days of the week begin at 0 instead of 1. There is no particular reason for this inconsistency except an outbreak of geekiness.

For example:

```
45 12 13 05 4 Wall 2
"System going down for maintenance NOW"
```

```
root@nanday:/etc# at noon tomorrow
warning: commands will be executed using /bin/sh
at> Wall "System going down to maintenance NOW"
at> <EOT>
job 2 at Thu May 23 12:00:00 2019
```

Figure 2: A successful command entered into at.

```
root@nanday:/var/mail# atq
2 Thu May 23 12:00:00 2019 a root
```

Figure 3: Atq lists scheduled jobs.

```
bb@nanday:~$ crontab -e
Select an editor. To change later, run 'select-editor'.
 1. /bin/nano          <---- easiest
 2. /usr/bin/jmacs
 3. /usr/bin/joe
 4. /usr/bin/jpico
 5. /usr/bin/jstar
 6. /usr/bin/nvim
 7. /usr/bin/rjoe
 8. /usr/bin/vim.basic
 9. /usr/bin/vim.gtk
10. /usr/bin/vim.tiny
Choose 1-10 [1]:
```

Figure 4: The first time you run crontab, you are asked to choose an editor.

```
1W /tmp/crontab.ZtUQe0/crontab
# Edit this file to introduce tasks to be run by cron.
#
# Each task to run has to be defined through a single line
# indicating with different fields when the task will be run
# and what command to run for the task
#
# To define the time you can provide concrete values for
# minute (m), hour (h), day of month (dom), month (mon),
# and day of week (dow) or use '*' in these fields (for 'any').#
# Notice that tasks will be started based on the cron's system
# daemon's notion of time and timezones.
#
# Output of the crontab jobs (including errors) is sent through
# email to the user the crontab file belongs to (unless redirected).
#
# For example, you can run a backup of all your user accounts
# at 5 a.m every week with:
# 0 5 * * 1 tar -zcf /var/backups/home.tgz /home/
#
# For more information see the manual pages of crontab(5) and cron(8)
```

Figure 5: Root's crontab includes helpful comments.

Table 1: Descriptive Time Strings for Cron

String	Meaning
@reboot	Run once at startup
@yearly	Run once a year, "0 0 1 1 *"
@annually	Run once a year, "0 0 1 1 *"
@monthly	Run once a month, "0 0 1 * *"
@weekly	Run once a week, "0 0 * * 0"
@daily	Run once a day, "0 0 * * *"
@midnight	Run once a day, "0 0 * * *"
@hourly	Run once an hour, "0 * * * *"

sends a message to all users about upcoming maintenance at 12:45 on Wednesday, May 13.

The first five fields do not have to be all filled out, so long as those that have an entry produce a legitimate date. For example, the day of the week, the fifth column could be left blank except for an asterisk so that the command would run on any May 13, no matter what the day of the week. In addition, each of the first five fields can be filled by a comma-separated list of entries. If you want to run a job at a regular interval, you can add an interval divided by 10 to the minutes field (for instance, */20).

You can see a list of jobs by running `cronjob -l` and remove the entire file with `cronjob -r`.

All commands are written by default to `/var/spool/cron/crontabs`, with separate entries or `crontabs` for each authorized user. `cron` also reads `/etc/crontab`, `/etc/cron.daily`, `/etc/cron.weekly`, and `/etc/cron.monthly` (which may have to be created). These directories consist simply of shell scripts that run 15 minutes after boot time or set in `/etc/sysconfig/cron` file.

However, these entries in `-etc` subdirectories are actually a redundancy, because the root user can get the same scheduling by selecting the fields to fill in the normal `crontab`. It can be confusing to use two different forms of scheduling, and the trend in recent years is not to use them at all. Instead, users can replace the five traditional fields of a cron job with the notations in Table 1.

Off and On with anacron

`cron` was designed for systems that are always running, such as servers. By contrast, `anacron` has the flexibility to run on machines that are frequently turned off and on, such as personal laptops and workstations. If a system is not turned on when an `anacron` job is scheduled, it simply runs the next time the system is booted.

Another difference between `cron` and `anacron` is that `anacron`'s jobs are stored in the `file/etc/anacronjob`. Jobs are added to `anacronjob` using a text editor. One job is defined per line, in the following format:

```
PERIOD(DAYS) DELAY(MINUTES) ID COMMAND
```

`PERIOD` is the number of days between each running of the command or else one of these tags: `@daily`, `@weekly`, or `@monthly`. The period works with the `DELAY` field, or the number of minutes after bootup that the job is run. Probably, you will want to run `anacron` jobs at least ten minutes after bootup, just to assure that no other commands are run at the same time from login scripts. Otherwise, no more precise timing is available, and when a command runs is relative to the boot time, not an absolute time by the calendar and clock, as with `cron`. The `ID` is the name of the command or script to run, the command, the actual path, and structure of the file. For instance,

```
7 10 trim.weekly 2
/bin/sh /root/trim.sh
```

runs the `trim` command for SSD drives once a week 10 minutes after bootup.

You can view scheduled `anacron` tasks with the command `ls -l /var/spool/anacron/` (Figure 6). Tasks scheduled daily, weekly, or monthly will be marked.

`anacron` is not set up with everyday users in mind. However, you can specify another `anacrontab` using `-t $~/etc/anacrontab` to specify a personal list of jobs, and `-S $~/var/spool/anacron` for a personal spool in your `~/profile`.

Choosing a System

Of the three scheduling systems, `cron` is probably the best-known. If you want to simplify your life and learn only one system, `cron` is undoubtedly the most versatile one. However, it can be needlessly complex, especially for non-root users.

If you only schedule occasional, one-time tasks, the `at` family of commands should suffice for you. However, if the exact timing of tasks is irrelevant to you, then `anacron` is ideal for regularly repeated tasks. In fact, some writers suggest that `anacron` is the scheduler best-suited to standalone laptops and workstations.

But whichever scheduler you use, you can be confident that you are using technology that has lasted several decades and that fits well with other classic Bash commands. Any problems you encounter with any of these schedulers is likely to be an obvious one, such as a problem with permissions or environmental variables.

Perhaps the greatest proof of the utility of these three schedulers is that, despite frequent attempts, desktop versions of them have never really caught on. All three are so well-designed that a GUI offers no advantage over the original command line applications. ■■■

Info

- [1] at: <https://linux.die.net/man/1/at>
- [2] cron: <https://linux.die.net/man/1/crontab>
- [3] anacron: <https://linux.die.net/man/8/anacron>

```
root@nanday:/var/spool/anacron# ls -l /var/spool/anacron/
total 16
-rw----- 1 root root 9 May 22 09:43 cron.daily
-rw----- 1 root root 9 Apr 28 07:09 cron.monthly
-rw----- 1 root root 9 May 16 09:51 cron.weekly
-rw----- 1 root root 9 May 20 09:16 trim.weekly
```

Figure 6: Use the `ls` command to view scheduled `anacron` jobs.

Too Swamped to Surf?



ADMIN Network & Security

ADMIN offers additional news and technical articles you won't see in our print magazine.

Subscribe today to our free ADMIN Update newsletter and receive:

- Helpful updates on our best online features
- Timely discounts and special bonuses available only to newsletter readers
- Deep knowledge of the new IT



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

Get deeper insights into your system with eBPF

Keen Observer

Use the eBPF in-kernel virtual machine to identify resource bottlenecks and optimize your installation.

By Mayank Sharma



eBPF [1] is a relatively new addition to the Linux kernel that takes over more monitoring, security, and networking duties from individual kernel modules. Originally called the Berkeley Packet Filter, BPF came to life in 1992 [2] in order to provide a better and optimized mechanism to filter packets.

BPF was first used as an HTTP packet filter in BSD. Several decades later, it was completely rehashed and took on new tasks. The new version of BPF is what is known as enhanced BPF or eBPF. In addition to various new features, eBPF also has a new mechanism to connect to the Linux kernel. Instead of just redirecting packets, eBPF can attach itself to any kernel event or any socket. eBPF is tightly integrated with the Linux kernel and can be used as an efficient mechanism for Linux tracing. You can also use eBPF behind the scenes on your Linux machines to discover performance issues and bottlenecks.

Get Started

eBPF requires a kernel newer than v4.4 and one that has been compiled with the `CONFIG_BPF_SYSCALL` option. Neither of these requirements should be a problem if you are using one of the mainstream distributions like Ubuntu and updating it regularly.

To experience the tracing benefits of eBPF, install the tools from BPF Compiler Collection commonly referred to as bcc [3]. Fire up a terminal in an Ubuntu installation and type:

```
$ sudo apt install bpffcc-tools 2
linux-headers-$(uname -r)
```

This command will fetch the tools as well as the kernel headers for the kernel version that you are currently using. The bcc includes over 70 tools, and under Ubuntu, they are all installed in the `/usr/sbin` di-

rectory and will have a `-bpffcc` extension (Figure 1). The tools are, in fact, Python scripts that you can edit and modify as per your requirements, provided that you know what you're doing. If you are using another distro, refer to the bcc documentation [4] for distribution-specific installation instructions.

Keep a Close Watch

I'll start by showing how to keep an eye out for new processes using the `execsnoop` tool. This tool is especially useful for tracing processes that are short-lived, which is

```

bodhi@plato:~$ cd /usr/sbin/
bodhi@plato:/usr/sbin$ ls *-bpffcc
argdist-bpffcc          javaobjnew-bpffcc      runqslower-bpffcc
bashreadline-bpffcc    javastat-bpffcc        shmsnoop-bpffcc
biolatency-bpffcc      javathreads-bpffcc    slabratetop-bpffcc
biosnoop-bpffcc        killsnoop-bpffcc      sofdnsnoop-bpffcc
biotop-bpffcc          llcstat-bpffcc         softirqs-bpffcc
bitesize-bpffcc        mdflush-bpffcc        solisten-bpffcc
bpflist-bpffcc         memleak-bpffcc        sslsniff-bpffcc
btrfsdist-bpffcc      mountsnoop-bpffcc     stackcount-bpffcc
btrfs slower-bpffcc    mysqlqslower-bpffcc   statsnoop-bpffcc
cachestat-bpffcc       nfsdist-bpffcc        syncsnoop-bpffcc
cachetop-bpffcc        nfsslower-bpffcc      syscount-bpffcc
capable-bpffcc         nodedegc-bpffcc       tcalls-bpffcc
cobjnew-bpffcc         nodelist-bpffcc       tciflow-bpffcc
cpuid-bpffcc           offcputime-bpffcc     tclobjnew-bpffcc
cpuunclained-bpffcc   offwaketime-bpffcc    tc1stat-bpffcc
criticalstat-bpffcc   oomkill-bpffcc        tcpaccept-bpffcc
dbslower-bpffcc        opensnoop-bpffcc      tcpconnect-bpffcc
dbstat-bpffcc          perllcalls-bpffcc     tcpconnlat-bpffcc
dcsnoop-bpffcc         perlflow-bpffcc       tcpdrop-bpffcc
dcstat-bpffcc          perlstat-bpffcc       tcplife-bpffcc
deadlock_detector-bpffcc  phpcalls-bpffcc      tcpretrans-bpffcc
deadlock_detector.c-bpffcc  phpflow-bpffcc       tcpstates-bpffcc

```

Figure 1: Remember that all the eBPF utilities need root privileges to run.

```

bodhi@plato: ~
bodhi@plato: ~
bodhi@plato: ~
bodhi@plato: ~$ sudo execsnoop-bpfcc
PCOMM      PID  PPID  RET  ARGS
sshd       1314 29799 0    /usr/sbin/sshd -D -R
sh         1327 1314  0    /bin/sh -c /usr/bin/env -i PATH=/usr/local/sbin:/usr/
local/bin:/usr/sbin:/usr/bin:/sbin:/bin run-parts --lsbysinit /etc/update-motd.d > /r
env        1330 1327  0    /usr/bin/env -i PATH=/usr/local/sbin:/usr/local/bin:/
usr/sbin:/usr/bin:/sbin:/bin run-parts --lsbysinit /etc/update-motd.d
run-parts  1330 1327  0    /usr/bin/run-parts --lsbysinit /etc/update-motd.d
00-header  1333 1330  0    /etc/update-motd.d/00-header
uname      1334 1333  0    /usr/bin/uname -o
uname      1336 1333  0    /usr/bin/uname -r
uname      1337 1333  0    /usr/bin/uname -m
10-help-text 1339 1330  0    /etc/update-motd.d/10-help-text
50-motd-news 1340 1330  0    /etc/update-motd.d/50-motd-news
cat        1341 1340  0    /usr/bin/cat /var/cache/motd-news
cut        1344 1340  0    /usr/bin/cut -c -80
head       1342 1340  0    /usr/bin/head -n 10
tr         1343 1340  0    /usr/bin/tr -d \000-\011\013\014\016-\037
90-updates-ava 1345 1330  0    /etc/update-motd.d/90-updates-available
cat        1346 1345  0    /usr/bin/cat /var/lib/update-notifier/updates-availab
le
91-release-upgr 1347 1330  0    /etc/update-motd.d/91-release-upgrade
lsb_release 1349 1348  0    /usr/bin/lsb_release -sd

```

Figure 2: The execsnoop utility catches an incoming SSH connection.

to say those processes that end before you can track them via the traditional process monitoring tools like `top`. Keeping an eye out for these processes that usually miss your attention might help you optimize your installation.

Fire up a terminal and type:

```
$ sudo execsnoop-bpfcc
```

The tool will now keep an eye out for new processes. To give it something to pick up, perform some action (Figure 2), such as firing up a new terminal. This action will produce the output shown in Listing 1.

As you can see from this truncated output, `execsnoop` prints one line of output for each new process. The output shows the parent process or command name under the `PCOMM` column, the PID of the process along with its parent PID, and the return value of the `exec()` function under the `RET` column, as well as the command and any arguments under the `ARGS` column.

The `-x` option can be used to include failed execution (Listing 2).

You can similarly use the `-t` option to include a timestamp column and the `-n` option to match on a name. For instance, `sudo execsnoop-bpfcc -n ssh` will only catch processes where the command matches the specified name (in this case, `ssh`).

A related tool is `opensnoop`, which enables you to trace file opens. The `open()` system call brings up files for read and

write operations, and keeping an eye on this process can reveal a lot of details about how a program works behind the scenes. It can, for instance, help you identify all the data files, config files, and log files that are associated with an app and the manner in which they are accessed by the app. If an application performs poorly, it could be because it is

improperly configured and is wasting milliseconds trying to access files that don't exist.

The `opensnoop` tool traces the `open()` system calls and prints a line for each call that's found, as shown in Listing 3.

The first column is the process ID of the process that invoked the `open()` system call. The second column, `COMM`, displays the name of the process, and the third column displays the file descriptor as it was returned by `open()`. Then comes the error value if any error code is returned by `open()`, and the final column specifies the full path of the file used in the `open()` system call.

On an active Linux installation, the output of `opensnoop` will be very difficult to follow. It'll be helpful if you filter the output with `grep` to make sure you catch hold of what you're looking for. Another good idea is to filter using the process ID of the process that is of interest to you with the `-p` option or the `-n` option to filter on process name, such as:

```
sudo opensnoop-bpfcc -n 2
gnome-shell
```

Listing 1: execsnoop Output

```

$ sudo execsnoop-bpfcc
PCOMM      PID  PPID  RET  ARGS
gnome-terminal 14483 1      0    /usr/bin/gnome-terminal --window
gnome-terminal. 14486 14483 0    /usr/bin/gnome-terminal.real --window
bash         14492 2490  0    /bin/bash
lesspipe     14493 14492 0    /usr/bin/lesspipe
basename     14494 14493 0    /usr/bin/basename /usr/bin/lesspipe
dirname      14496 14495 0    /usr/bin/directory /usr/bin/lesspipe
dircolors    14497 14492 0    /usr/bin/dircolors -b

```

Listing 2: the execsnoop -x Option

```

$ sudo execsnoop-bpfcc -x
PCOMM      PID  PPID  RET  ARGS
nautilus    15815 15781 -2    /usr/local/sbin/net usershare info
nautilus    15815 15781 -2    /usr/local/bin/net usershare info
nautilus    15815 15781 -2    /usr/sbin/net usershare info

```

Listing 3: Tracing open() Calls

```

$ sudo opensnoop-bpfcc
PID  COMM          FD  ERR  PATH
1    systemd       16  0    /proc/372/cgroup
3642 gnome-shell    19  0    /proc/self/stat
14769 DNS Resolver #9 57  0    /etc/hosts

```

Listing 4: tcpconnect

```
$ sudo tcpconnect-bpfcc
```

PID	COMM	IP	SADDR	DADDR	DPORT
14786	Socket Threa	4	192.168.0.11	172.217.167.228	443
14786	Socket Threa	4	192.168.0.11	82.103.136.226	80
16530	wget	4	192.168.0.11	81.3.27.38	443
16530	wget	4	192.168.0.11	178.124.134.106	443

Network Inspector

Another useful eBPF tool is `tcpconnect`, which enables you to inspect active TCP connections by watching all `connect()` system calls. The `tcpconnect` tool prints all TCP connections, along with their source and destination addresses (Listing 4).

The first and second column as usual list the process ID and the name of the

process that is called `connect()`. The third column specifies whether you're using IPv4 or IPv6. The fourth and fifth columns are the source IP and destination IP for the connections, and the last column is the port number at the destination address.

One common option is `-U`, which appends the UID of the processes to the output. You can then use it along with

the `-u` option to filter the output using UID, such as:

```
sudo tcpconnect-bpfcc -Uu 1000
```

Another tool that is useful for debugging TCP processes is `tcpaccept` (Figure 3), which traces the `accept()` system call. The overhead of `tcpaccept` is negligible, and although `netstat` can do the job of both `tcpconnect` and `tcpaccept`, the two eBPF tools are more versatile. You can even use the `-p` option with both the tools to watch specific processes, such as:

```
sudo tcpaccept-bpfcc -p 14786
```

Peak Performance

One of the best uses for the eBPF tools is to help you tune your system for maximum performance by identifying and removing bottlenecks at various levels. You can begin by using the `runqlat` tool to chart how long threads spend waiting in the CPU run queues. It prints a summary of the scheduler run queue latency in the form of a histogram, as shown in Figure 4.

Then there's the `biolatency` tool, which comes in handy to visualize the latency of block device I/O. The `biolatency` tool keeps track of the elapsed time from when a device is called to its completion. Like `runqlat`, this tool will also print a histogram once it ends, either manually or after a specified duration. A typical invocation will look like:

```
sudo biolatency-bpfcc -D 6 2
```

The `-D` option instructs `biolatency` to print separate information for each block device. The first numeric value is the time interval for printing each summary, whereas the second numeric value informs `biolatency` of the total number of times it should collect information, after which point `biolatency` will automatically exit. Therefore, the previous command instructs `biolatency` to print the first histogram after 6 seconds of invoking the tool and another after another 6 seconds.

In addition to devices, there are also several tools for tracing filesystems. There's `ext4slower` for EXT4 filesystems, `xfsslower` for XFS, `btrfs slower` for BTRFS, `nfs slower` for NFS and `zfs slower` for ZFS file system. These tools will time the common filesystem operations

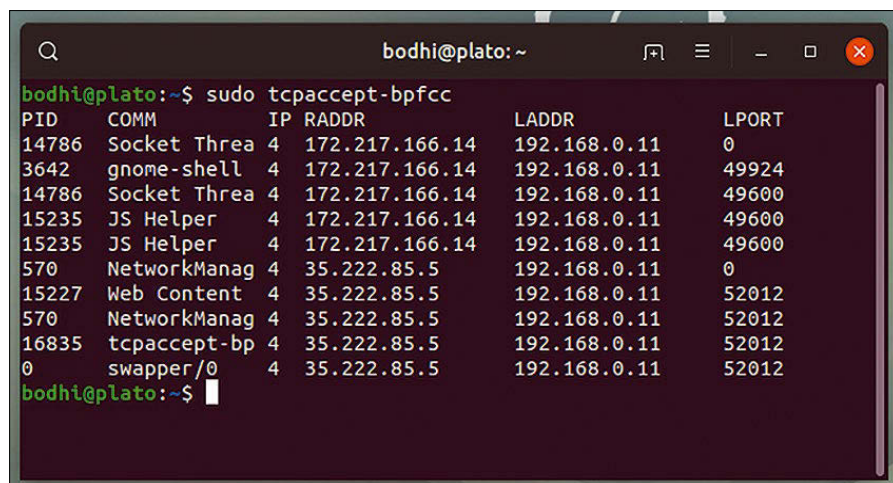


Figure 3: Note that `tcpaccept` will only trace successful TCP `accept()` calls and will not list attempts to connect to closed ports.

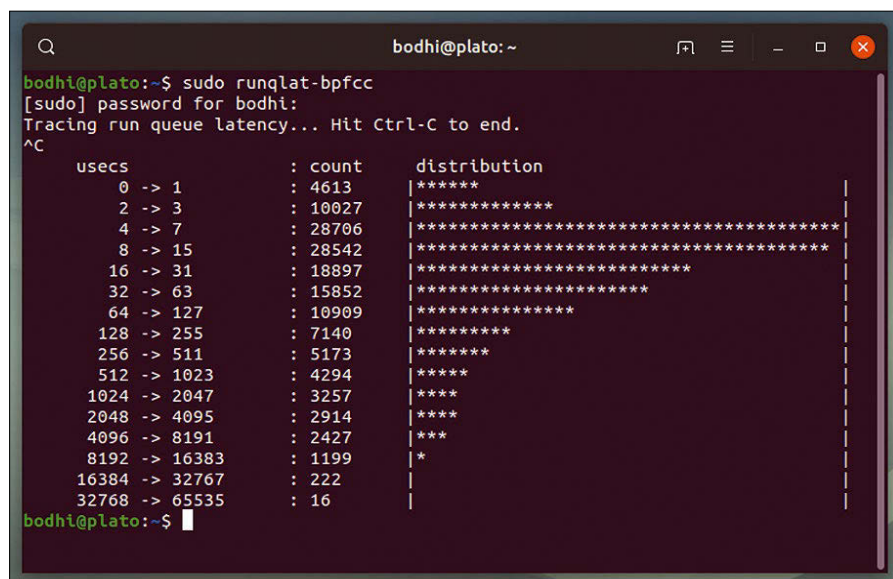


Figure 4: The `runqlat` tool helps chart the time that was lost while the CPU was busy elsewhere.

Listing 5: Specifying a Threshold

```
$ sudo ext4slower-bpfcc 100

Tracing ext4 operations slower than 100 ms

TIME          COMM          PID  T  BYTES  OFF_KB  LAT(ms)  FILENAME
00:23:01      systemd-journa 288  S  0      0      128.92   user-1000.journal
00:24:31      journal-offlin 288  S  0      0      132.98   system.journal
00:24:31      journal-offlin 288  S  0      0      104.47   user-1000.journal
```

and print a list of those that exceed a defined threshold. By default the threshold is set at 10ms, but you can customize it by specifying one manually (Listing 5).

The command in Listing 5 will display all filesystem operations that are slower than 100 ms. It measures the time it takes from when an operation is called from the virtual filesystem to its completion and flags it if it exceeds the specified threshold. This tool is ideal for picking up performance issues caused by slow disk I/O at the filesystem level. It is a lot better than statistics plotted by popular performance monitoring tools, since they depict the performance

of the disk, when in fact the bottleneck can also be due to the inability of the filesystem to respond to the requests flooding in.

We've only touched upon some of the eBPF tools that are at your disposal to trace and inspect various areas of your installation. Remember, however, that just because you have the performance measurement tools, it doesn't mean that you'll be able to streamline the performance of your box. Interpreting the results of the trace requires a fair bit of understanding of how Linux works and its internals. So make sure you invest some time reading up about the internals of the Linux kernel before you begin to uti-

lize these tools to chip away milliseconds and optimize your installation.

Also know that eBPF has a greater mandate than just tracing. Thanks to its architecture, it can also play a role in system security. It can be used to monitor and detect intrusions and may even become the de-facto means for enforcing firewalls in Linux. ■■■

Info

- [1] eBPF in the Linux Kernel: <http://www.brendangregg.com/ebpf.html>
- [2] "The BSD Packet Filter: A New Architecture for User-level Packet Capture" by Steven McCanne and Van Jacobson: <http://www.tcpdump.org/papers/bpf-usenix93.pdf>
- [3] bcc Project: <https://github.com/iovisor/bcc>
- [4] bcc Installation: <https://github.com/iovisor/bcc/blob/master/INSTALL.md>

Author

Mayank Sharma is a technology writer and you can read his scribbles in various geeky magazines on both sides of the pond.

A Webzine for High-Performance Computing Specialists



ADMIN
Network & Security

If you work with high-performance clusters, or if you're ready to expand your skill set with how-to articles, news, and technical reports on HPC technology.

admin-magazine.com/HPC

PIRATEBOX
Handy anonymous file server for parties and meetings

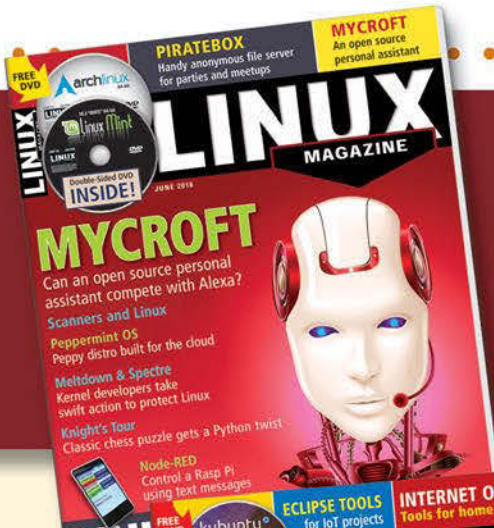
MYCROFT
An open source personal assistant

Linux MAGAZINE
JUNE 2018

MYCROFT
Can an open source personal assistant compete with Alexa?
Scanners and Linux
Peppermint OS
Peppy distro built for the cloud
Meltdown & Spectre
Kernel developers take swift action to protect Linux
Knight's Tour
Classic chess puzzle gets a Python twist

Node-RED
Control a Rasp Pi using text messages

FREE DVD
ArchLinux
Double-Sided DVD INSIDE!



Linux MAGAZINE
JULY 2018

ECLIPSE TOOLS
for IoT projects

INTERNET OF THINGS
Tools for home automation

madddog: A brief
Resetter: Peppermint OS
Audacity: Distro
Growth: Designing a vector graphics

WWW.LINUXMAGAZINE.COM



Linux MAGAZINE
AUGUST 2018

INTERNET OF THINGS
Tools for home automation

Bluetooth Tricks
Track movements within your house

Linux on Old Hardware
Tablets on Linux
Configure graphics settings with xsetwacom

Raspberry Pi 3 B+
Does the world need a new Rasp Pi?

Python Gambling
Simulating games of chance

Logism
Design a digital circuit



Linux MAGAZINE
AUGUST 2018

MAPPING TOOLS
Create and edit digital maps with QGIS and QMapShack

4 Open Source Microblogging Tools
Kubantu Up Close
Explore the new 18.04 release

9 Handpicked Raspberry Pi Projects
Caddy
HTTPS server out of the box

ODROID-C2
This single-board system is twice as fast as the competition

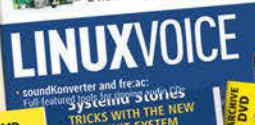
Ogg Vorbis
Free format for audio files

A Python Script
That solves the Chinese ring puzzle



Linux VOICE
FOSSPicks
GnuCash 3
XLEngine

Tutorials
FOSSPicks
GnuCash 3
XLEngine



Linux VOICE
FOSSPicks
GnuCash 3
XLEngine

Tutorials
FOSSPicks
GnuCash 3
XLEngine



Linux MAGAZINE
SEPTEMBER 2018

Systemd
Getting more from the elusive new Linux init system

10 Top PDF Readers

Arduino Programming for Open Hardware Projects

Homegrown Facial Recognition
Build facial recognition into your own Python scripts

Maker Tricks
Monitor a beehive with a Raspberry Pi

Cover Your Ass(ets)
Back up your system files with CIVA

FOSSPicks
Cutter Disassembler
Chibi MIDI Editor



Linux VOICE
FOSSPicks
GnuCash 3
XLEngine

Tutorials
FOSSPicks
GnuCash 3
XLEngine



Linux MAGAZINE
NOVEMBER 2018

NET CORE
Up close with the classic Linux version control system

The benefits of dotnet on Linux

Big Microsoft's flagship work on Linux

Little-known OS to compete with BSD?

and-mapping tools
visualize decisions and organize your best ideas

Electron Framework
Cross-platform apps with JavaScript



Linux MAGAZINE
OCTOBER 2018

PRIVACY
Snoopers and protect identity

your music player with free firmware

ers
organize all your news sources

ces for your Rasp Pi

Ready to Go?
Smart queries in the powerful Go language



Linux VOICE
FOSSPicks
GnuCash 3
XLEngine

Tutorials
FOSSPicks
GnuCash 3
XLEngine



Linux MAGAZINE
NOVEMBER 2018

NET CORE
Up close with the classic Linux version control system

The benefits of dotnet on Linux

Big Microsoft's flagship work on Linux

Little-known OS to compete with BSD?

and-mapping tools
visualize decisions and organize your best ideas

Electron Framework
Cross-platform apps with JavaScript



Linux MAGAZINE
DECEMBER 2018

INNOVATIVE DISTRO
These hidden gems: simpler enlightenment, anonymous surfing even onware
UI designers get creative with KDE
Lockdown security focus on isolation
cheatsn
Syntax tips at your fingertips
Upribox 2
Use a Rasp Pi to filter ads and trackers

Git Workshop
Working with remote repositories

Purism Librem
Linux laptop special security features

FOSSPicks
Vivaldi Podcast Manager
MIBSE: Old School BBS
Lava Online Editor

Tutorial
1005 Fronts



Linux VOICE
FOSSPicks
GnuCash 3
XLEngine

Tutorials
FOSSPicks
GnuCash 3
XLEngine



Linux MAGAZINE
FEBRUARY 2019

CUSTOMIZE THE BOOT MENU
Clean up your startup for clarity and fewer mistakes

PCIe SSDs
Make the storage fit your hardware

IoT Tricks
Store data in memory with Redis

Exploring Ubuntu 18.10
Pulse Sensor
Measure your heartbeat with a Raspberry Pi

EncryptPad
Text editor with easy encryption



Linux MAGAZINE
MARCH 2019

VirtualBox Hacks
Save time and extend the power of your virtual machines

Scapy
Automate packet analysis with Python

Put a Recording Studio on a Raspberry Pi

Elementary OS
Elegant Linux with an ambitious vision

Create a Cartoon
With open source tools

Tools for Writers
Find words and organize your thoughts

PDF Tricks
Clean up blotches and fading text



Linux VOICE
FOSSPicks
Vivaldi Podcast Manager
MIBSE: Old School BBS
Lava Online Editor

Tutorial
1005 Fronts



Linux VOICE
FOSSPicks
GnuCash 3
XLEngine

Tutorials
FOSSPicks
GnuCash 3
XLEngine



Linux VOICE
FOSSPicks
GnuCash 3
XLEngine

Tutorials
FOSSPicks
GnuCash 3
XLEngine



Linux MAGAZINE
MARCH 2019

VirtualBox Hacks
Save time and extend the power of your virtual machines

Scapy
Automate packet analysis with Python

Put a Recording Studio on a Raspberry Pi

Elementary OS
Elegant Linux with an ambitious vision

Create a Cartoon
With open source tools

Tools for Writers
Find words and organize your thoughts

PDF Tricks
Clean up blotches and fading text



Linux VOICE
FOSSPicks
GnuCash 3
XLEngine

Tutorials
FOSSPicks
GnuCash 3
XLEngine



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

Expand your Linux skills with:

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

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

Subscribe now!
shop.linuxnewmedia.com/subs

GET IT NOW!
FAST DELIVERY
WITH OUR PDF
EDITION





Integrated client/server solution

Class Reunion

Intranets with multiple servers and services require careful configuration. With Karoshi Linux, even complex structures can be set up in no time at all. *By Erik Bärwaldt*

Educational institutions often require complex IT infrastructures that cover all areas of school life. These not only include learning and educational software for the students, but also subject-specific applications and administration. In order to reconcile all requirements and areas (e.g., to be able to use existing proprietary software), cross-platform interfaces must be implemented.

Above all, the software must be easy to use, because often the school network is not managed by IT administrators, but by teachers with limited IT knowledge, who were given responsibility almost incidentally. The UK-based Linux Schools project [1] is tackling this problem with an all-in-one solution named Karoshi [2], which has proven its value in British schools for almost two decades now.

Requirements

The current Karoshi v12.0 server is based on Ubuntu 18.04 LTS. This is also true of the associated client, which, however, has only reached version 6. Both images are available exclusively for 64-bit systems. The hardware requirements for a test environment are just 512MB RAM and about 10GB free disk space. The client ISO

image weighs in at about 3.6GB, while the server only needs about 1.7GB. Both images are hybrid images that also work on USB sticks.

However, production use requires more powerful hardware. The developers recommend a minimum of one dual-core CPU, 4GB RAM (maximum: 64GB), and 10GB mass storage, plus 500MB capacity per student or user. Since Karoshi also supports other subordinate central computers in addition to the main server, the capacities can be spread among the different machines depending on the application scenario. The installation requires wired Internet access; the graphical installer does not offer a setup dialog for a WiFi network.

Ready to Rumble

After transferring the ISO to a DVD or USB stick, you can boot Karoshi Linux for the install. The live system loads a lean Xfce desktop in which the Ubuntu Ubiquity installer launches automatically. It transfers the system to the computer in a few steps.

After rebooting, you first need to agree to the free AGPL license. The dialog for the initial installation of the server services then starts automatically. From here on, you need working network access.

First you have to decide whether the system you are installing will be the primary server or an additional machine. The routine lets you restore an existing domain controller from another server or to reconstruct from a backup archive. If you decide on a new setup, the dialog asks you what the server's purpose will be. The three alternatives here are *Education*, *Business*, and *Home*. The Karoshi server can thus also be used on your home network or in a small business; the installation routine automatically adapts the server services to match.

In the next step, the installation wizard queries the physical network interface and configures the netmask and DNS server based on the existing IP addresses for the interface. If there are several interfaces in the system, make sure that you select the correct interface. Karoshi also recognizes WiFi interfaces, but does not provide setup for them.

The wizard also prompts you for the IP address of the *Gateway* interface in the network access dialog (i.e., your gateway to the active intranet).

The installation routine then restarts the system and sets up a Samba server in a largely automated process. Only the data necessary for the domain and the matching authentication passwords need

Lead image © Author: 123RF.com

to be entered. Thanks to the Samba domain, the Karoshi server also acts as a central computer in heterogeneous environments where some clients run Windows.

Web Administration

In the last step of the initial setup, you enter a username and a password for the server's administrative web interface. You can now configure it via a completely web-based interface. The routine sets up a matching icon on the desktop.

When first enabled, the web interface launches a wizard that helps you configure various settings. However, you might want to skip the wizard, since the web interface covers the wizard's feature set completely anyway, and a system update makes more sense as the first step.

In the main window, select *System* in the *Update Web Management* group. Then press the *Check for updates* button in the window's right pane. Karoshi now displays a protocol view in which the individual scripts are retrieved.

After completion, a list with the available updates is displayed at the same location (Figure 1). Click *Apply all Patches* in the top right corner, and the system will set up the updates. Then restart the system and enable web administration by pressing the *Web Management* button on the desktop.

Role Play

The group view appears on the window's left side, the corresponding contents on the right side in an interface that looks a bit old-fashioned. You can now start setting up the server and the desired services. To do this, you can assign different roles to the server; Karoshi processes individual scripts for each role and sets up the services in just a few steps. Time-consuming manual configuration at the prompt is not needed.

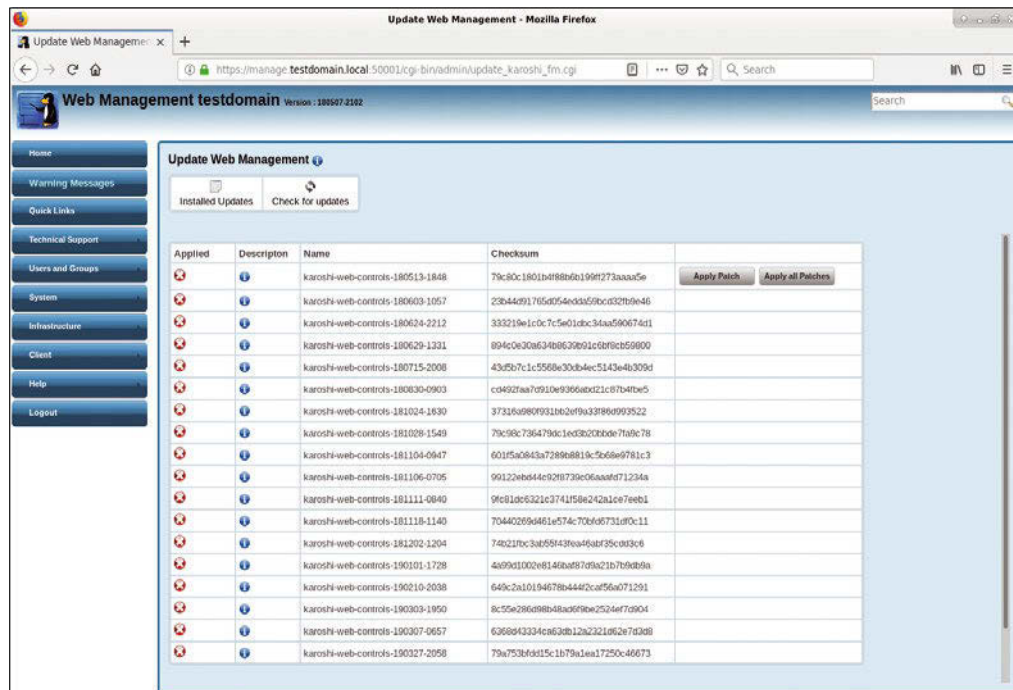


Figure 1: Karoshi server updates are best handled with the web interface.

To view a server's current roles, first click on the *System* group, and then on the *Show server* option on the left in the web interface. A list of all the servers available on the intranet with their respective roles now appears on the window's right side. In the *Server roles* column, the primary server is always the *Domain controller*, the server for *Users and groups*, and the *File server*.

When you click on the green monitor icon in the *Add Role* column to the right of each active server, a list of available

server services appears. Clicking on the monitor icon lets you select from a list with more than 20 services. The choice extends from backup, print, or email servers through to CMS, e-learning, and project management services. If you mouse over the monitor icon, Karoshi displays a tool tip for each service.

The tool tips also show you which module dependencies exist and let you know if a service (such as the backup server) cannot be installed on the main system (Figure 2).

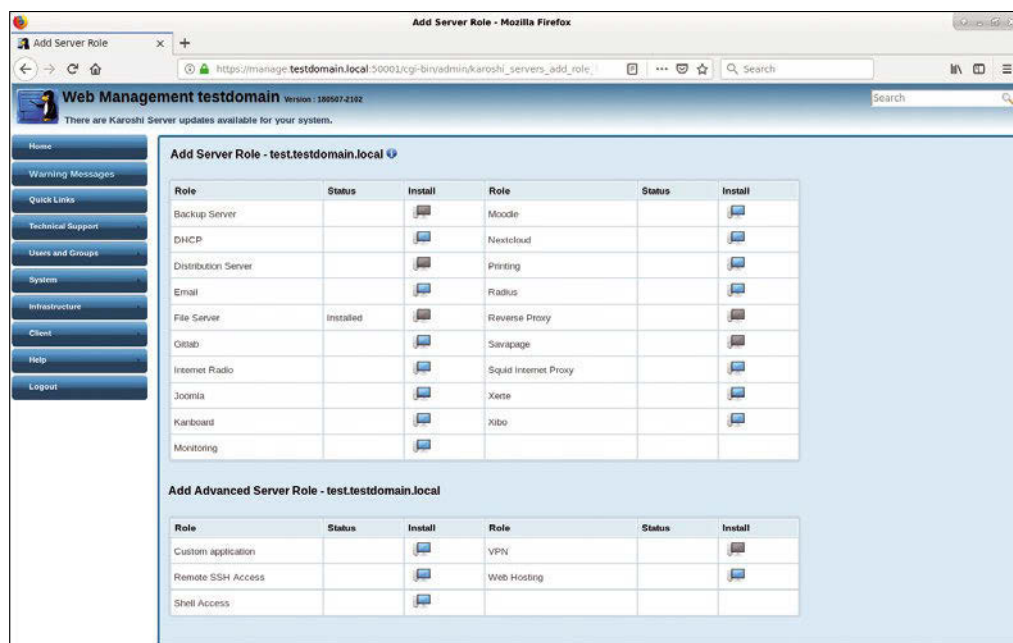


Figure 2: Install additional services with a mouse click.

To install a service, simply click on the monitor icon. Then Karoshi runs a script that sets up the service and configures it in a few interactive steps. After completing a dialog, click on the *Submit* button. When the procedure is complete, the system automatically adds the service to the list of *Server roles*.

If you then click on the dialog for adding a new server role again, the newly activated service is shown as *Configured* in the *Status* column. If the backup service cannot be set up on the main server, a link to the assigned backup server appears in the *Backup* column.

You can modify the configuration of the configured services in further dialogs. The *Backup* group in the main window offers you the option of setting up the backup parameters in the list view on the left. You can adjust printer queues in the *Printers* group on the left in the main window.

To disable services again, click the monitor icon in the *Remove role* column. Then all active and deletable services on the respective server are listed together with a message stating that existing data must be backed up before deletion. After pressing *Delete*, you have to confirm the deletion process by entering a numerical code; only then will the server role be removed.

Additional Packages

In addition to the application-dependent server services, Karoshi Linux can be kit-

ted out with numerous other useful applications. These include the free SOGO [3] groupware, the Dovecot [4] email server, the Moodle [5] learning and course management software, the Joomla CMS [6], and the Xerte [7] e-learning development environment.

Xibo [8] also provides a digital signing solution. Karoshi Linux also integrates the ownCloud [9] storage platform and the GlusterFS [10] distributed filesystem. The latter combines mass storage capacities on several servers to create a single filesystem. The Squid proxy server [11] and E2guardian [12] provide security on the network.

Group Dynamics

As the next step, it makes sense to create groups and users in order to simplify administration, especially for larger infrastructures. In the main window on the left, look for the *Users and Groups* category; you will find the detailed settings in the corresponding submenu.

Start by creating new groups, or manage existing ones, via the Group Management dialog. After clicking on this, Karoshi displays a list of existing groups; this will primarily mean the system groups. A horizontal buttonbar at the top of the window lets you manage existing groups, create new ones, or delete ones you don't need (Figure 3).

After a click on *New Primary Group*, a small dialog for creating a new group appears. When you get there, enter a

name, a description, and – if several servers are active on the intranet – the name of the server. Then press *Submit* to create the group.

Next, proceed to create new user accounts using *Add Users* in the *Users and Groups* category, and assign the users to individual groups. In the tabular *Group Management* view, you can then view all the associated users in the *Members* column of the respective group.

From a Distance

As a rule, you will not want to manage the Karoshi server directly on the system's desktop, but administer the services from a workstation on the intranet. It does not have to use Linux. To do this, call up the address `https://<Server-IP>:50001` in the remote computer's web browser.

Since access to the Karoshi server is exclusively encrypted via HTTPS, a security warning appears the first time you try to access it in the web browser; you need to set up an exception for this. Then authenticate against the server in the usual way with your username and password.

Division of Labor

Since Karoshi lets you distribute the server services across several computers on the intranet, you may need to assign maintenance tasks to several administrators, especially for larger infrastructures.

Even in schools, several teachers might manage the different server systems. For this purpose, the primary admin creates additional administrator accounts that fill three roles under Karoshi. In addition to primary administrators, there are conventional admins and technicians.

Primary administrators have access to the entire feature set and can also make other users administrators. While traditional admins have access to the entire system, they are not allowed to grant or revoke administrator privileges to other users. In the role of technician, users only have access to selected administration areas.

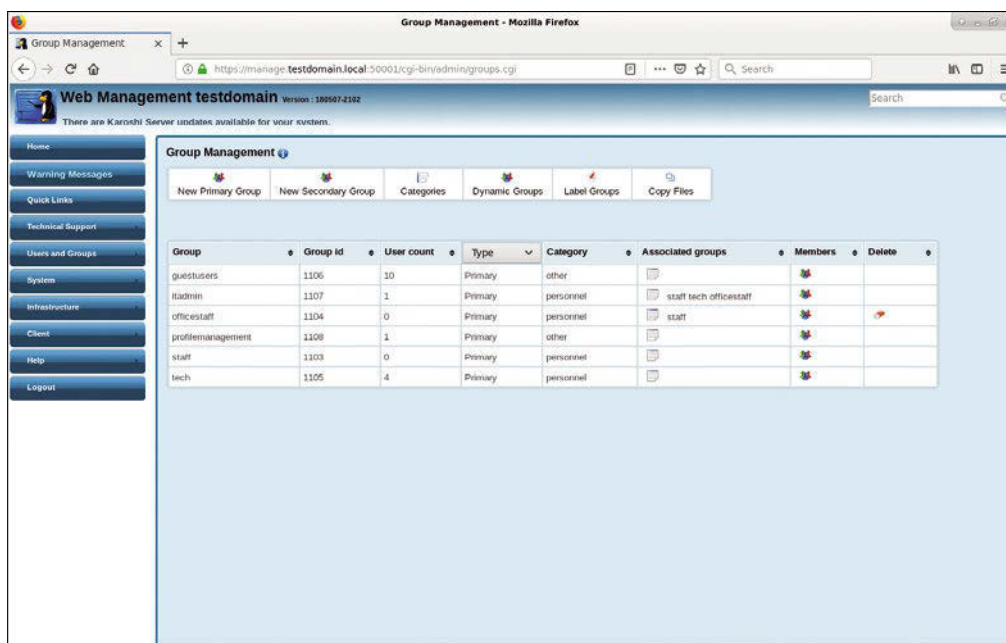


Figure 3: You can also manage groups in the web interface.

Clients

The client for your workstations can also be found on the project site. Since the developers have coordinated the client and server, the client does not require any complicated network configuration work. After booting the client from the DVD, select in the GRUB boot menu whether you want the system to boot in live mode or to complete the installation directly.

The live system opens a simple Xfce desktop with some icons on the desktop; the *Karoshi Setup* starter lets you configure the system. If you decide to install the software, a lean Ubiquity installer is fired up and quickly bundles the operating system onto your hard disk.

After a restart, the configuration routine, which opens a connection to the router, is automatically loaded. In addition, the system tries to directly connect to the domain controller; it should also be logged onto the intranet at this point. The intranet connection must be via Ethernet, however. WiFi access did not work in our lab. Although the routine

detects the WiFi interface and also sets up the corresponding kernel module, it does not offer a dialog for entering the SSID or the WPA2 key.

The client setup routine prompts you for the username and password of a user already created on the server, as well as some server data during the initial configuration. You must have created at least one other user on the Karoshi server in addition to the administration account. After entering the corresponding access credentials, the client routine configures access to the server, restarts the system, and logs on to the server. The client desktop shows more icons, because now the Samba shares also appear.

When creating the client user on the server, you will want to carefully assign the appropriate groups to avoid clients gaining access to folders that later on will not be intended for them. You can set these options in the *Members* column of the *User and Groups* | *Group Management* dialog, where you set the desired configuration for each group member.

The administrator can also see which users are logged on to a server using the *Client* | *Client connections* option in the configuration dialog. After selecting a server, the system lists the currently active computers. On the Linux client itself, you can open the server's web configuration with the *Karoshi Management* starter on the desktop.

Software

Both variants of the distribution use the Synaptic graphical front end for package management. If you call this tool on the server, a message will first appear warning you about updating the server via Synaptic and pointing to the corresponding *Updates* | *Update Server* option in the web-based configuration interface. Synaptic should only be used to install additional packages and programs. More than 60,000 packages are available after updating the sources.

The client integrates numerous additional repositories into Synaptic that do not only contain software by the Linux Schools project. This explains why the

FIND THE RIGHT TOOL!

Check out our new Linux Administration corner!

<http://www.linux-magazine.com/administration>

SPONSORED BY



Linux
Professional
Institute

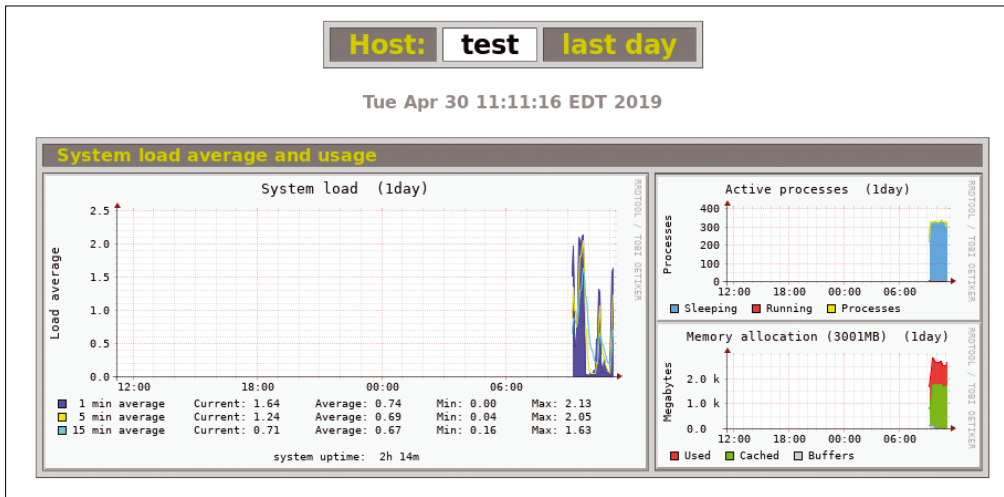


Figure 4: Monitorix gives you a quick overview of the load on the Karoshi server.

front end lists more than 62,000 packages. The client can also be updated using traditional update methods without disrupting services.

The client already offers a solid basic selection of programs for daily use. Besides LibreOffice, Firefox, Thunderbird, and Gimp, it includes the Freeplane mindmapper; in the *Graphics* category, it includes Scribus, Simple Scan, FreeCAD, and Blender. The *Internet* group includes FileZilla, FTP, and SFTP programs, and the Wireshark analysis tool.

The *Multimedia* submenu also offers a huge selection. Besides the Brasero and Xfburn burning programs, the record-MyDesktop screencast tool, the Ardour digital audio workstation, the OpenShot Video Editor, and the VLC media player are available. In the *Accessories* submenu, you will also find a virtualization environment in the form of VirtualBox.

Watchful

Karoshi Linux offers a variety of other features, such as remote client installa-

tion on multiple computers and sophisticated monitoring features. You will find the system displays in the web interface's various dialogs.

The *Infrastructure* | *System monitoring* menu provides meaningful graphics for a quick overview of the load on numerous system components. Its underpinnings are the free Monitorix tool, which Karoshi automatically dumps onto the storage medium during the server install (Figure 4).

You will also find an *Assets* tab in the same menu that you can use to keep an inventory of all your intranet components, including video projectors and whiteboards. The same menu also provides information about the DHCP and DNS server, as well as the ARP status.

In the Client menu, there is also the possibility to enter the client locations. Clients do not only include computer systems on the intranet, but also peripheral devices, such as printers controlled by the server via CUPS.

Server	Zone	Role
test.testdomain.local	Internal	Domain Controller Users and Groups File Server

Figure 5: Server Information generates some useful load tables.

Logging

The lists in the *System* | *Server information* menu and in the *Event logs* submenu are not very colorful; the emphasis is on clarity here. If you have multiple servers on your intranet, you will need to select the lists separately by device.

While the event logs provide information about the possible causes of errors, *Server Information* provides details of the mounted file-systems and the server utilization by the individual processes. Particularly on lower-performance servers with just a few CPU cores, high

workloads can indicate problematic applications (Figure 5).

Conclusions

With Karoshi Linux, even newcomers and part-time administrators can set up various server systems without weeks of training and manage a complete intranet. The British developers have not only oriented the system on the requirements of the education sector, but have also considered small companies, freelancers, and private home networks. The Linux client to match the server can also be used in computer racks to quickly create a functional IT infrastructure that meets school requirements without the admin having to laboriously install special programs. ■■■

Info

- [1] Linux Schools project: <https://www.linuxschools.com/forum/index-main.php>
- [2] Download Karoshi Linux: <https://sourceforge.net/projects/karoshi/files/?source=navbar>
- [3] SOGo: <https://sogo.nu>
- [4] Dovecot: <https://www.dovecot.org>
- [5] Moodle: <https://moodle.org/?lang=en>
- [6] Joomla!: <https://www.joomla.org>
- [7] Xerte: <https://www.xerte.org.uk/index.php?lang=en>
- [8] Xibo: <https://xibo.org.uk>
- [9] ownCloud: <https://owncloud.org>
- [10] GlusterFS: <https://glusterfs.org>
- [11] Squid: <http://www.squid-cache.org>
- [12] E2guardian: <http://e2guardian.org/cms/index.php>

Complete Your Open Source Library with Archive DVDs!

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

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



Order Your DVD Now!
shop.linuxnewmedia.com

The sys admin's daily grind: Mi Flora sensors

Salad Grower

Columnist Charly Kühnast recently attached Mi Flora humidity sensors to his potted plants. At first, they only transmitted junk on Bluetooth, but armed with the right tools and a Rasp Pi, Charly now reaps a rich harvest of data. *By Charly Kühnast*

A long time ago, I wrote about my little Pomodo Pi project [1] in this magazine. It involved me monitoring my tomato plant's soil humidity and watering the plants when there was a risk of them drying out by automatically opening a solenoid valve when the humidity dropped below a certain value. The Vegetronix sensors I still use for this are high quality and durable, but, unfortunately, they need an extra A/D converter and wiring them involved some tinkering.

Meanwhile, I have bought some humidity Mi Flora plant sensors (Figure 1). Their manufacturer, Xiaomi, envisages sending the data to a smartphone app, but I never bothered installing it; instead I pick up the data directly via Bluetooth.

I run the whole thing on a Raspberry Pi, but of course any Linux PC equipped with Bluetooth hardware will do. Software-wise you need the *bluez*, *python3*, and *python-pxpect* packages, which are quickly installed.

Recording the Weather with Bluetooth

The next step is to scan the environment for Bluetooth transmitters. This can usually be done with the command

```
sudo hcitool scan
```



Figure 1: The Mi Flora sensor, which uses Bluetooth, is a spin-off product from the Chinese smartphone gadget market.

but since the sensor is capable of low-energy transmission, I did the scan by typing:

```
sudo hcitool lescan
```

The result was a list of MAC addresses and the short names of nearby devices that can speak Bluetooth Low Energy. Hey presto, one of them was my sensor:

```
[...]
C4:7C:8D:6A:5E:17 Flower care
```

Now I have the MAC address and want to see if the sensor is sending any data:

```
sudo gatttool -Z
--device=C4:7C:8D:6A:5E:17 -Z
--char-read -a 0x35
```

The feedback doesn't need to mean anything to me at first glance:

```
Characteristic value/descriptor: aa bb -Z
cc dd ee ff 99 88 77 66 00 00 00 -Z
00 00 00 00
```

The important thing is that there is a response at all, because on GitHub there is a small tool that can interpret the data from the sensor. I cloned it to my current directory, as follows:

```
git clone https://github.com/
open-homeautomation/miflora.git
```

The package contains a small Python script by the name of *demo.py*, which reads and displays different measured values from the sensor. Now let's run it:

```
python3 /home/pi/miflora/
demo.py--backend gatttool poll -Z
C4:7C:8D:6A:5E:17
```

Voila! The output looks good:

```
Getting data from Mi Flora
FW: 3.2.1
Name: Flower care
Temperature: 23.0°C
Moisture: 40
Light: 193
Conductivity: 247
Battery: 100
```

The humidity is given as a percentage and the luminous intensity in lux. The Conductivity is shown in microsiemens – it indirectly says something about the nutrient content of the soil.

That's the full extent of my electronic green thumb (at the moment, Figure 2), and I'm already looking forward to the grand opening of my salad bar. ■■■

Info

[1] "Pomodo Pi" by Charly Kühnast, *Linux Magazine*, issue 177, August 2015, [http://www.linux-magazine.com/Issues/2015/177/Charly-s-Column-PomodoPi/\(language\)/eng-US](http://www.linux-magazine.com/Issues/2015/177/Charly-s-Column-PomodoPi/(language)/eng-US)

Author

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

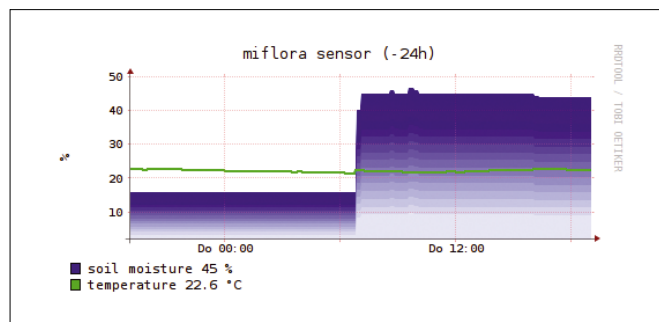
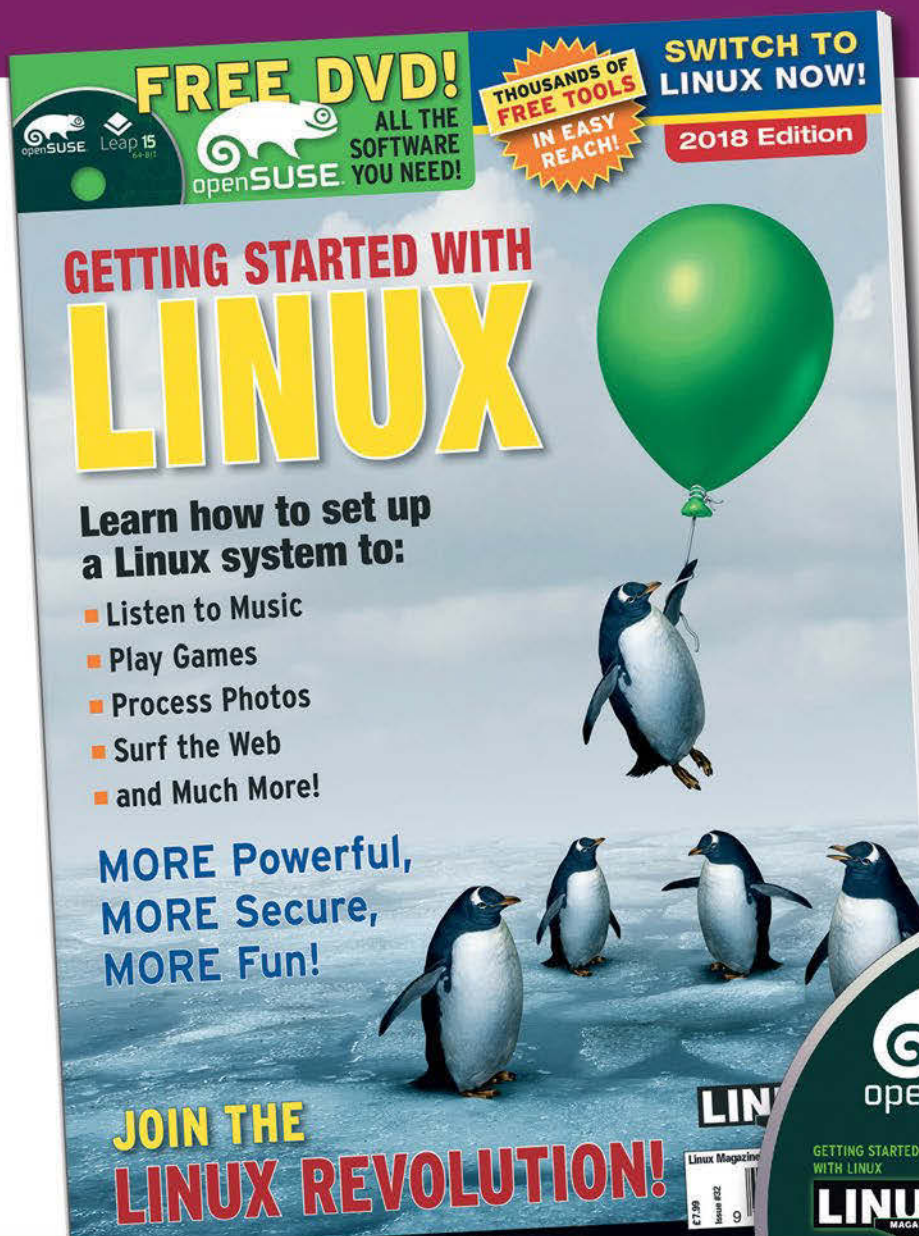


Figure 2: 24 hours in the life of a potted plant under Charly's care.

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

This single issue shows beginners how to:



- Install Linux
- Download and install free software for your Linux system
- Create documents and spreadsheets
- Play games
- Surf the web
- Process photos
- Play music and videos
- and much more!

HELP OTHERS JOIN THE LINUX REVOLUTION!

ORDER ONLINE:
shop.linuxnewmedia.com/specials

Go program finds photos with nearby GPS coordinates

In the Hood

Every photo you take with your mobile phone stores the GPS location in the Exif data. A Go program was let loose on Mike Schilli's photo collection to locate shots taken within an area around a reference image.

By Mike Schilli

Just recently, my favorite restaurant in San Francisco, Chow, shut down unexpectedly. On top of the traumatic experience of having to find a new eatery, I was overcome by the desire to find old photos of the place from the good old days on my mobile phone. But how? I sure didn't tag them, but who does, anyway? Having said that, every cell phone photo contains GPS information, and the phone's photo app can group the photos as dots on a map.

Of course, over the years, I had outsourced the photos to other media. Not to worry, my new favorite programming language, Go, comes with image processing routines, prompting me to browse my photo collection for photos taken in or near the restaurant.

Author

Mike Schilli works as a software engineer in the San Francisco Bay area, California. Each month in his column, which has been running since 1997, he researches practical applications of various programming languages. If you email him at mschilli@perlmeister.com he will gladly answer any questions.



To-Do

The Unix `exiftool` tool finds the metadata of a JPG file in a flash, leaving social media users wondering what juicy bites of data they are giving to Facebook and company when they post them. In addition to the date and time, the altitude, and the direction of the camera, there are also GPS coordinates that record the exact location on the earth's surface where the picture was taken (Figure 1). Online guru Kevin Mitnick even reports that the authorities once tracked down a Bolivian drug lord, because he had published a vacation photo that still contained the metadata of his secret whereabouts [1].

Verbose Photos

Quite surprisingly, in the Exif section of the image file, metadata such as the latitude and longitude of a shot's location are by no means stored in a computer-friendly format. Instead, the mobile phone might put the string `122 deg 25' 46.82'' W` there under the GPS Longitude tag and the letter `W` for western longitude under the GPS Longitude Ref tag.

To convert this string into something that can be used later to calculate distance from a reference point, you need to use a library function that converts 122 degrees, 25 angular minutes, and 46.82 angular seconds into floating-point format and ne-

gates the numerical value for the `W` in western longitude, because only eastern longitudes are positive. The result is a value of `-122.429672` for the longitude; based on this, along with the latitude, which is determined in a similar way, another library can then determine the geographical offset to the longitude and latitude of another image.

Now the distance between two points on the earth's surface, given as latitude and longitude, cannot be calculated quite as trivially as within a two-dimensional coordinate system. But if you quickly put on your old trigonometry hat from your school days, you can still work it out. The technical term for the curved distance is "orthodrome" ([2], Figure 2); it's a segment of the great circle on the (approximated) spherical surface of the earth, which is why the whole thing is also known as the "great circle distance." Thanks to the Internet, you don't even have to type this by hand, but can turn to a Go library like `go1ang-geo` [3].

With these tools, whipping up an algorithm in the script shown today (Listing 1, [4]) is easy as pie. It accepts a reference image and a search path for the photo collection. It extracts the reference



```

$ exiftool hydrant.jpg | grep GPS
GPS Latitude Ref      : North
GPS Longitude Ref    : West
GPS Altitude Ref     : Above Sea Level
GPS Time Stamp       : 04:18:04.06
GPS Speed Ref        : km/h
GPS Speed            : 0
GPS Img Direction Ref : True North
GPS Img Direction    : 75.69078947
GPS Dest Bearing Ref : True North
GPS Dest Bearing     : 255.6907895
GPS Date Stamp       : 2015:03:31
GPS Altitude         : 53.1 m Above Sea Level
GPS Date/Time        : 2015:03:31 04:18:04.06Z
GPS Latitude         : 37 deg 45' 5.20" N
GPS Longitude        : 122 deg 25' 46.82" W
GPS Position         : 37 deg 45' 5.20" N, 122 deg 25' 46.82" W
$

```

Figure 1: The mobile phone photo contains the GPS data of the location where it was shot.

GPS coordinates from the former and then rummages through all the images in the photo collection one after another, comparing their GPS coordinates with the reference and reporting a hit if the distance to the reference image is below a preset minimum.

Deep Sea Diver

The `filepath.Walk` function from Go's core treasure trove plumbs into the depths of the files in the photo collection referenced as a directory no matter how deeply nested. The call in line 45 accepts a callback function (`Visit()` defined in line 51). To ensure that this function has the GPS data of the reference image ready for comparisons, a structure of the type `Walker` (defined as of line 14) gets created as its receiver, which is how Go ties functions to structs as objects.

First, lines 40 to 43 initialize a `Walker` instance in `w` by setting the `refLat` and `refLong` attributes of the GPS data to the values of the reference image, and then `Walk()` in line 45 receives `w.Visit` as a callback function. Since `Visit()` in line 51 is defined with a receiver of the `*Walker` type, as a result, the file system walker calls the `Visit()` callback function for each file found, but passes the `Walker` structure filled with the reference data to it each time, which means that `Visit()` can conveniently pick up the location of the reference image in line 63 and use it to compute the offset distance.

The callback also uses a regular expression in line 53 to check whether the file just found also has a `.jpg` suffix in its name; thanks to the `"(?i)"` expres-

sion in the regex string, this also matches uppercase letters as in `.JPG`. The somewhat strange `MustCompile()` call for compiling the regular expression in line 53 is attributable to Go's strict error management. Every time a function gets called that could possibly go wrong, the program has to check if all systems are go, or an error has occurred, in which case it has to initiate rescue actions. Now, in theory, compiling a regular expression could go wrong (for example, if it contains illegal syntax), but with a static (and hopefully tested) string, this is impossible.

In this way, functions with a "must" in the name, like `MustCompile()` for regular expressions, save the programmer from handling errors by internally aborting the program with a `Panic()` if

something goes wrong. By definition, the function itself does not return an error, because it only returns if everything actually works out.

Two Libraries for One, Hallelujah!

On GitHub, several libraries for reading Exif metadata from JPG photos vie for the favor of Go programmers. The most handy package I found was `goexif2`, which doesn't freak programmers out with thousands of options, but uses `Decode()` to analyze an image and returns the longitude and latitude as floating points with `LatLong()`. The project by GitHub user `xor-gate` is no longer under development according to the readme, but it still worked perfectly for the given task.

If you are looking for enhancements, you will find about a dozen forks, but deciding on a specific one is like tossing a coin. I found the original is good enough, and it can be installed in the usual style for Go:

```
go get github.com/xor-gate/goexif2/exif
```

Following which, programs can utilize its functions (e.g., by lines 5 and 83 in Listing 1).

The `GeoPos()` function from line 76 expects an image path as a string and returns two floats and an error value to the caller. First it opens the image file for reading, interprets the JPEG data with `Decode()`, then uses `LatLong()` to read the

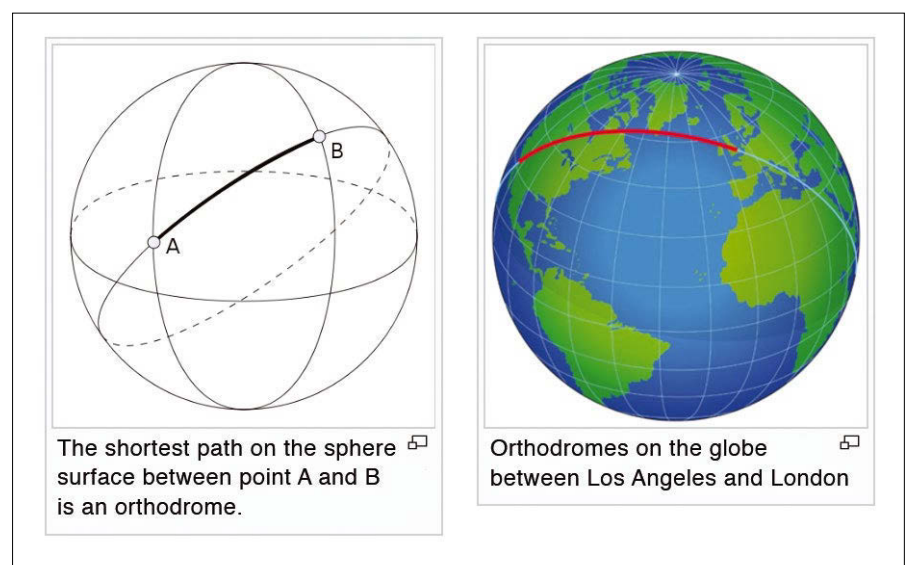


Figure 2: The shortest connection between two points on a spherical surface. Wikipedia, in the public domain, CC BY-SA 3.0

Listing 1: geosearch.go

```

01 package main
02
03 import (
04     geo "github.com/kellydunn/golang-geo"
05     exif "github.com/xor-gate/goexif2/exif"
06     "log"
07     "os"
08     "path/filepath"
09     rex "regexp"
10 )
11
12 const maxDist = 0.1
13
14 type Walker struct {
15     refLat float64
16     refLong float64
17 }
18
19 func main() {
20     if len(os.Args) != 3 {
21         panic("usage: " + os.Args[0] +
22             " refimg searchpath")
23     }
24     refImg := os.Args[1]
25     searchPath := os.Args[2]
26
27     log.Printf("Pics close to %s\n", refImg)
28     geoSearch(refImg, searchPath)
29 }
30
31 func geoSearch(refImg string,
32     searchPath string) {
33     log.Printf("Walking %s\n", searchPath)
34
35     refLat, refLong, err := GeoPos(refImg)
36     if err != nil {
37         panic(err)
38     }
39
40     w := &Walker{
41         refLat: refLat,
42         refLong: refLong,
43     }
44
45     err = filepath.Walk(searchPath, w.Visit)
46     if err != nil {
47         panic(err)
48     }
49 }
50
51 func (w *Walker) Visit(path string,
52     f os.FileInfo, err error) error {
53     jpgMatch := rex.MustCompile("(?i)JPG$")
54     match := jpgMatch.MatchString(path)
55     if !match {
56         return nil
57     }
58
59     lat, long, err := GeoPos(path)
60     if err != nil {
61         return nil
62     }
63     p1 := geo.NewPoint(w.refLat, w.refLong)
64     p2 := geo.NewPoint(lat, long)
65
66     dist := p1.GreatCircleDistance(p2)
67     if dist > maxDist {
68         return nil
69     }
70
71     log.Printf("File: %s\n", path)
72     log.Printf("Distance: %f\n", dist)
73     return nil
74 }
75
76 func GeoPos(path string) (float64,
77     float64, error) {
78     f, err := os.Open(path)
79     if err != nil {
80         return 0, 0, err
81     }
82
83     x, err := exif.Decode(f)
84     if err != nil {
85         return 0, 0, err
86     }
87
88     lat, long, err := x.LatLong()
89     if err != nil {
90         return 0, 0, err
91     }
92
93     return lat, long, nil
94 }

```

GPS Latitude and Longitude Exif tags, analyzes their angular minutes and seconds, and converts them into floating-point values.

`golang-geo` was selected as the library featuring geometry functions for determining the distance between two locations. Line 4 adds it to the program under the `geo` nick; the installation again

needs to be completed up front – as shown above – with `go get`.

Distance on a Great Circle

The programming logic as of line 63 determines the distance between the currently examined image and the reference shot. Using longitude and latitude pairs, it calls `NewPoint()` to fill a Go structure

for `golang-geo` and then calls the `GreatCircleDistance()` function with the start point `p1` and the end point `p2`. The distance between the two points in kilometers is returned as a floating-point number. If it is above the maximum distance of 100 meters (0.1 kilometers, or about 100 yards) specified in line 12, the return statement in line 68 ignores the

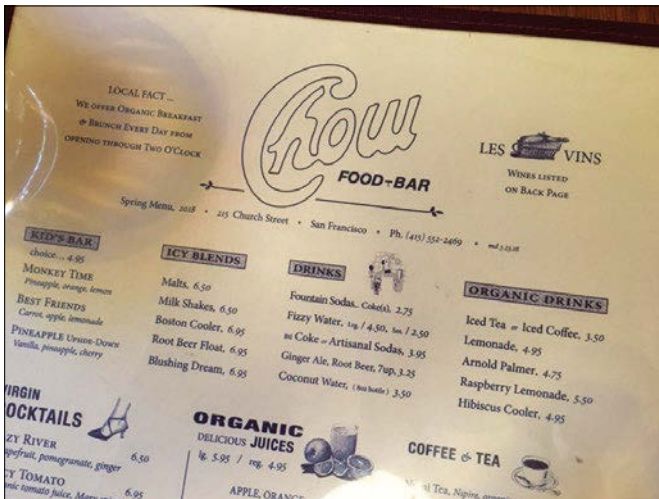


Figure 3: The reference photo for the image search shows the menu of what was once Mike Schilli's favorite restaurant.



Figure 4: Found: Another snapshot, taken after receiving the check in the same restaurant on another day.

image; otherwise, lines 71 and 72 output the name of the photo file and its offset to the reference image.

While searching for photos of the now closed restaurant mentioned at the beginning, the algorithm actually discovered something interesting. I had passed in a reference photo of the restaurant's menu, shot recently (Figure 3), as a salt. The search came back with a photo of a wooden table, showing empty beer and wine glasses, no doubt at the same restaurant, after the waiter had brought the bill in a clean water glass, as was the custom at this place (Figure 4). Eureka, what a historic moment!

Big AI Brother

Imagine the possibilities if the GPS data of a photo collection are available. An AI program using the nearest neighbor method could, for example, form clusters of the snapshot locations – as already ex-

plained in a previous issue [5] – and thus determine the most often frequented locations of the mobile phone owner. If you think about it, it's downright scary! But social media companies, as you know, live off it to keep the lights on. ■■■

Info

- [1] Mitnick, Kevin. *The Art of Invisibility*. Little, Brown and Company, 2017: <https://www.amazon.com/Art-Invisibility-Worlds-Teaches-Brother/dp/0316380504>
- [2] Orthodrome: https://en.wikipedia.org/wiki/Great-circle_distance
- [3] golang-geo, a library for calculating distances between geographical points: <https://www.github.com/kellydunn/golang-geo>
- [4] Listings for this article: <ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/225/>
- [5] "Calculating Clusters with AI Methods" by Mike Schilli, *Linux Magazine*, issue 145, December 2012, p. 62

THE COMPLETE
LINUXVOICE

ARCHIVE DVD: ALL 32 ISSUES!

LINUXVOICE

ARCHIVE DVD: ALL 32 ISSUES!

ISSUE 200 JULY 2017

LINUX
MAGAZINE

DVD

3,400
PAGES OF

LINUX AND OPEN SOURCE LOVE!

3,400
PAGES OF

LINUX AND OPEN SOURCE LOVE!

Since April 2014, Linux Voice has showcased the very best that Free Software has to offer. Now you can get it all on one searchable DVD.

**Includes all 32 issues in
EPUB, PDF, and HTML format!**

Order now!

shop.linuxnewmedia.com



SystemRescueCD – a live system that rescues data and systems

Emergency Medicine

The SystemRescueCd live system contains numerous tools that you can use to recover deleted files or a defective system. *By Tim Schürmann*

The SystemRescueCd live system above all offers programs with which you can reanimate defective data carriers and recover data. It includes the Firefox browser, which can also be used to search for solutions to a problem on the Internet if the permanently installed system fails to boot. Finally, SystemRescueCd provides useful tools for everyday work, such as creating or shrinking hard disk partitions. The live system relies on standard tools such as the well-known GParted for partitioning hard disks.

Bloated

As the CD in the name indicates, the SystemRescueCd fit on a CD for a long time. In version 6.0.0, however, the developers replaced the existing substructure with Arch Linux. As a result, the SystemRescueCd 6.0.2 (the latest release when this article was written), occupies almost 871MB of disk space. With a little luck, you can just about burn it onto an extra length CD (100-minute CD). But in any case, SystemRescueCd can be booted from a DVD or USB stick.

On the downside, the live system now only runs on 64-bit systems with Intel

or AMD processors. If you want to save an ancient system with a 32-bit processor, you first need to remove the hard disk and, for example, connect it to another system via an external hard disk enclosure. Alternatively, you can turn to the older SystemRescueCd v5.3.2, which you can still find in the project archive [1]. Furthermore, SystemRescueCd will not start on systems where the secure boot mechanism is enabled: You first need to disable this in the BIOS or UEFI settings.

DIY Burning

To start using SystemRescueCd, go to the project website [2]. When you get there, click on *Site map* and then *Download* in the page's left margin; in the table that appears, then click on the file name next to *Download link*. Burn the resulting ISO image onto an extra-long CD or DVD using an appropriate program. The developers recommend K3b, Brasero, Xfburn, or cdrecord.

If you want to boot the live system from a USB stick, you only have to write the file with the extension `.iso`

to an empty USB stick. The SystemRescueCd developers recommend the `dd` command-line tool for Linux.

In Listing 1, replace `systemrescuecd-6.0.2.iso` with the file name of the downloaded ISO image and `/dev/sdc` with the device name of the USB stick. Caution: `dd` overwrites all the data on the target medium. Make sure that `of=` is followed by the USB stick's device name. If in doubt, you can use `lsblk` to list all data carriers. Further assistance for this process can be found on the project's website [3].

Alternatively, you can order a USB stick with preinstalled SystemRescueCd from various retailers, such as OSDisc [4].

Booted

As soon as you boot a system from the prepared start medium, the menu in Figure 1 appears. Pressing the Enter key enables the first menu item and boots the live system directly. If you select *Boot SystemRescueCd and copy system to RAM*, SystemRescueCd first copies the

Listing 1: Writing SystemRescueCD to a USB Stick

```
dd if=systemrescuecd-6.0.2.iso of=/dev/sdc; sync
```

Lead Image © Author; 123RF.com

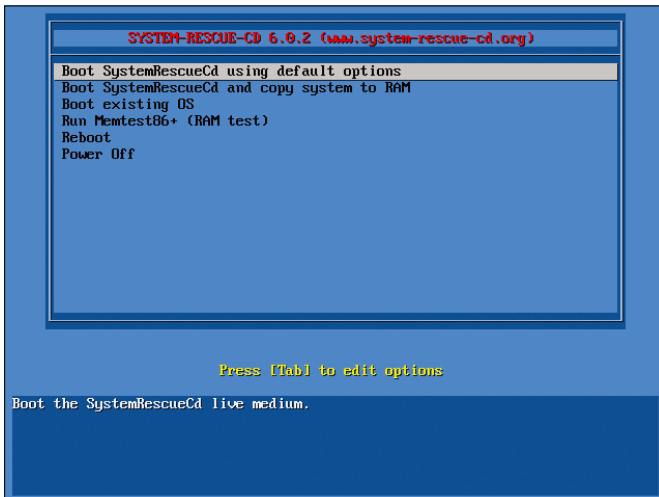


Figure 1: On UEFI systems, only the first option is available; you need a BIOS system to access the other options.

complete boot media to the main memory and then starts the live system from there. After booting, you can remove the boot media, which can be advantageous for notebooks with only a few USB ports or only one DVD drive.

The other options in Figure 1 are only available if you launch the SystemRescueCd on a system with BIOS. *Boot existing OS* lets you start a system already installed on the computer. This is handy if you accidentally forgot the DVD with SystemRescueCd in the drive or the bootloader does not work. *Run Memtest86+ (RAM test)* starts the program of the same name, which checks the main memory for defects.

You should always run the tool if you suddenly notice strange system behavior while working on your system or if Linux starts to swap data to the hard disk extremely frequently. Older versions of SystemRescueCd had some additional tools in the menu, which had to be removed with version 6.0.0. *Reboot* and *Power Off* let you restart or shut down.

The Force Is with You

If you choose to boot SystemRescueCd, you will end up at the plain vanilla prompt shown in Figure 2 after the boot process. The live system has already logged you in as the omnipotent *root* user, so you are allowed to fiddle with all the knobs and dials – all the more reason to think carefully about the commands you will be running.

By default, SystemRescueCd uses the US keyboard layout, so you may want

to change the layout by calling `setkmap` and selecting the desired layout from the list. Note that this setting only applies to the command line.

You can launch the graphical user interface with the `startx` command; this simultaneously displays the lean Xfce desktop environment on the screen. All the major graphical programs are provided in the start menu (located behind the mouse icon at the bottom left of Figure 3).

The applications under *Settings* set up the desktop environment and the live system. For example, the screen settings are hidden behind *Display*, and the key-

board layout can be changed with *Keyboard* in the *Layout* tab.

Under *Accessories*, you will find several text editors, which you can use to edit configuration files. Besides the classic Vim, there are also more user-friendly offerings such as Joe's Own Editor and FeatherPad. Alternatively, you will find Geany under *Development*, while Nano is available at the command line.

Office applications like LibreOffice or Gimp are deliberately not provided. Since the current SystemRescueCd is based on Arch Linux, you can install additional software using the `pacman` [5] package manager. The syntax for the search is

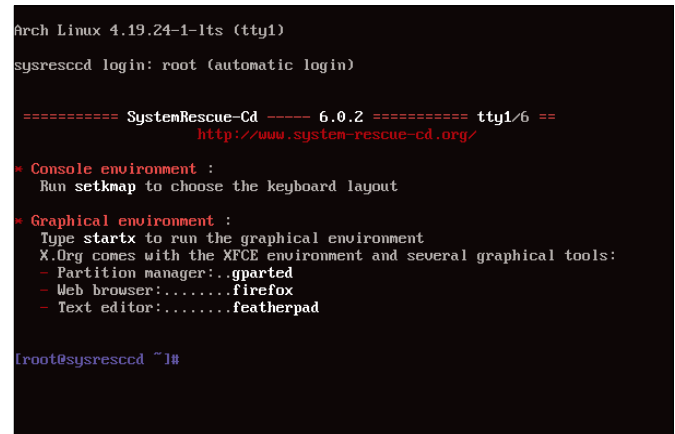


Figure 2: SystemRescueCd boots to the command line by default. This is especially useful if Linux fails to identify the graphics card.



Figure 3: You will find quick start icons for the four particularly important applications Xfce Terminal, Firefox, GParted, and FeatherPad in the panel's bottom left corner.

Listing 2: Setting Up the Network

```
01 # systemctl stop NetworkManager
02 # ip address add 192.168.1.101/24 dev enp0s3
```

```
pacman -Ss <search key>
```

to install, type

```
pacman -S <package>
```

Going Online

SystemRescueCd automatically tries to connect to the Internet. In the background, the live system relies on NetworkManager. A left click on the network icon bottom right in the panel shows the active connections that you

can disconnect manually with *Disconnect*. If required, you can set up a connection yourself by right-clicking on the network icon and selecting *Edit Connections* or by selecting *Settings | Advanced Network Configuration* from the start menu.

At the command line, you can also set up the network with the familiar tools, such as *ip* or *wcncfig* for WLAN connections. To do this, however, you first need to disable NetworkManager (Listing 2, line 1). Then enable the desired interface and assign an IP address using *ip*; in line 2 of Listing 2, the Ethernet interface goes by the name of *enp0s3*.

can disconnect manually with *Disconnect*. If required, you can set up a connection yourself by right-clicking on the network icon and selecting *Edit Connections* or by selecting *Settings | Advanced Network Configuration* from the start menu.

The TigerVNC Viewer, which you will find under *Internet* below the start menu, lets you quickly connect to a VNC server.

System | LSHW lists the hardware installed on the computer, thus giving you an overview of the system (Figure 4). First click *Refresh* and then double-click *Computer*. Then double-click on the terms in the other columns to access the desired components. In this way, you can find out, among other things, which firmware is used on a device and whether Linux has detected the device in question at all.

At the command line, *lshw* lists all the hardware components. You can keep an eye on the rescue system's load and active processes with *Htop* and *Task Manager*. You will find both in the start menu below *System*. *Htop* can also be called directly in a terminal window (Figure 5).

Disk-Jockeying

File systems can be easily mounted by double-clicking on the *Home* icon on the desktop to start the Thunar file manager and then clicking on the data carrier in the sidebar. If you mount a disk at the command line, avoid mounting it directly in the */mnt/* directory: This could crash the live system.

Instead, create a directory like *backup/* below */mnt/* and then mount the disk in the */mnt/backup/* folder. ZIPs and other archives can be opened by double-clicking or using *Xarchiver*, which you can access below *Accessories | Xarchiver* in the start menu. SystemRescueCd can

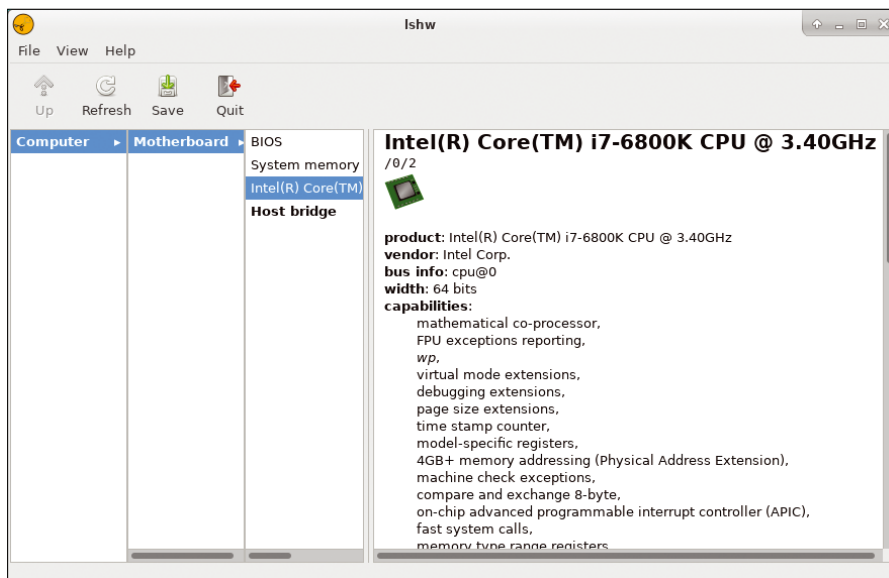


Figure 4: Here *lshw* has detected an Intel processor among other things. All the black terms on the left side can be double-clicked to access further information.

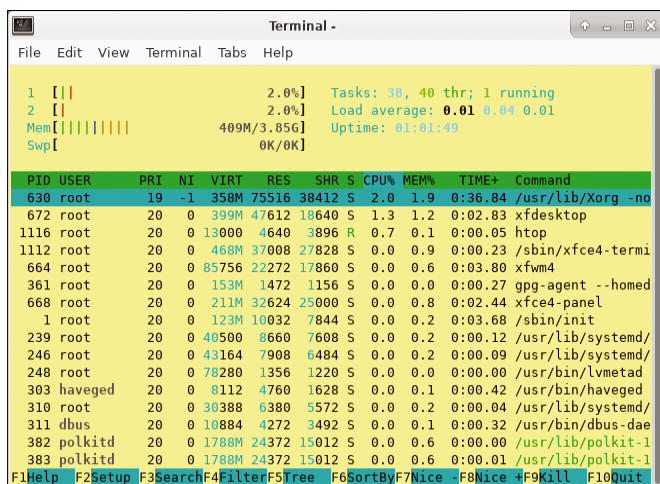


Figure 5: *Htop* helps you keep track of the active processes, as well as the system load and memory consumption of the live system.

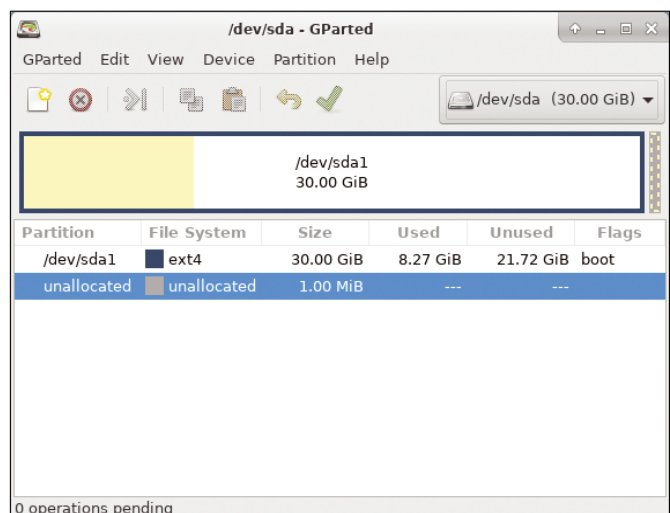


Figure 6: GParted wants to change the first hard disk (*/dev/sda1*), which occupies 30GB on the hard disk.

handle ZIP as well as many other formats, such as RAR. At the command line, you need to use command-line programs such as `unzip` or `unrar` to unpack the files.

One especially useful tool is hidden below *System | GParted*. The graphical partition editor lets you change the current partitioning or create new partitions (Figure 6). Before you call a function, first make sure that you have selected the correct data carrier in the drop-down list top right. In the list below, you will find all the current partitions; the bar chart between toolbar and list visualizes the capacity usage. Using the Partition menu, you can apply various actions to the partition currently selected in the list (e.g., changing its size with *Resize/Move*).

Check tells GParted to check the filesystem on the partition for faults and try to correct them. However, the tool cannot do magic and is powerless in case of hardware defects. GParted first collects all the actions you have selected. To actually apply them, click on the green check mark in the toolbar or select *Edit | Apply All Operations*.

Reanimation

PhotoRec and TestDisk help rescue deleted files. Both are called in a terminal window. If the tools have unearthed a lot of files, the program hiding behind *System | Bulk Rename* can help. It assigns multiple files new file names in one fell swoop.

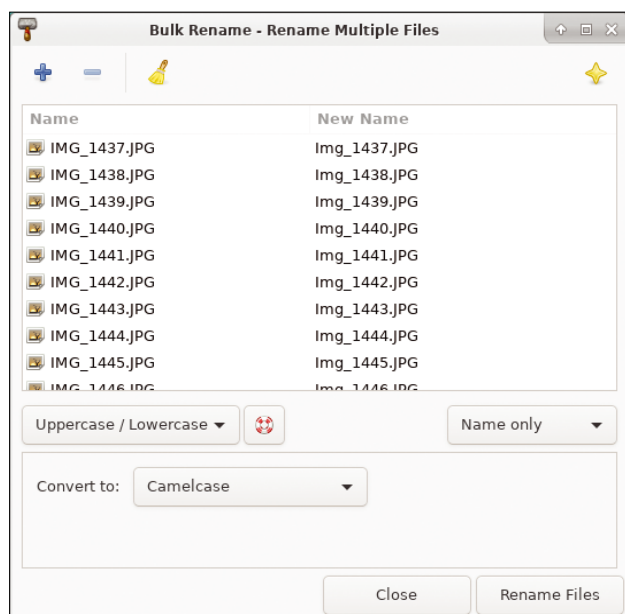


Figure 7: With these settings, Bulk Rename changes the capitalization of the file names.

First click the plus icon, and then select all the files to be given new

names. The files now appear in the list on the right, showing the future new file names. You can then use the drop-down lists at the bottom of the window to configure the desired changes (Figure 7).

The Partimage and the Partclone tools are also launched in a terminal window. The programs create complete images of partitions and restore these backups on demand. This is useful even if the hard drive is on its last legs. If you clone the disk, you can immediately continue working with the copy, thus avoiding aggravating the damage to the original.

Partimage and Partclone only save occupied sectors of known filesystems and abort on read errors. Ddrescue copies a data carrier bit by bit. It thus also captures seemingly unoccupied areas of the data carrier in which valuable data could still be stored, as well as supporting less common filesystems. However, the backup will take up the same amount of disk space as the existing partition.

The command from Listing 3, line 1, for example, writes the contents of the `/dev/sda1` partition to a file named `backup.img`. You can save the partition

Listing 3: Creating a Backup with ddrescue

```
01 # ddrescue /dev/sda1 backup.img
02 # sfdisk -d /dev/sda > sdatabelle.bin
03 # cat sdatabelle.bin | sfdisk /dev/sda
```

At Your Command

SSH and SCP are also included in the SystemRescueCd feature set; the `lftp` FTP client is also available in the start menu under *Internet*. To start the SSH daemon on the live system, use:

```
systemctl start sshd
```

If you want to connect as root from another system with this daemon, you have to assign a password to the account using `passwd root`. `Rsync` can be used for file backups if required. `Rsnapshot`, which was included with older SystemRescueCd versions, is no longer part of the current version.

If you want to make sure that all confidential data on a disk has been destroyed, then run the `shred` tool. The call to

```
shred -v /dev/sda
```

deletes the whole hard disk `/dev/sda` irretrievably. The data on the disk can then no longer be restored – not even by PhotoRec and its cohorts.

Conclusions

SystemRescueCd is a lean toolbox for emergency recovery. Many of the useful programs are command-line tools. It makes good sense to first try out, and familiarize yourself with the use of, the tools on a test system. This is the only way to ensure that you stay calm and choose the right parameters if worse comes to worst. Various instructions on the SystemRescueCd website [6] will help you with this. ■■■

tool. The command from line 2 moves the table from the hard disk `/dev/sda` to a file named `sdatabelle.bin`. You can then use the command from line 3 to restore the table if necessary.

Info

- [1] SystemRescueCD 5.3.2: <https://osdn.net/projects/systemrescuecd/storage/releases/5.3.2>
- [2] SystemRescueCD: <http://www.system-rescue-cd.org>
- [3] Setting up SystemRescueCD on a USB stick: <http://www.system-rescue-cd.org/Installing-SystemRescueCd-on-a-USB-stick>
- [4] OSDisc: <https://www.osdisc.com/products/linux/systemrescuecd>
- [5] Arch Linux Wiki on pacman: <https://wiki.archlinux.org/index.php/pacman>
- [6] SystemRescueCD online manual: <http://www.system-rescue-cd.org/manual>



MakerSpace

DDS-based Rasp Pi function generator

Pass the Hat

A touch display, a case, and a custom add-on board transform the humble Rasp Pi into a high-performance function generator that rivals expensive commercial offerings. *By Andrew Malcolm*

Modern laboratory instruments are a marvel of integration, combining hardware, software, and often mechanical components to produce versatile and highly functional units that bring real value to the engineers and technologists who use them. In this article, I aim to show how such an instrument might be put together by combining the Raspberry Pi with several elements to provide a low-cost and flexible function generator that can rival its more expensive commercial cousins on price and possibly beat them in terms of flexibility.

Function Generator

A signal or function generator is a versatile frequency source able to output a signal in a number of output wave-shapes and at an adjustable amplitude and frequency from direct current (DC) up into the high-frequency electromagnetic radio frequency (RF) region, depending on the application. Sophisticated instruments can also include frequency, phase, or amplitude modulation useful for testing radio transmitters and receivers.

No engineer's workbench is complete without such an instrument. Together with an oscilloscope, it allows an engi-

neer to characterize the frequency and phase response of an amplifier, to test and calibrate counters and frequency meters, and to replace built-in oscillators in devices under test or during development.

A variable frequency and amplitude signal is often required for uses beyond the purely electronic, such as for applications in mechanical and acoustic engineering for testing loudspeakers and driving vibration platforms and a plethora of other electronically excited mechanical devices. In the field of medicine, variable-frequency sources are used to test the response of the human ear, as well as to drive ultrasound devices used in imaging. The list goes on and on.

Many of the commercially available instruments cost upward of several hundred dollars. Because of their self-contained nature, modifying their function is not generally possible, and embedding them into a larger system can be difficult because of their size and power requirements. Modern analog integrated circuit technology makes available to the instrument designer a vast array of building blocks, many of them programmable, to help build a sophisticated instrument with a minimum of components. The addition of a microcontroller or other embedded computer enables the designer

to add extra features and produce a fully featured, flexible instrument that is easy to use.

The instrument presented here is based on a Raspberry Pi, an excellent basis for personalized instrumentation, the official Pi 7-inch LCD screen and case, and a direct digital synthesis (DDS) function generator on a Hardware Attached on Top (HAT) board of my design (see the “Specifications” box).

Hardware

Practically all modern signal or function generators are implemented by DDS (see the Direct Digital Synthesis box). Because DDS chips are readily available for a few dollars, you can easily build an instrument with similar performance and greater flexibility for a few tens of dollars. The maximum output frequency is limited by the semiconductor technology used to build the chip and the master or “clock” frequency (usually derived from a crystal oscillator) used to drive the chip. Because the generation is purely digital, the frequencies produced have no lower bound. For this project, I have chosen the AD9833BRMZ [1] DDS chip from Analog Devices, which has a range up to 12.5MHz. As a side benefit, this chip can also produce triangular and square wave outputs.

A useful generator must also have a means to adjust output amplitude. Amplitude control can be achieved many ways, from simple, mechanical variable potentiometers (pots) to their semiconductor equivalents. Many of

these approaches have limitations because of excessive stray capacitance, linearity, stability, and the ability to control the amplitude with software. An optimum solution is to use a multiplying DAC, wherein the reference signal is replaced by the signal to be controlled. These so-called multiplying DACs are available and can control amplitudes of frequencies into the tens of megahertz and beyond. The AD5443Y-RMZ [3] from Analog Devices is such a chip, providing 10-bit resolution of amplitude control.

Both of the chips selected have SPI interfaces, so they are ideal for interfacing to the Raspberry Pi. Additionally, both are supported by Analog Devices’ Linux Industrial I/O Subsystem drivers (see the “IIO” box).

Because the generated signal is a stepwise approximation to the ideal sine wave output, the output must be filtered. The filter chosen for this design is a ninth-order elliptic low-pass filter. Filters of this sort are a complex topic beyond the scope of this article, but you can find a detailed explanation online [5]. A passive filter comprises only resistors, capacitors, and inductors.

A number of online resources (e.g., RF Tools [6]) offer simple design tools that allow you to dial in the required parameter and then deliver a schematic of the filter, the component values, and a graphical phase and frequency response. The filter for the current design has a corner cut-off frequency of around 12.5MHz, or half of the chosen crystal drive frequency of 25MHz. This filter is an anti-alias filter, and by attenuating the higher frequencies (greater than the Nyquist frequency), it prevents the aliased components from being output.

A wide-band power amplifier is employed to produce output that can be used to drive other instruments at useful power levels. The OPA564AIDWP from Texas Instruments [7] provides an output of +/-10V peak-to-peak at more than 1A across the 10MHz frequency range, so it is ideal for this application. It has built-in overcurrent protection and a thermal shutdown mode, so it is immune to the type of abuse that a bench instrument might encounter.

The remaining components that make up a practical design are connectors, op-

amps for level shifting and buffering, potentiometers for calibration, power supplies, and filters. The power supply used to power the instrument must be of sufficient capacity to supply both the Rasp Pi and the DDS board. The official universal supply [8] is recommended. The block diagram in Figure 1 shows how the essential elements are connected to form the complete instrument.

The excellent open source schematic capture and printed circuit board (PCB)

Direct Digital Synthesis

Direct digital synthesis is a well-known technique for generating repetitive waveforms from a single, high-frequency source such as a crystal oscillator. The concept is fairly simple. Samples of the required waveform are stored in RAM or ROM (in the case of a sine wave, only a quarter of a full cycle need be stored, and the rest inferred by symmetry). A counter is then used to access the stored samples sequentially, and the samples are passed to a digital-to-analog converter (DAC), which then outputs the desired waveform.

By varying the frequency fed to the counter, the frequency of the output may be varied. The resulting waveform is a stepped approximation of the desired waveform; an output filter is required to reconstruct the exact waveform. This filter is generally a high-order passive filter with a corner frequency set as half the reference clock frequency. You can find more information about DDS in an online tutorial [2].

IIO

The Linux Industrial I/O Subsystem (IIO) is an attempt at a unified device driver interface for chips such as data converters, sensors, and frequency sources. The wiki [4] gives an excellent in-depth description. From the perspective of this article, the main thing to note is that, once loaded, a device driver for a particular device exposes readable and writable parameters as entries in the /sys tree of the filesystem that allow user programs to sample and modify the device’s function by file semantics, with no low-level (I2C/SPI/serial/etc.) programming required. High-speed devices also support a streaming interface.

Specifications

- 0.01Hz to 10MHz, adjustable in 0.01Hz steps
- Sine, square, and triangle outputs
- 50-ohm or low-impedance output with jumper option
- Output adjustable from +/-1mV to +/-5V, peak-to-peak
- Transistor-transistor logic (TTL) sync output
- Ninth-order filter for sine purity
- Linux drivers or access through the serial peripheral interface (SPI) bus
- Current limit warning LED
- Onboard supplies for analog section
- SMA connector for output

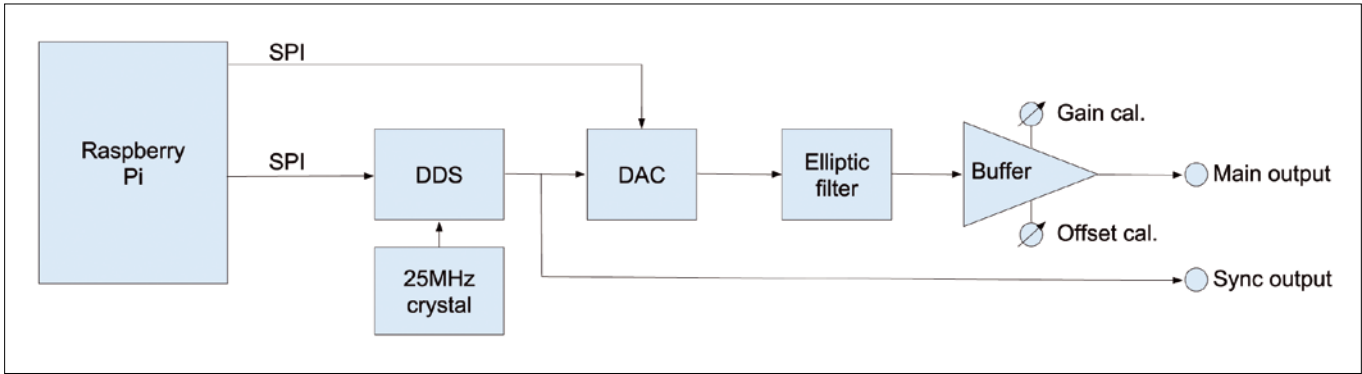


Figure 1: DDS function generator block diagram.

design tool KiCad [9] was used to capture the design and layout of the two-layer PCB. The schematic (Figure 2) is a complete implementation of a practical DDS function generator, with power supplies, filtering, synchronization output, level shifters between stages, and calibration potentiometers to set null the output offset in the power amplifier and to set the full scale gain to the value indicated in the GUI.

Along with the schematic, KiCAD presents a 3D render of the fully laid out

PCB (Figure 3). The KiCAD 3D models are invaluable, allowing designs to be incorporated into larger mechanical models to ensure the mechanical positions of connectors and other elements are correct and that no component fouls any nearby mechanical element, such as features of an enclosure.

Mounted on a Raspberry Pi, the DDS board fits within the official 7-inch Rasp Pi display case [10]. A small modification to the case is required to allow the output connector to protrude through

the case wall (Figure 4). Two vertical saw cuts toward the HDMI connector remove a small rectangle of plastic and do not in any way affect the structural integrity of the case.

The Software

Today's instruments have generally moved away from front panels loaded with knobs and switches to a more generic user interface with an LED or LCD screen and possibly a single multifunction control knob. The interface designed

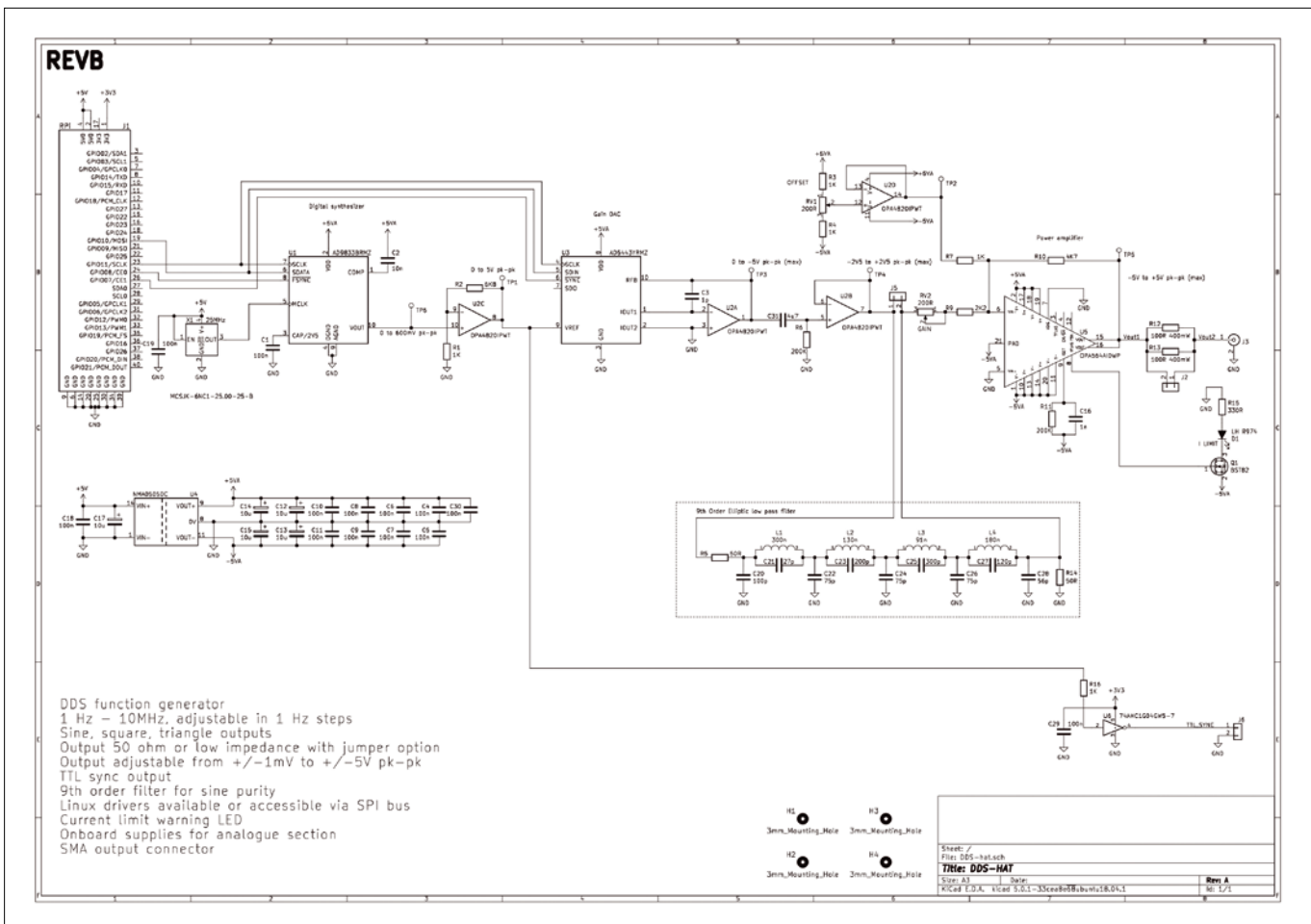


Figure 2: DDS function generator schematic.

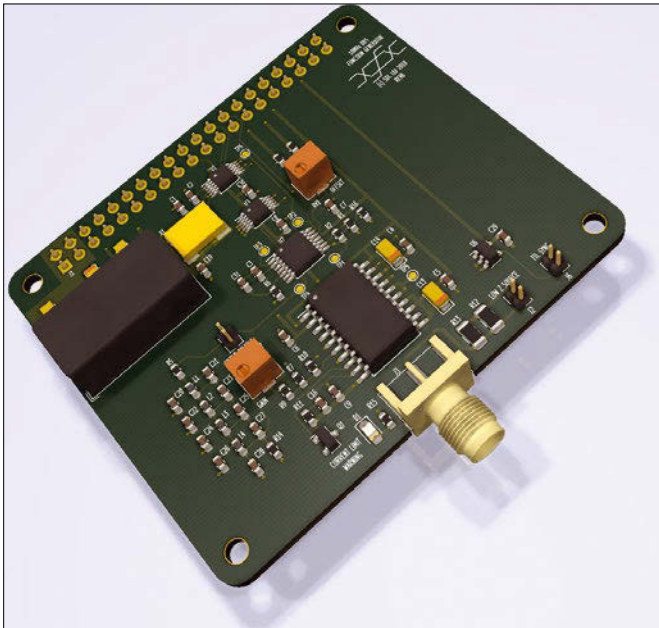


Figure 3: KiCAD 3D PCB layout.

for the current instrument is intended to run on the standard Rasp Pi 7-inch touch LCD [11], so the user interface will be entirely touch-based, allowing a compact instrument to be built into the standard case. The whole setup is packaged as an instrument, with a Python-based GUI allowing the full control of frequency, amplitude, and waveshape that you would expect from a commercial instrument.

As already stated, many instruments use a multipurpose spinner, or control dial, to change operating parameters that are displayed on an LED or LCD screen. The current GUI uses the same paradigm (Figure 5), with a circular GUI dial that controls frequency and amplitude. Two buttons associated with the dial select the decade [12] currently being edited, and another two allow you to jog the setting up or down for precise control. The frequency of voltage being edited is displayed in the central area of the screen, with the voltage displayed below. A series of buttons selects the function of the

dial, and other buttons control output waveform and frequency range.

The IIO subsystem drivers hide the details of the SPI interface that are used to communicate with the DDS and DAC chips. The parameters that can be varied, such as frequency, level, and waveshape appear as files in the /sys tree. Simply writing valid values to those files causes SPI messages to be sent to the chip. (Although it would be possible to bypass the IIO drivers, i.e., not load them at all, and generate SPI messages directly, it would require a detailed understanding of the SPI protocol used by each chip and, in my opinion, is a lot of hard work for no appreciable gain.)

The IIO driver takes care of scaling and all the low-level communication de-



Figure 4: A modified Rasp Pi case allows the board's connector to protrude.

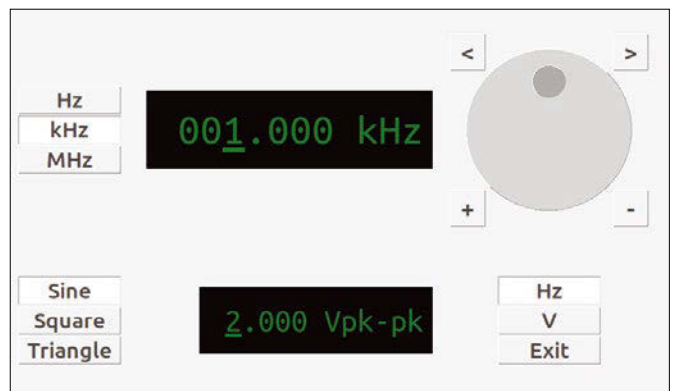


Figure 5: The GUI for the DDS-based Raspberry Pi function generator. Table 1 describes the function of each region of the GUI.

tails, which makes interfacing with the chip from a high-level language such as Python extremely convenient. The core methods required to communicate with the DDS board are shown in Listing 1. The `iio.py` routines control the hardware through the IIO device drivers and their entries in the /sys tree and represent the least amount of code required to control the generator.

GUI

If you want to design your own GUI or embed the generator in another system,

Table 1: GUI Functions

Position	Function
Top left	Range selector
Top center	Frequency output indicator with cursor and units.
Top right	Left and right arrows select decade (the cursor under the '1' character in the frequency output indicator). Plus and minus jog up or down 1 unit. Dial scrolls frequency up or down. If you scroll past 9, the cursor moves to the left and continues to increment.
Bottom left	Waveform selector.
Bottom center	Voltage output indicator with cursor and units.
Bottom right	Selector for dial function.

Listing 1: iio.py

```

01 import sys
02
03 #
04 # This file provides routines to control the hardware via the iio
05 # device drivers and their entries in the /sys tree.
06 # This is the least amount of code required to control the generator;
07 # all the other files simply provide a GUI front end.
08 # If you want to design your own GUI or embed the generator in another system,
09 # this is all you need.
10 #
11
12 ## dds registers
13 frequency_register='/sys/bus/iio/devices/iio:device1/out_altvoltage0_frequency0'
14 wavetype_register='/sys/bus/iio/devices/iio:device1/out_altvoltage0_out0_wavetype'
15 enable_register='/sys/bus/iio/devices/iio:device1/out_altvoltage0_out_enable'
16
17 # dac registers
18 voltage_register='/sys/bus/iio/devices/iio:device0/out_voltage0_raw'
19
20 # disable calls to hardware in test mode
21 test=(len(sys.argv)>1)
22
23 # enable output
24 def enableOutput(enable):
25     if test==False:
26         fo=open(enable_register,'w')
27         if enable==True:
28             fo.write('1')
29         else:
30             fo.write('0')
31
32 # write frequency in Hz to DDS
33 def setFrequency(frequency):
34     output=str(int(frequency))
35     #print('F='+output)
36     if test==False:
37         fo=open(frequency_register,'w')
38         fo.write(output)
39
40 # write to DDS one of 'sine', 'square', 'triangle'
41 def setWavetype(wavetype):
42     #print('T='+wavetype)
43     if test==False:
44         fo=open(wavetype_register,"w")
45         fo.write(wavetype)
46
47 # DAC full scale is 4096 (2^12), so scale from
48 # 5000 millivolts 1.23=4095/10000
49 def setVoltage(voltage):
50     output=str(int(voltage*4095/10000))
51     #print('V='+output)
52     if test==False:
53         fo=open(voltage_register,'w')
54         fo.write(output)

```

the code in Listing 1 is all you need. The Python files that create a convenient GUI to control the instrument can be downloaded from the *Linux Magazine* FTP site [13], along with the code for `iio.py`. The GUI routines include:

- `dds_gui.py`
- `vlab.py`
- `flab.py`
- `dial.py`

The `dds_gui.py` program for the function generator uses the Tkinter GUI toolkit [14]. The code mostly sets up the GUI elements and wires the button events to the appropriate routines. At the end, the program sets the instrument into its default state, enters the main loop, and waits for user input.

The `vlab.py` and `flab.py` routines extend the capabilities of a Tkinter Label and add editing facilities (for voltage and frequency, respectively), enabling a cursor to be moved from digit to digit and for that digit to be incremented and decremented.

The `dial.py` routine implements a custom Tkinter widget (control) that simulates the dial or spin knob often found on these types of instrument. Buttons allow you to jog values up and down by a single digit and to move up and down by a decade. These controls work closely with the frequency and voltage labels.

The Linux Setup

Although the following procedure should work on any modern Linux distro, I used Arch Linux on my Rasp Pi. A fully bootable SD card image of the instrument's OS and all the source code are available on my GitHub page [15].

To access the DAC and DDS chips, some device drivers must be loaded. In turn, those drivers must be given information on the location and capabilities of the chips. These types of devices do not support auto-discovery, so in the world of embedded ARM processors (and increasingly elsewhere), a device tree is used to locate and load suitable drivers for devices. Listing 2 shows the device trees for the DAC and DDS chips.

If you have used any Rasp Pi HATS before, you might be familiar with adding overlays in the `config.txt` file. In the present case, one overlay file, `dds_hat.dtbo`, is copied to the `/boot/overlays` directory and referenced from `/boot/config.txt`:

Listing 2: Device Trees

```
# dac_tree.txt
$ ls -l /sys/bus/iio/devices/iio\:device0/out*
/sys/bus/iio/devices/iio:device0/out_voltage0_raw
/sys/bus/iio/devices/iio:device0/out_voltage0_scale

# dds_tree.txt
$ ls -l /sys/bus/iio/devices/iio\:device1/out*
/sys/bus/iio/devices/iio:device1/out_altvoltage0_frequency0
/sys/bus/iio/devices/iio:device1/out_altvoltage0_frequency1
/sys/bus/iio/devices/iio:device1/out_altvoltage0_frequency_scale
/sys/bus/iio/devices/iio:device1/out_altvoltage0_frequencysymbol
/sys/bus/iio/devices/iio:device1/out_altvoltage0_out0_wavetype
/sys/bus/iio/devices/iio:device1/out_altvoltage0_out0_wavetype_available
/sys/bus/iio/devices/iio:device1/out_altvoltage0_out_enable
/sys/bus/iio/devices/iio:device1/out_altvoltage0_phase0
/sys/bus/iio/devices/iio:device1/out_altvoltage0_phase1
/sys/bus/iio/devices/iio:device1/out_altvoltage0_phase_scale
/sys/bus/iio/devices/iio:device1/out_altvoltage0_phasesymbol
```

```
lcd_rotate=2
gpu_mem=64
initramfs initramfs-linux.img
followkernel
dtoverlay=dds_hat
```

```
"/bin/sh -c 'chown -R alarm:alarm
/sys/bus/iio/devices/iio\:device0/*'
ACTION=="add", PROGRAM=
"/bin/sh -c 'chown -R alarm:alarm
/sys/bus/iio/devices/iio\:device1/*'"
```

The `.dtbo` overlay is a binary file produced by compiling a `.dts` file with the device tree compiler [16], available with most ARM-based distros. As stated earlier, the overlay tells the kernel where to find a specific device (e.g., its SPI address), something about its configuration (e.g., clock frequency), and which device driver to load. This information is passed to the device driver so that it can communicate with the device and set it up appropriately. The overlay provides a description to the kernel of the device and the environment in which it exists.

Once the drivers are loaded during boot-up, the appropriate entries should appear in the `/sys` file tree, and you will see that the entries correspond to those used by the Python routines in Listing 1.

To avoid having to run the `dds_hat` program as root, you can add some `udev` rules so that non-sudo users can access the IIO subsystem:

```
$ cat /etc/udev/rules.d/99-iio.rules
ACTION=="add", PROGRAM=
```

These rules are quite broad, and this approach is not recommended on a multiuser server; however, for an embedded device like this, it seems acceptable.

You should now be able to run the Python program and enjoy your function generator!

Conclusion

Although scope for improvement always exists, both for hardware and software, I hope the instrument described here and the details of how it was built, its function, and how it works internally prove to be of interest. This DDS HAT board could be put to a number of other uses with different software. For example, you could produce complex amplitude, phase, and frequency modulation schemes. Some examples can be found in the datasheets and associated application notes for the DDS chip that would allow integration into a communications system. ■■■

Info

- [1] AD9833 DDS specs: <https://www.analog.com/media/en/technical-documentation/data-sheets/ad9833.pdf>
- [2] DDS tutorial: <https://www.analog.com/media/en/training-seminars/tutorials/MT-085.pdf>
- [3] DAC chip: https://www.analog.com/media/en/technical-documentation/data-sheets/ad5426_5432_5443.pdf
- [4] Linux IIO system: <https://wiki.analog.com/software/linux/docs/iio/iio>
- [5] Elliptic filters: https://en.wikipedia.org/wiki/Elliptic_filter
- [6] Filter design tool: <https://rf-tools.com/lc-filter/>
- [7] TI power amplifier: <http://www.ti.com/lit/ds/sbos372e/sbos372e.pdf>
- [8] Raspberry Pi universal power supply: <https://www.raspberrypi.org/products/raspberry-pi-universal-power-supply/>
- [9] KiCAD: <http://kicad-pcb.org/>
- [10] Raspberry Pi touchscreen case: <https://thepihut.com/products/raspberry-pi-official-7-touchscreen-case?variant=15155531396>
- [11] Raspberry Pi 7-inch display: <https://www.raspberrypi.org/products/raspberry-pi-touch-display/>
- [12] Decade: [https://en.wikipedia.org/wiki/Decade_\(log_scale\)](https://en.wikipedia.org/wiki/Decade_(log_scale))
- [13] Code for this article: <ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/225/>
- [14] Tkinter for Python reference: <http://infohost.nmt.edu/tcc/help/pubs/tkinter/web/index.html>
- [15] Author's GitHub project page: https://github.com/andrewrussellmalcolm/dds_hat
- [16] Device tree compiler reference: https://elinux.org/Device_Tree_Reference

Author

Andrew Malcolm (MIEE, CEng) works as a software engineer for Guru Systems (<https://www.gurusystems.com/>), a fast-growing IoT hardware and SaaS company working on low carbon energy projects. In the evenings and weekends, he likes to combine software engineering with his first love, hardware engineering. With all the open source tools available, he is never short of devices to design. The Raspberry Pi has proven to be a source of inspiration, and to date, Andrew has designed five HATs for the Raspberry Pi. He is currently working on micro-stepping motor drives for a Pi-based laser cutting machine. You can contact him at andrewrussellmalcolm@gmail.com.



MakerSpace

Support for the Librem 5 Phone Communications Tools

Ahead of the Librem 5 phone release, Purism releases a suite of communications applications called Librem One, including Chat, Mail, and more. But how accessible are these apps for average users? *By Bruce Byfield*

With the release of the Librem 5 phone approaching, Purism has presented a foretaste of how it will be supported. Purism has released the first version of a suite of communications services called Librem One, consisting of Chat, Social, Tunnel, and Mail applications designed for privacy and encryption [1]. Although Purism has yet to reveal its plans for an app site to match Google Play or the Apple App Store, Librem One by itself is likely to be a key factor in the success of the Librem 5 or future mobile devices. Without such software support, the Librem 5's chances to survive in a saturated market would probably be slim – as the BQ Aquaris M10 tablet loaded with Ubuntu Touch proved several years ago. However, in its first release, Librem One manages to be only partially successful.

You may have already seen the promotional campaign for the Librem One services [2] (Figure 1). If so, ignore it. The campaign includes an ad featuring the voice of Linux podcaster Bryan Lunduke and the slogan, “We don’t look at your junk.” While the campaign must have seemed to someone a clever way to explain to unsophisticated users the importance of security and privacy, a charitable description of the ad would be that it is heavy-handed and full of repetitive adolescent innuendo. It badly misses the audience of early adopters, and feels badly out of sync with Librem One – not least because it suggests that the services are more user-friendly than they really are.

Instead, better to look at the Librem One campaign and roadmap directly. Users can sign up for a free account for Chat and Social, or pay \$7.99 a month

for a single account or a five member family pack for \$14.99 a month for complete services that include Mail. Other services will be added as registration for the services rises: starting with Librem Files, and continuing with Backup, Contacts, Pay, and Dial. Given that the addition of Files kicks in at 50,000 backers – 10 times the number as I write, a quarter of the way through the crowdfunding campaign – these additional services may not be delivered for a while. Ultimately, though, the intention is to provide secure and decentralized replacements for everything from Facebook to Gmail and PayPal. The policy behind the Librem One services is terse, and to the point:

No Ads

We do not have advertising.

No Tracking

We do not track you.

We Respect You

We do not sell nor share anything. We do not build personal, social, nor behavioral profiles of people. We do not alter nor shape user behavior. We do not target people. We do not control people. [3]

Configuration

The Librem One services are lightly documented, but simple enough that most early users should be able to set them up with no more than minor difficulties.

What has been released is Android and



Figure 1: Librem One’s ad is at odds with the services offered.

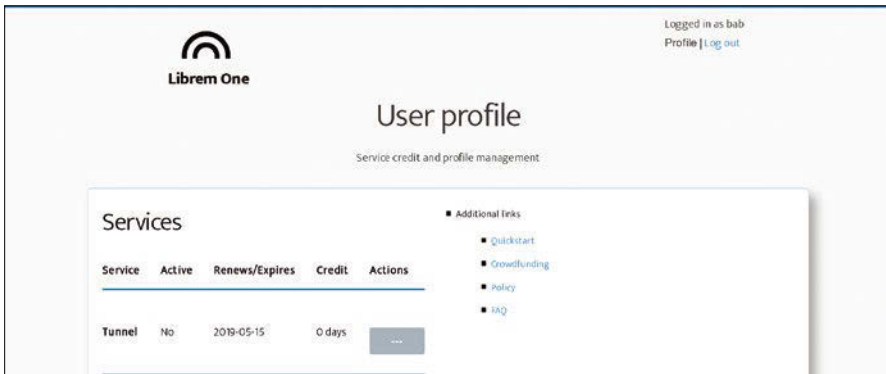


Figure 2: Librem One's user profile also includes some basic help.

iOS versions of the services, which allow users not on Purism products to take advantage of them.

To set up an account, go to <https://librem.one/accounts/login/> and set up a user profile (Figure 2). After creating a login name and passphrase, you are taken to a summary of account activity. In order to get the full experience, before going further, you might want to read the FAQ posted to the right of the account activity, which includes links to instructions about how to delete services like Facebook and Twitter.

Even more importantly, have a look at the Quickstart on the same page. It is the only documentation that Librem One currently has, aside from snippets of on-line help. The most important takeaway is that you must log in separately to each service to configure it, a tedious detail that could surely be replaced by a centralized login with options for those who do not wish to activate all the services.

Still, configuring each service does take users on a tour of Librem One, which serves as an introduction to what is available. The tour is worth taking, because the interface designers have taken some time to design a minimalist interface with well-positioned buttons. In fact, I would go so far as to say that the interfaces are among the cleanest I have seen on any mobile device.

Along with the apps, you may also want to download the Librem One Hub (Figure 3), so that all apps display in the same window. In addition, if you plan to encrypt, you should download OpenKeychain so that you create keys.

Librem Social

One of the best places to start becoming acquainted with Librem One is Social (Figure 4). Social is a Mastodon server [4]

based on Tusky [5] – that is, it is an interface that provides a decentralized replacement for sites like Facebook. Start with a search for Librem, to give you a taste of how messages are displayed, and the different views you can have of a feed. As I write, Social is lightly populated compared to Facebook, but you can encourage friends to subscribe to a free account so that you can communicate with each other.

You can customize your Social display in a limited number of ways, including adjusting the text size, turning on an indicator for bots, requiring followers to be approved by you, and filtering the information displayed with each post. As always with instances of Mastodon, I wonder how Social would scale if it ever became as widely used as Facebook, but that is a problem for the future. For now, though, Social is a much better layout

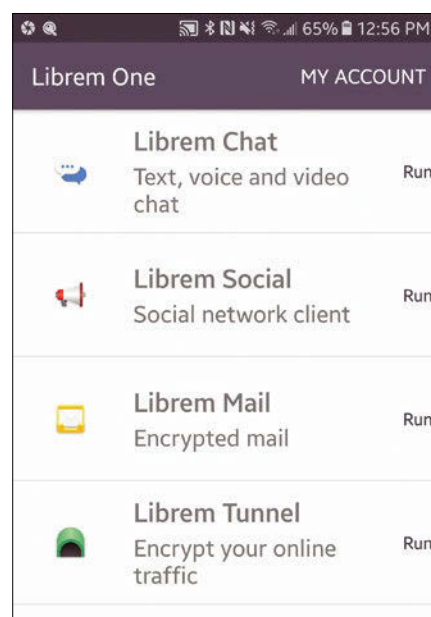


Figure 3: The hub is optional, but useful on the limited screen of a phone.

than Facebook's Messenger and is likely to remain so for some time to come.

Librem Chat

Despite a similar purpose, Chat (Figure 5) has an entirely different interface than Social, since it is based on Matrix [6]. However, this inconsistency does not seriously affect usability.

Like Social, Chat is a useful place to start exploring Librem One, thanks to the existence of numerous chat rooms. *Librem One General Chat* is an obvious place to start, although there are rooms for numerous social causes and technical discussions as well, many with several hundred users. These can be found in the *Browser* directory, which opens when you open the search field.

Librem Mail

Mail (Figure 6) is adequate for basic email. It supports attachments and can draw upon a device's *Contact* listings for recipients, as well as add custom signatures and manage multiple identities. Basic configuration is no harder than entering your Librem One name and passphrase and the composition window could not be simpler.

However, I would be reluctant to rely on it for my daily communication, be-

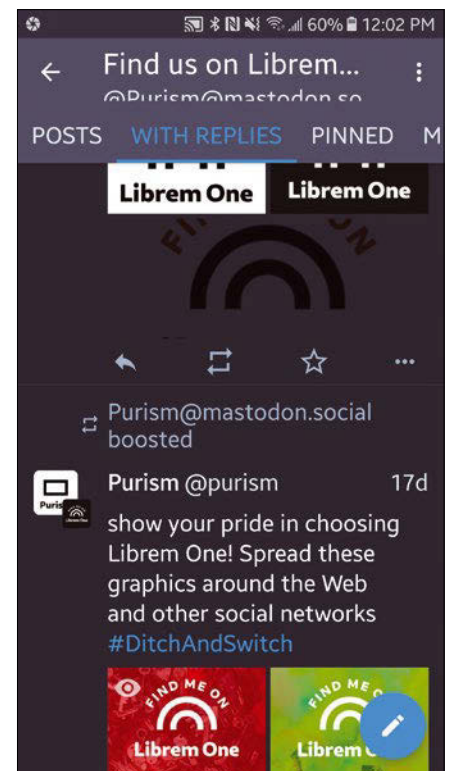


Figure 4: Social is a Mastodon server for social media.

cause it currently lacks the ability to filter messages into different folders. You cannot, for example, set all the messages from a mailing list to download into a designated folder.

Even more importantly, while end-to-end encryption is available, it is

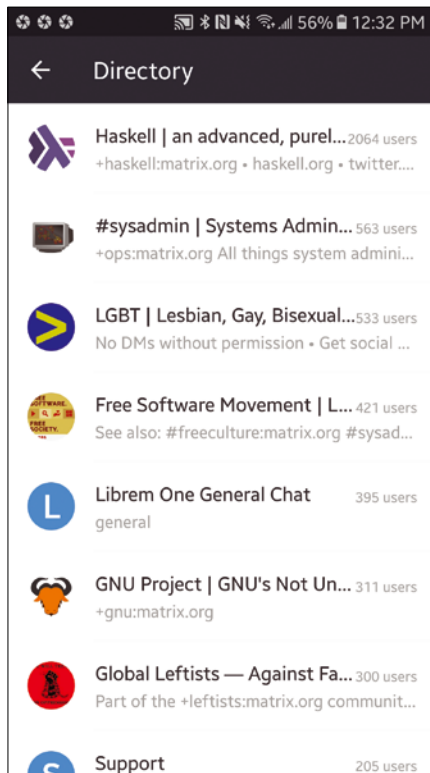


Figure 5: Chat features both individual chats and chat rooms.

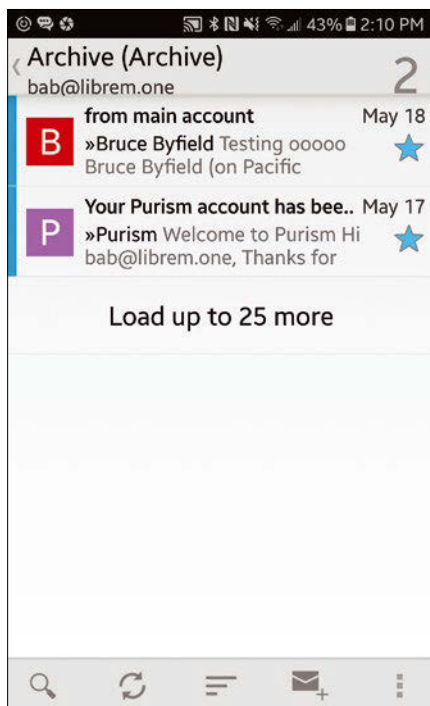


Figure 6: Mail is best-suited to small volumes of mail.

needlessly complex. First, you have to create your keys in OpenKeychain [7] (Figure 7). Then you have to activate encryption in three separate places to configure Mail generally. Should you discover one of these places in the wrong order, you will need to fumble around to find the others. Moreover, once you have enabled encryption, the interface assumes that you know the basics of how to use it. Considering that encryption is advertised as one of the features of Librem One, requiring users to jump through these hoops defeats the point of having encryption in the first place.

Librem Tunnel

Anyone who has used a Virtual Private Network (VPN) can appreciate the security it supplies. However, configuration

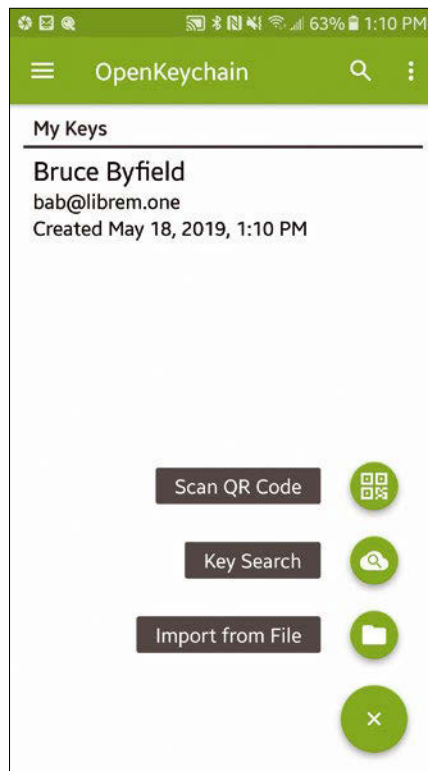


Figure 7: Install OpenKeychain if you want encryption.

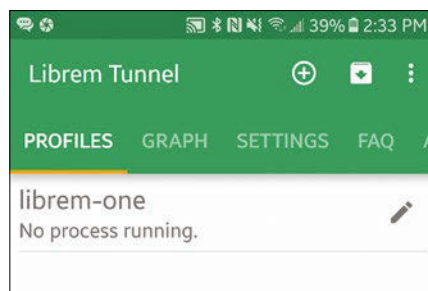


Figure 8: Tunnel manages VPNs, but not conveniently.

can be difficult, which is why I had high hopes for Tunnel (Figure 8). While I usually prefer command-line configuration, a graphic interface seemed exactly what was needed to make setting up a VPN quick and easy.

Unfortunately, if I assume the persona of a novice user, Tunnel offers few of the advantages that justify the use of a graphical interface. Help is hidden in an FAQ in the middle of the top-level menu, and begins with “Get a working config (tested on your computer or download from your provider/organization).” In the very first sentence, novice users are left behind, and the FAQ only gets worse from there. Few novices are likely to get around to the Graph or Settings menus, and, if they do look out of curiosity, they are not going to be any the wiser. At the most, novices might manage through trial and error to connect to online storage, but even that process will probably be none too clear. The result is that what could be the star app is all too likely to be the least used.

Potential Still to Come

I like the idea of Librem One immensely. It has the potential to play an influential role in spreading the use of open hardware. However, even allowing for a first release, the implementation seems rough and inconsistent – especially if the point is to bring security and privacy concerns to the general user. The current release has promise, but that promise is only partly realized.

Still, Librem One is worth watching to see how it develops. It should be especially interesting to see how Librem One compares to the similar set of tools being developed by Gaël Duval’s /e/ Foundation [8]. ■■■

Info

- [1] Librem One: <https://librem.one>
- [2] Ad: <https://www.youtube.com/watch?v=V0a03NRpX3Y>
- [3] Librem One policy: <https://librem.one/policy/>
- [4] Mastodon: <https://mastodon.social/about>
- [5] Tusky: <https://tusky.app/>
- [6] Matrix: <https://matrix.org/blog/index>
- [7] OpenKeychain: <https://www.openkeychain.org/>
- [8] /e/ Foundation: <https://e.foundation/>

Shop the Shop

shop.linuxnewmedia.com

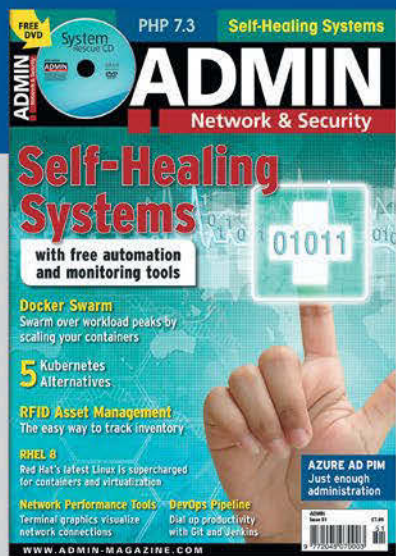
Want to subscribe?

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

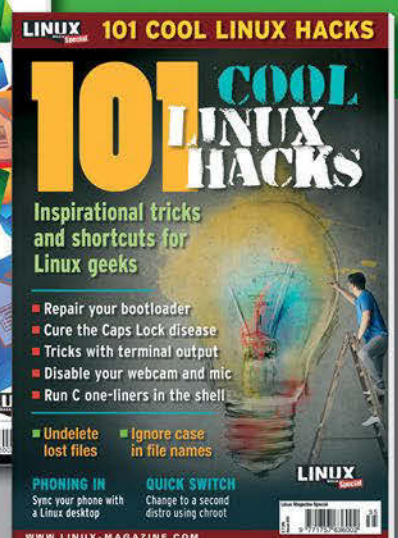
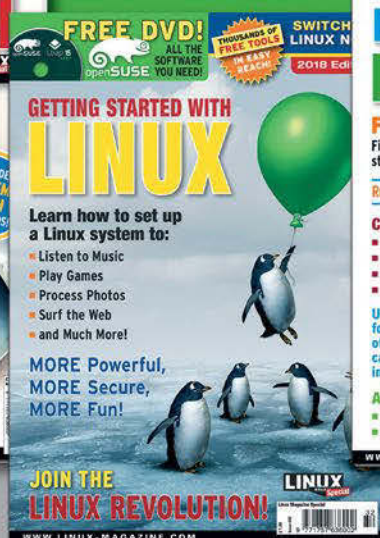
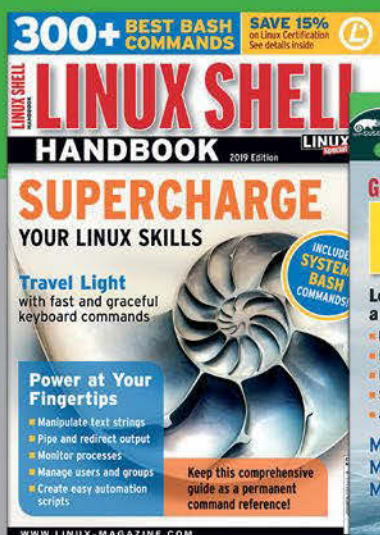
Discover the past and invest in a new year of IT solutions at Linux New Media's online store.

▶▶ shop.linuxnewmedia.com ◀◀

DIGITAL & PRINT SUBSCRIPTIONS



SPECIAL EDITIONS





MakerSpace

Go programming on a Raspberry Pi Way to Go

We show you how to create a Go web app that controls Raspberry Pi I/O. *By Pete Metcalfe*

Go, or GoLang [1], is used widely for web development. Created by Google in 2009, Go is one of the fastest growing programming languages. One of its advantages is speed: Go compiles directly to native executable files in Windows, macOS, iOS, and Linux operating systems. However, Go can also be run in interpreted mode, like Python, while debugging. In this article, I create a Go web app that talks to the Raspberry Pi hardware.

Getting Started

To install Go on a Rasp Pi, enter:

```
$ sudo apt-get install golang
```

To test the install, you can check the Go version number:

```
$ go version
go version go1.7.4 linux/arm
```

A simple “Hello World” Go program named `hello.go`,

```
package main
func main() {
    println("Hello World");
}
```

can be run in interpreted mode with:

```
$ go run hello.go
Hello World
```

Running in interpreted mode is handy when you are doing a lot of debugging, but the compiled Go program runs considerably faster. To compile and run the `hello.go` example, enter:

```
$ go build hello.go
$ ./hello
Hello World
```

Now that the Rasp Pi has Go installed and working, the next step is to set up the Rasp Pi hardware.

Raspberry Pi Setup

Raspberry Pi connects to sensors and hardware through the general purpose input/output (GPIO) pins. For my test setup, I connected an LED to physical pin 7 (Figure 1) and a 330-ohm resistor to prevent damage to my Rasp Pi.

By default, all the Rasp Pi GPIO pins are set as inputs, but you can configure the mode of pin 7 to output then set the output (light the LED), and read back the pin status – all with the `gpio` command-line utility:

```
$ gpio mode 7 output 2
# set pin 7 as an output
$ gpio write 7 1 # 1=set, 0=reset
$ gpio read 7
1
```

The `gpio write 7 1` and `gpio write 7 0` commands let you test the circuit and toggle the LED on and off.

Go and GPIO

A number of Go libraries and options can read and write to GPIO pins. For this example, I shell out (i.e., spawn programs through an intermediate shell) to the `gpio` command-line utility. For prototyping, I like this approach because I can test and verify the GPIO commands manually before writing any code.

Listing 1, `writopin7.go`, uses the `os/exec` library to create the shell command variable `testCmd` (line 10) that defines the `gpio` command and its parameters. The shell command is executed by `testCmd.Output()` (line 11). The `gpio write` command has no output, but the `gpio read` (defined in line 18) returns the state of the GPIO pin (line 24).

To run the code in interpreted mode, enter:

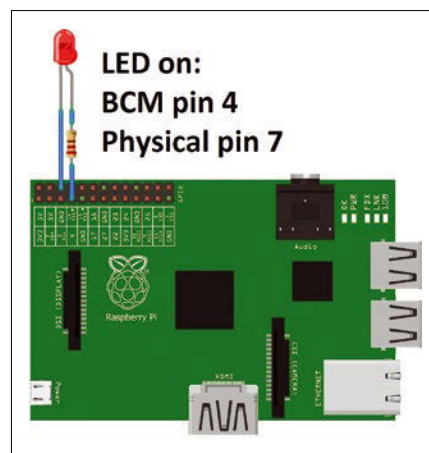


Figure 1: Rasp Pi LED setup.

Listing 1: writepin7.go

```

01 package main                15     }
02                               16
03 import (                     17     // Read back the GPIO pin 7 status
04     "os/exec"                18     testCmd = exec.Command("gpio","read", "7")
05 )                             19     testOut, err = testCmd.Output()
06                               20     if err != nil {
07 func main() {                21         print("Error on read, error:")
08                               22         println(err)
09     // Write to GPIO pin 7 to light LED  23     } else {
10     testCmd := exec.Command("gpio","write", "7", "1")  24         print("GPIO Pin 7 value : ")
11     testOut, err := testCmd.Output()    25         println(string(testOut))
12     if err != nil {            26     }
13         print("Error on write, error:")
14         println(err)          27 }

```

```

$ go run writepin7.go
GPIO Pin 7 value : 1

```

If the circuit is wired correctly, the light should be on.

Simple Go Web App

The `web_static.go` web application in Listing 2 shows the static web page in Listing 3. The web app uses the `net/http` library to define a default handler (line 9), that calls the static page (line 10). The app then listens and serves up data on port 8081 (line 14).

To compile and test `web_static.go`, enter:

```

$ go build web_static.go
$ ./web_static
Default Web Page

```

When you use a web browser to call the Rasp Pi address at port 8081 (Figure 2), you should see the message *This is a*

Listing 2: web_static.go

```

01 package main
02
03 import (
04     "net/http"
05 )
06
07 func main() {
08     // Create default web handler, and call the static web page
09     http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
10         http.ServeFile(w, r, "web_static.html")
11         println("Default Web Page")
12     })
13     // start listening on port 8081
14     http.ListenAndServe(":8081", nil)
15 }
16 }

```

static test page from the Go application every time the page is called.

GPIO Web App

The final step is to pull everything together so that a web page can pass parameters to a Go app that shells out to the `gpio` command-line utility. To begin, you create new web page (`go_buttons.html`) with two buttons (Listing 4). HTML anchor tags pass the `/on` and `/off` parameters to the web app.

A `CACHE-CONTROL` meta tag set to `NO-STORE` ensures that the web page always refreshes. If you don't include this meta tag, the web page might only update once.

The Go GPIO web app `go_buttons.go` (Listing 5) now includes two additional `http.HandleFunc` handler functions: one for the `/on` parameter (line 30) and one for the `/off` parameter (line 37). These handler functions pass the required pin state to a function called `gpio`.

The newly created `gpio` function acts very much like the earlier `writepin7.go` code (Listing 1), in that it shells out twice to the `gpio` command-line utility: the first time to write a value and the second time to read the value back. Web page requests, GPIO pin results, and errors are shown in the terminal.

The syntax to compile and run the `go_buttons.go` code with some sample web requests is:

```

$ go build go_buttons.go
$ ./go_buttons

```

```

Default Web Page
Web Page sent : ON
GPIO Pin 4 value : 1

```

Listing 3: web_static.html

```

01 <html>
02   <head>
03     <title>GO PI Static Page
04     </title>
05   </head>
06   <body>
07     <h1>GO PI Static Page</h1>
08     <hr>
09     This is a static test page
10   </body>
11 </html>

```

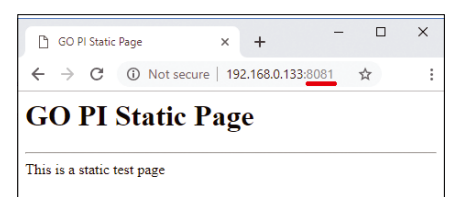


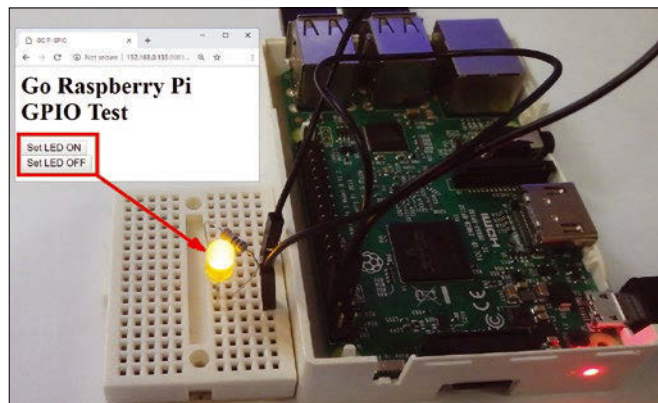
Figure 2: Go web app with a static page.

Listing 4: web_buttons.html

```

01 <html>
02 <head>
03 <title>GO PI GPIO</title>
04 <META HTTP-EQUIV="CACHE-CONTROL" CONTENT="NO-STORE">
05 </head>
06 <body>
07 <h1>Go Raspberry Pi GPIO Test</h1>
08 <a href="/on"><button>Set LED ON</button></a><br>
09 <a href="/off"><button>Set LED OFF</button></a>
10 </body>
11 </html>

```

**Figure 3:** Go Rasp Pi GPIO web app and setup.**Listing 5: go_buttons.go**

```

01 package main
02
03 import (
04     "net/http"
05     "os/exec"
06 )
07 func gpio( pinval string) {
08     testCmd := exec.Command("gpio","write", "7", pinval)
09     testOut, err := testCmd.Output()
10     if err != nil {
11         println(err)
12     }
13     testCmd = exec.Command("gpio","read", "7")
14     testOut, err = testCmd.Output()
15     if err != nil {
16         println(err)
17     } else {
18         print("GPIO Pin 4 value : ")
19         println(string(testOut))
20     }
21 }
22
23 func main() {
24     // Handler for the default Web page
25     http.HandleFunc("/", func(w http.ResponseWriter, r *http.Request) {
26         http.ServeFile(w, r, "go_buttons.html")
27         println("Default Web Page")
28     })
29     // Handler for a "/on" parameter. Send a 1 to gpio
30     http.HandleFunc("/on", func(w http.ResponseWriter, r *http.Request) {
31         http.ServeFile(w, r, "go_buttons.html")
32         println("Web Page sent : ON")
33         gpio("1")
34     })
35 }
36     // Handler for a "/off" parameter. Send a 0 to gpio
37 http.HandleFunc("/off", func(w http.ResponseWriter, r *http.Request) {
38         http.ServeFile(w, r, "go_buttons.html")
39         println("Web Page sent : OFF")
40         gpio("0")
41     })
42     //
43 http.ListenAndServe(":8081", nil)
44 }

```

**Figure 4:** Raspberry Pi rover.

```

Default Web Page
Web Page sent : OFF
GPIO Pin 4 value : 0

```

Figure 3 shows the web output and the test setup.

Summary

Go offers a robust platform for web applications. From the basic web page example in this article, I created more advanced Rasp Pi projects, such as the rover shown in Figure 4 that required writing to multiple GPIO pins.

For a polished application, I would rather use a native Go library for GPIO calls, but for prototyping and troubleshooting, I found the `gpio` command-line utility was very useful.

The next step for my Rasp Pi projects will be to add JavaScript and Ajax to show dynamic values. ■■■

Info

[1] Go programming language:
<https://golang.org/>

Author

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

For many of us mortals, the two greatest challenges are managing money and managing time. We tackle both these pressing issues in this month's edition of LinuxVoice. You'll find out how to keep your bank account balanced with the classic banking tool GnuCash. We also show you Fanurio, a time-tracking tool oriented for the professional user with advanced features for reports, invoicing, and other tasks pertinent to the enterprising freelancer.

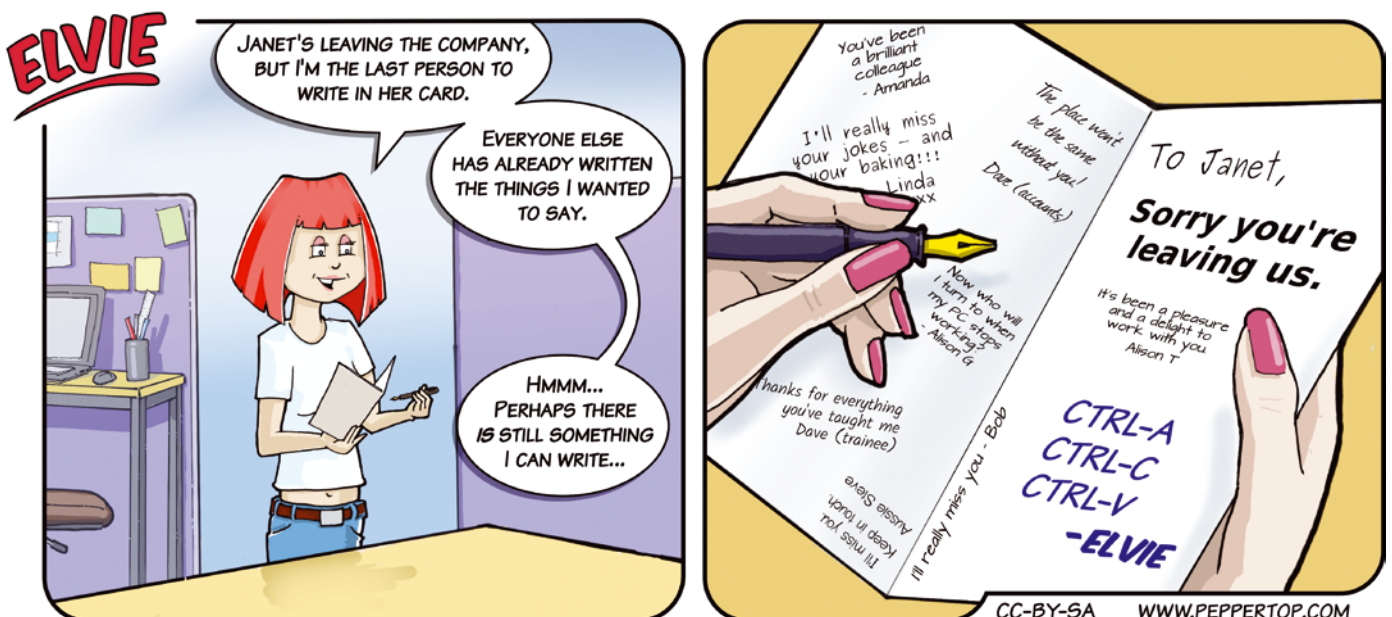
Elsewhere in this month's LinuxVoice, we conclude our series on Bash scripting with a look at incorporating user input into Bash scripts, and our study of 3D printing continues with an article on preparing your design for printing.



Image © Olexandr Moroz, 123RF.com

LINUXVOICE ▶

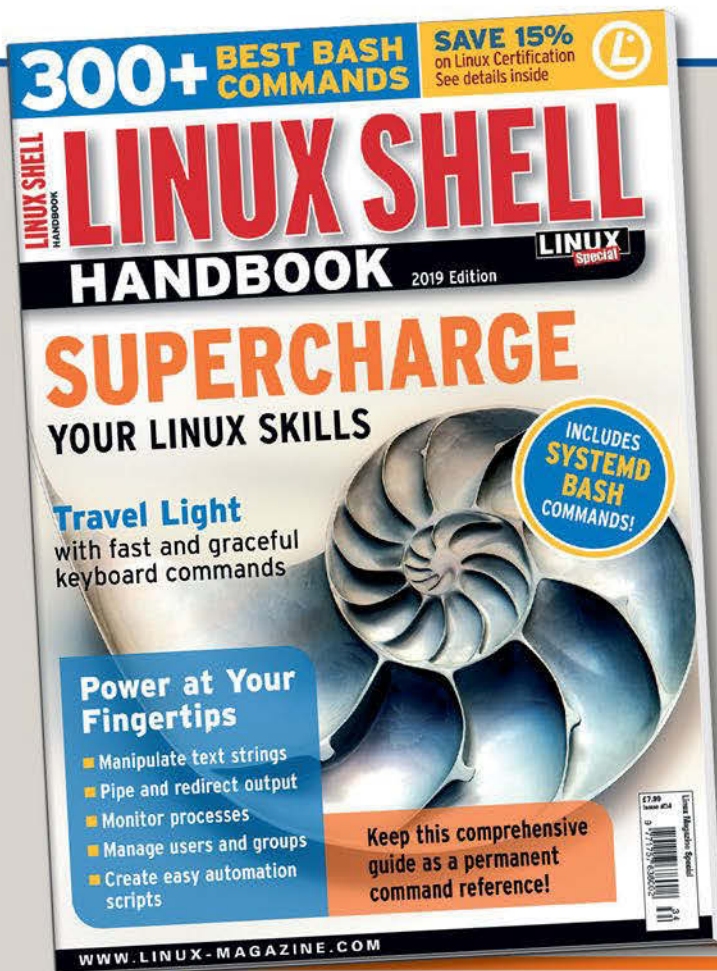
Doghouse – 5G	67
<i>Jon "maddog" Hall</i>	
The recent uproar over Huawei has more to do with intellectual property and 5G than a fear of malware.	
GnuCash 3.5	68
<i>Nate Drake</i>	
Put your personal finances in order or manage your small business with GNUCash 3.5.	
Fanurio Time Management	72
<i>Erik Bärwaldt</i>	
If you need to accurately measure time spent on individual projects and tasks, you need something more sophisticated than yellow sticky notes.	
FOSSPicks	78
<i>Graham Morrison</i>	
This month we explore Olivia, DIN Is Noise, Kaidan, termshark, Worldview, Snipes, and more!	
Tutorials – Shell Scripts	84
<i>Marco Fioretti</i>	
Letting your scripts ask complex questions and give user feedback makes them more effective.	
Tutorials – 3D Printing	90
<i>Paul Brown</i>	
Paul Brown looks at how to prepare your design for 3D printing.	



EXPERT TOUCH

After selling out in 2018, the new *Linux Shell Handbook* is available now! The 2019 edition is packed with utilities for configuring and troubleshooting systems.

**2019
Edition!**



The *Shell Handbook* provides a comprehensive look at the world inside the terminal window. You'll learn to navigate, manipulate text, work with regular expressions, and customize your Bash settings.

Exclusive Bonus: Each issue includes a special discount saving you **15% off LPIC-1 certification** from Linux Professional Institute (LPI). Look for the special code inside your copy!

Here's a look at just some of what you'll see in the new Shell Handbook:

- Customizing Bash
- Pipes and Redirection
- Regular Expressions
- Text Manipulation Tools
- Systemd
- Bash Scripting
- Networking Tools
- And much more!

Make the *Shell Handbook* your permanent desktop reference on the world of the terminal window.

ORDER ONLINE:
shop.linuxnewmedia.com/specials

MADDOG'S DOGHOUSE



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®.

The recent uproar over Huawei may have more to do with intellectual property and 5G than a fear of malware. BY JON "MADDOG" HALL

The race to true 5G

Recently there has been a lot of news about the large electronics firm, Huawei. They have been accused of generating software in their products (particularly their electronics products) that would allow the Chinese government to spy on other countries. Huawei, of course, denies this and points out that the USA through its various intelligence agencies not only has the capabilities to do this, but in the past has been known to spy on people and governments using policies outside US law.

Now companies like Google have decided to terminate their contracts with Huawei and to deny Huawei access to Android software, the Google Play store, and other Google services. Huawei has responded by telling the world that they have been working on their own operating system for seven years, will be releasing it in October of 2019, and it is much better and faster than Android.

The real danger here is that there will now be three major camps in the mobile world, and this may extend to minor (or even major) differences in communications. Instead of coming closer together, we will move further apart.

Already there are two major camps in mobile operating system design: iOS and Android. Despite Huawei saying that their operating system will be compatible on the application level (and perhaps even allow Apple applications to run on the same platform), there are always issues of compatibility ... and at a minimum it means more work for application vendors to do testing on yet another platform, even if the source code APIs remain exactly the same. And (of course) it means more expense for Huawei and everyone else in delivering applications overall.

Most of this issue is not about Android anyway. It is about the communications infrastructure and whether malware can be inserted into the communications code and particularly 5G. If it were about Android or the Android apps, then people could simply buy their software from a vendor they trust and use applications they trust.

In my opinion, the main thrust of this issue is the fact that Huawei has done a lot of work in 5G, and while companies in the West are still having people use "4.999999G" under various names and brands, Huawei is rolling out their 5G. If Huawei's

implementation becomes the standard for the world, then the West (and particularly the USA) might lose close to a half-trillion dollars in revenue to China.

China's government and industry made the investment in 5G technology. They worked hard to roll it out fast, and they beat the USA to the finish line. It is estimated that the USA is about 18 months behind China in the technology.

Huawei has already announced deals with Russia, Canada, and countries in Europe to deliver 5G. The United States is in danger of becoming an island, with its own version of 5G perhaps not interoperable with that of China, Russia, Canada, and others.

There are also the perceived trade issues around China and their "theft" of intellectual property. While China does have a different viewpoint on intellectual property than the West does, they are certainly not alone in this view.

I find it more than a little ironic that while intellectual property battles have happened with the USA in the past, a classic story was the theft of intellectual property that happened when Francis Cabot Lowell visited the United Kingdom from 1810 to 1812 and stole the intellectual property that allowed the United States to move from being an agricultural nation to an industrial nation.

The United States does not really talk about Mr. Lowell and his theft of intellectual property from the United Kingdom, nor do we talk very much about how we took our intellectual property over to China to exploit their inexpensive work force and continue to do so.

The issue of Huawei putting malware in their source code would be relatively easy to fix if that were the real issue. All Huawei would have to do is allow their customers (major telecommunications, utility, and IoT firms around the world) to be able to inspect the source code and build the firmware and binaries to run on Huawei's equipment. Then the level of trust moves from Huawei to whoever distributes the equipment. Huawei could protect their IP the same way we did in the days before software copyright and patents, through contract law.

If, on the other hand, the real issue is the loss of 500 billion dollars in profits because Huawei built a better 5G, all the open source software and hardware in the world will not help to solve that problem. ■■■

Bookkeeping the FOSS way

Smart Money

Put your personal finances in order or manage your small business with GnuCash 3.5, a free and open source software program designed to handle both basic and complex accounting data.

BY NATE DRAKE

If you're starting your own business or want to take greater control of your personal finances, you may have been tempted to subscribe to one of the many popular proprietary (and sometimes expensive) accounting products on the market such as Sage and Intuit QuickBooks. There is some sense in this: If you choose a less well-known product that is later discontinued, all your financial data could become unreadable overnight. Many of the freeware finance programs also don't support more advanced features like double-entry bookkeeping.

GnuCash offers a ready solution to both these dilemmas. As free and open source software, there's far less risk of support being discontinued altogether, and it can open and save transaction data in a number of formats. Despite being free of charge, the program is suitable for complex accounting data and supports double-entry bookkeeping, multiple currencies, and credit card transactions out of the box.

GnuCash 3.5 [1], which is the focus of this review, contains a truly staggering number of features. We have focused here on setting up accounts and inputting transaction data, but we encourage readers to take the time to explore the program's extra functions, too (see the "Other Features" section).

A Little Background

GnuCash was first developed in 1997. The first stable release was in 1998. It was initially designed to imitate some of the features found in Intuit QuickBooks but soon evolved and began to offer more. There are versions for Windows and macOS, as well as Linux (Figure 1). An Android version was released in 2015.

GnuCash stores all data you enter in the flexible .xnm1 format. It is also one of the few bookkeeping tools to support multiple currencies, and GnuCash developers claim it is the only open source program of its kind.



Figure 1: Download GnuCash from the application's main page.

The software can track your finances any way you see fit. If you just want to balance your checkbook, GnuCash has that covered. You may want to do more in-depth bookkeeping, such as tracking credit card spending, assets, and liabilities. While GnuCash is perfect for personal use, it is a very powerful tool for business users. Users can also track investments and loan payments.

Installing GnuCash

GnuCash comes preinstalled in certain distributions of Linux. Failing this, there are several different ways to install GnuCash. Ubuntu users can navigate to *Ubuntu Software*, type in *GnuCash*, and click on the results (Figure 2). From here, press *Install*. Once the installation is complete, click *Launch* to open. The GnuCash website [1] also contains instructions for installing the program in other popular distributions.

Each time you open the application, a pop-up box with a tip appears. If you don't want to see this, just uncheck the box underneath the message.

Setting Up Main Accounts

Once install is complete, you can set up your main GnuCash account. The software is suitable for both personal and business accounting.

It is best to set up five principle accounts. These include *Assets*, *Expenses*, *Income*, *Liabilities*, and *Equities*. Every transaction you make can be categorized under these headings. This makes your finances easier to manage.

You can simplify setup by using the GnuCash wizard. Although this is a fast way to get started, some users may find there are way more accounts and sub-accounts than needed. The sheer number of accounts can be overwhelming at first,

but they can easily be unchecked and removed from your books.

To get started, open GnuCash. Click on *File* | *New File*. From here, you can choose whether to use the wizard or enter your information manually. A pop-up box appears called *New Account Setup Hierarchy*. Click *Forward* to use the setup wizard.

Next, select the currency you wish to use in your accounts, and then press *Forward*. The next page relates to *New Book Options*, and there are four tabs to go through: *Accounts*, *Budgeting*, *Business*, and *Counters*. The setup wizard can walk you through all these.

However, more adventurous users can click *Cancel* when the setup window launches and enter information manually. Once you have cancelled, ensure that you can see *Accounts* on the left-hand side of the page. If you can't see this, go to *View* | *New Accounts*.

Next, click on *Actions* and scroll to *New Account*. From here, you can start entering your main accounts. We started with *Assets* as the first account typed, checking it was set to our currency of choice, US dollars (Figure 3). Next, under *Account Type*, we selected *Assets*. You can also choose to set a different color for each account, enter an account code, description, and any notes about this account.

Setting Up Sub-Accounts

From looking at your current screen on GnuCash, you can see that it is not worth much to you as is. So in order to start getting the most out of the application, you now need to start populating it with sub-accounts. This can be done the same way as for main accounts except that you click

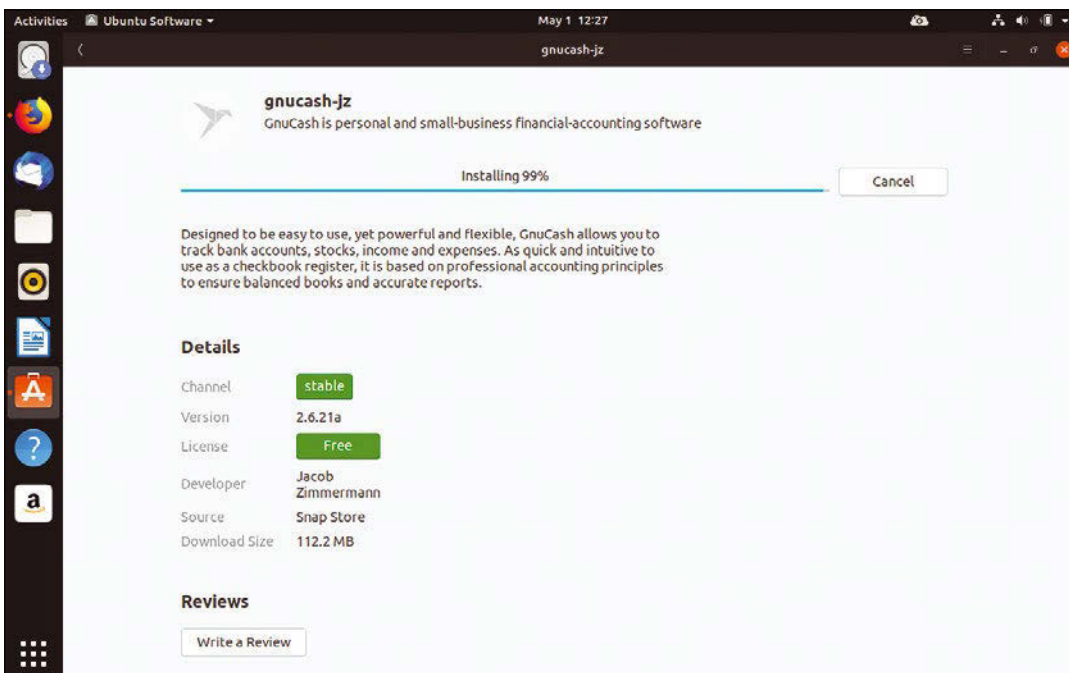


Figure 2: You can use the Ubuntu Software Center to install GnuCash.

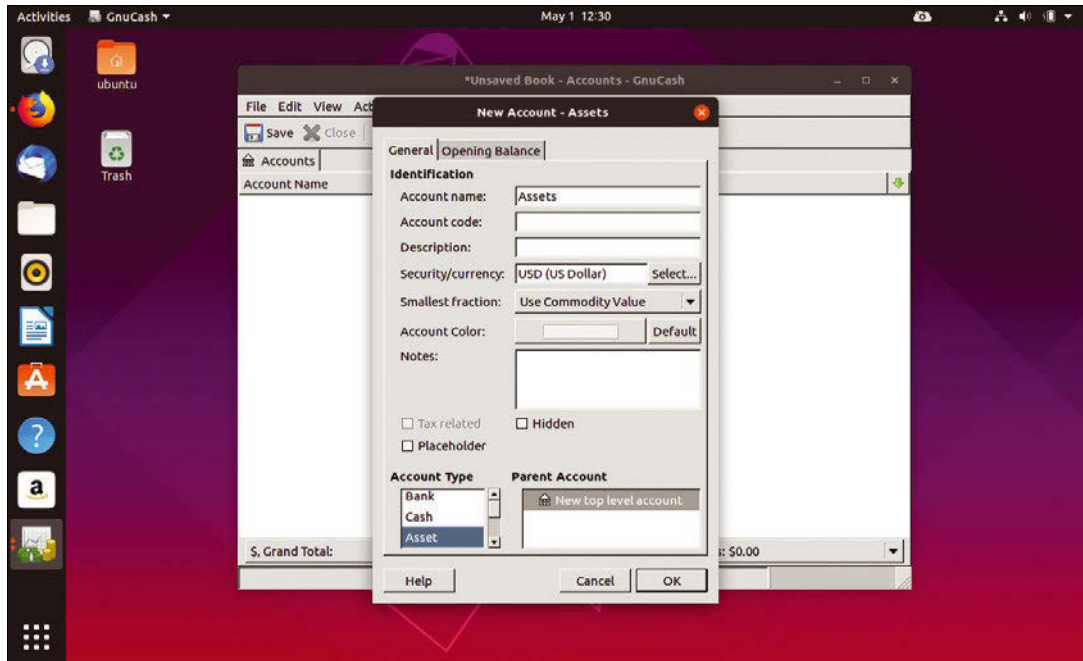


Figure 3: GnuCash includes an account setup wizard. You can also enter your information manually.

the drop-down menu under the *Parent Account Type* section. GnuCash will then show which section this comes under (e.g., *Assets*). When you choose the parent account, depending on your choice, the *Account Type* options also change. At this stage, there is no need to worry about having an exhaustive list of sub-accounts. You can add more at any time by repeating the above steps.

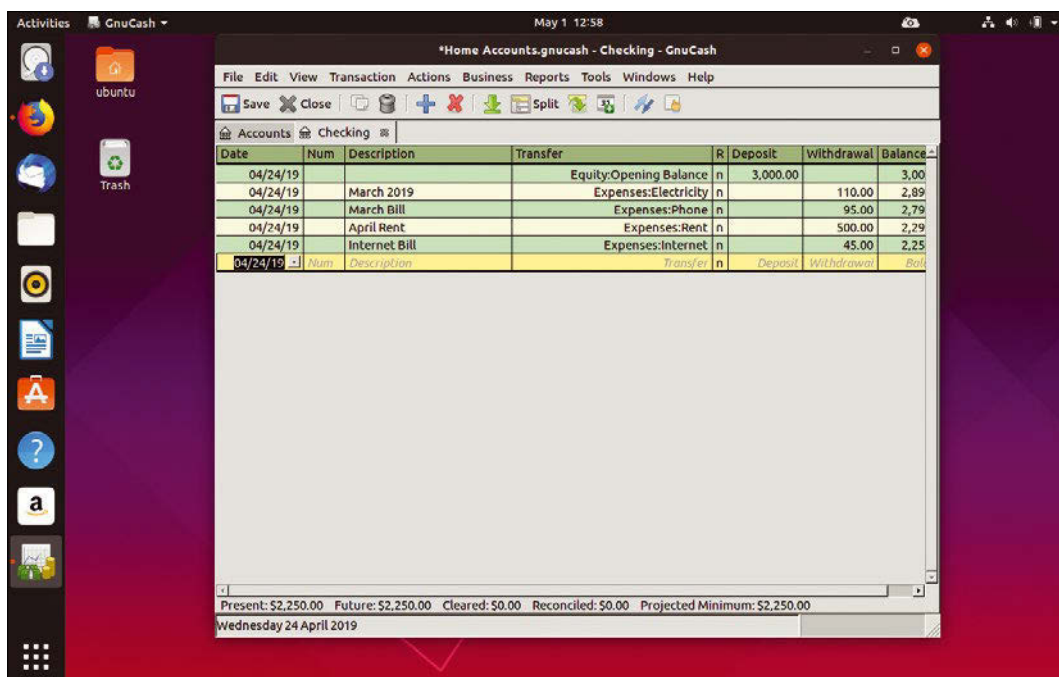
Once all these have been entered, it is now time to add your opening balance. Again, go to *Accounts | New Account*. Input *Opening Balance* as your account name. This should then go under *Equity* in *Parent Account*. This will now be saved as a sub-account on the main page.

Figure 4: You can enter figures by clicking on the exact transaction you want calculated. These will then show up on your main screen.

Inputting Opening Figures

Once all your accounts have been set up, you need to enter your opening figures. When we were doing our accounts, we began by inputting an opening balance under *Savings*. To do this, double-click on *Savings*. A dedicated tab will now open beside your main account page. Enter the amount of savings you have under the *Deposit* section (e.g., \$2,000). Under the drop-down menu in *Transfer*, select *Equity:Opening Balance*. On the *Accounts* tab, you will see there is now \$2,000 in both the *Savings* and *Equity:Opening Balance* sub-accounts.

GnuCash is fairly intuitive in that it calculates the totals in each parent and sub-account de-



pending on the opening balances you enter. For more information on the basics of inputting transactions and reading outputs, read through the GnuCash Tutorial and Concept Guide [2].

Basic Transactions

Once you have all your accounts and sub-accounts set up and your opening balances are filled in, you can begin entering transaction data.

As with most financial software, you'll need a basic grasp of the double-entry system so you can interpret all the data

in the *Accounts* section. When you work with GnuCash, you will always be concerned with at least two accounts. For every change in value on one account, there must be a balancing change in another account. This is known as the principle of balance.

GnuCash divides your data into three organizational levels: files, accounts, and transactions. These are categorized based on their complexity. One file can contain multiple accounts and each account can hold multiple transactions.

Begin by inputting your expenses. Go to an expense (e.g., *Phone*) and double-click. In the selected tab enter the amount and where it is coming from (e.g., *Assets:Checking*). When you go back to your main page, you will see that your *Checking* account has now been debited by the amount entered in the bill. This amount also shows up under *Expenses*.

You can also enter these transactions by going directly to *Checking*. From here, enter your description and in the *Transfer* section and choose the category of expenses. Make sure to enter the amount under *Withdrawal*, not *Deposit*.

You can choose to pay by credit card: This will then be displayed under the *Liabilities* section where you have entered the type of card you use.

At some stage during this process, you may want to pay a bill or input an amount from an account that is not already created while you are in the middle of a transaction. To do this, enter your chosen field (e.g., *Checking*). Fill in the description. Under *Transfer*, fill in an entirely new name rather than clicking on the drop-down menu (Figure 4). Next, press Enter. A pop-up box will appear stating that this account does not exist and asking whether you would like to create it. Click Yes. You can now add this account in the *New Account* box that appears.

Simple vs. Split Transactions

Each transaction that takes place in GnuCash splits at least two ways. This usually just involves a current account and a single remote account. This is known as a “simple” transaction.

However, there are times when your incoming transactions may be divided in numerous ways, and in GnuCash these are known as “split” transactions.

As most of our expenses are taken out of our salary, it is safe to say this is where the majority of split payments occur. If this is the case, start in your *Income | Salary* account and not it in your *Checking* account as you have been doing for the rest of your expenses.

Double-click on *Salary*, and enter description and salary amount. Before finalizing this entry, select *Split* from the top of the screen. Fill in the *Transfer* section but do not press Enter. Click Tab until you reach the next row and add your next

split entry. Continue this until you are finished splitting your paycheck.

Other Features

GnuCash is extremely feature rich, so much so that the application guide runs to almost 300 pages. Although we can't provide a complete list here, some extra features caught our attention.

GnuCash can import transaction data from the SWIFT network and other international banking institutions. Currently the software supports files in QIF, OFX/QFX, CSV, MT940, MT942, and DTAUS formats. Just go to *File* and *Import* to start the process.

GnuCash also has a very handy loan repayment calculator which can add these payments into your account. It can be used to calculate payment periods, interest rates, the present value, periodic payments, and future values.

Another useful feature is GnuCash's reporting function, which allows you to display numerous transactions in a variety of formats. There are several commonly used reports, which can be customized to suit your needs. However, if you feel these are insufficient, you can create your own custom reports.

Summing Up

The main advantage of GnuCash is that it builds upon layers of complexity. Users with little or no bookkeeping experience can use the simple setup wizard to create main accounts, sub-accounts and enter their transaction data within minutes. The GnuCash project's open source nature may also provide some reassurance to business owners who worry about trusting their financial data to private companies that can go bust at any time. Although we focused on the Linux version of GnuCash, the software is cross platform, meaning that financial data can be shared across operating systems, though perhaps not quite as easily as with a cloud-based finance platform like Sage. As the GnuCash website notes, although the program does come bundled with various flavors of Linux, they don't always include the latest version. Make sure to follow the instructions in the *Installing GnuCash* section to enjoy all the latest features. ■■■

Info

- [1] Download GnuCash: <https://www.gnucash.org/download.phtml#distribution>
- [2] GnuCash Tutorial and Concept Guide: <https://code.gnucash.org/docs/C/gnucash-guide.pdf>

The Author

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

Innovative time management with Fanurio

Stopwatch

If you need to accurately measure time spent on individual projects and tasks, you need something more sophisticated than yellow sticky notes. Fanurio helps you keep an eye on your time, your projects, and more.

BY ERIK BÄRWALDT

Efficient time management is an important key to success, especially for busy freelancers, as well as small companies. However, the free time-management programs available for Linux are oriented more toward the private user than the professional; they often simply measure the time you spent working at the computer.

Fanurio [1], developed and distributed by the Romania-based Fanurio Time Tracking SRL, fixes this problem. In addition to actually timing your work, it also supports statistical analysis and includes an invoicing module that allows freelancers to invoice their customers for time worked.

License Models

You can try Fanurio as a 15-day trial version available from the manufacturer's website. The developers provide the software both as a DEB package and as a tar archive that can be used across distributions [2]. If you need an extended trial period, you can email the company for an extension.

The single-user version of the software written in Java costs \$59 for a lifetime license with a year of support; you can extend the support period by another year for \$29.

The server version for one user with one year of support and updates costs \$89 in licensing fees; an upgrade from the standalone version to the server version costs \$30. The server variant for workgroups, which is currently still in the beta phase, will be available for \$119 after release [3].

Installation

After downloading the DEB package for Debian, Ubuntu, Linux Mint, and their derivatives, you can conveniently install the package by double-clicking: GDebi or the Ubuntu Software Center will handle the installation. During the install, the routine automatically creates a starter for calling the program in the `Office` subdirectory.

In a terminal, you can set up Fanurio by entering the command:

```
sudo dpkg -i <package>
```

To do this, first move the tar archive into an empty directory and then unpack it with the command:

```
tar xzf <package>
```

Since no subdirectory is created during unpacking, you will want to unpack the archive in a newly created directory; otherwise, a collection of subdirectories and individual files will be created in the download directory. After unpacking, call the application by entering the `./fanurio1ocal.sh` command in a terminal.

First Launch

The first time you start the program, you are taken to a configuration wizard, which first prompts you for the license or unlocks the trial version. You can then use the wizard to create a data directory in which Fanurio later will store your personal data, such as your name, address, and – if you want to use the billing module – also company details. You then enter the data directly. In another dialog box, you need to select a currency (the default is euros). You can also specify other currencies to be used in the invoice module (Figure 1).

After this, the invoicing module can be activated in another window. This later also influences the software's display functions for the individual tasks that the system captures. The corresponding dialog also lets you enable tax rates; the application can manage up to three different rates for automated VAT calculation later on. Projects and tasks are configured in two additional dialogs. It is also possible to decide whether or not Fanurio should compute travel times and expenses for completing individual tasks.

Clicking on the *Finish* button, bottom right, in the window closes the wizard, which then loads the program's clearly structured main window (Figure 2). At the top, there is a menubar with a small buttonbar

below for quick access to the most important functions. To the right, there is a timekeeper.

Below this, you see three tabs that display your existing projects, tasks, and the timekeeping table in separate context-sensitive windows. If the invoice module is activated, two additional tabs for payment transactions supplement the interface. The next time you launch the software, the main window now opens directly without calling the wizard again.

Master Data

In the next step, you create the master data. To do this, press the button on the far left of the buttonbar to create a new customer. In the separate dialog that now opens, enter all relevant customer data in the individual tabs. Then press the *Create* button. The new customer now appears on the left in the tree view in the main window.

The next step is to enter your first project. Open the dialog for this by clicking on the second button from the left in the buttonbar. Fanurio breaks projects down into individual tasks, which it processes chronologically. Before you create a project, you will therefore need to consider which sub-tasks it includes.

If, for example, you have to travel within the scope of the project, then the travel costs can be recorded directly in the project (if the invoicing module is activated) and later invoiced to the customer. The dialog for entering a project is clear-cut. In addition to the customer, you only need to enter a project name, which can be supplemented with comments in a free text field if required.

If you have already created some customers, select the project's name from a selection list in

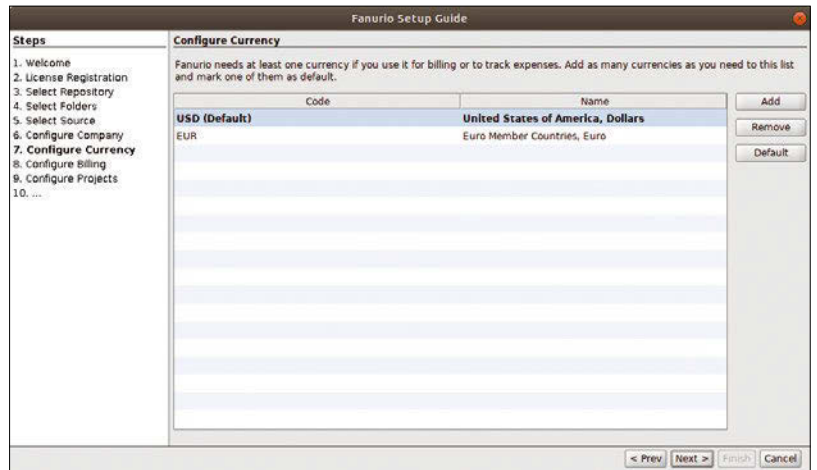


Figure 1: Fanurio can bill in US dollars, as well as in many other currencies.

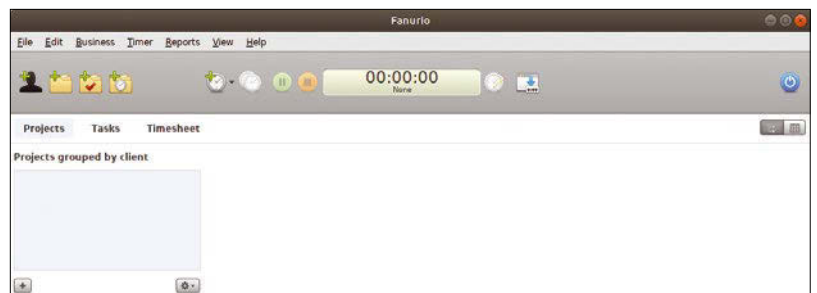


Figure 2: No bells and whistles: Fanurio focuses on a deliberately simple interface.

the project's entry dialog. Then click on the third button from the left in the toolbar to switch to the entry dialog for tasks. There you first enter a name and then select a corresponding category in a selection field, assuming you have created this category during the initial configuration.

If there is no suitable category yet and it seems worthwhile to categorize the task you just en-

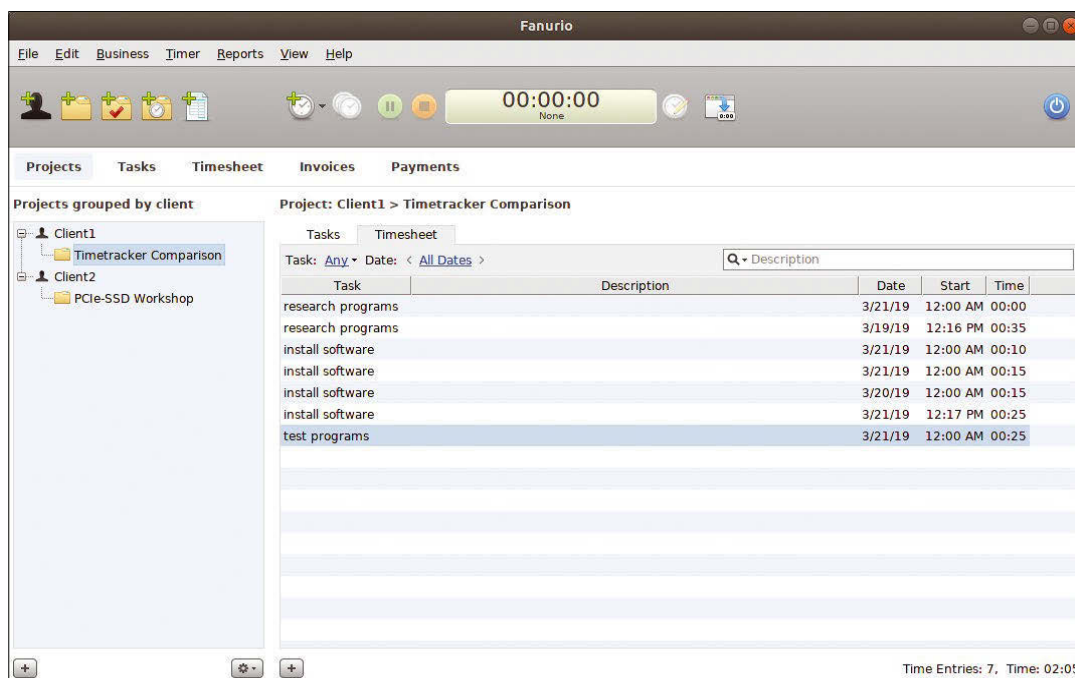


Figure 3: Thanks to several tabs, Fanurio can also be used to manage extensive tasks clearly.

tered, then create a new category with the *New* link to the right of the task. Enter some information about the task in a free text field. Then you can record the time you spent working on the project; first define the start of the task, its duration, and any breaks in the *Time*: line using the *New Time* link.

In the task window, instead of showing the start time, Fanurio shows the duration of the task. In the *Estimated time*: field, enter a time estimate if required. If there is a due date, this can be defined with a mouse click using a displayed calendar. Then click on *Done* in the lower right-hand corner of the window to complete the task entry. In the main window, Fanurio now displays the entered task in a tabular view in the right large segment (Figure 3).

In the *Tasks* tab, Fanurio only shows you the date and the expected time for completion of the task. More detailed information on the individual project tasks can be found in the *Timesheet* tab. This tab also contains a task description, if you entered one, and the scheduled start time for the task.

To enter further time entries, tap on the small **+** button bottom center in the program window below the timesheet. To the right, you can see how many tasks a project contains and how much time has been allocated to complete them.

Stopwatch

If you want to record the times for individual tasks automatically, start the timer by clicking on the

Start New Timer button in the middle of the main window. While you are working on the task, the timer runs in the background. To take a break, press *Pause timer*, and after completing work on the task, click *Stop timer*.

While you are working, the application window can be minimized by clicking on the *Switch to Mini Timer* entry in the View menu for the timer display. Clicking on the enlarge button on the right in the mini timer lets you restore the original program window.

As soon as you stop the timer, Fanurio displays an input window in which you type the necessary information about the task you just completed. Then press the *Add* button, bottom right. Fanurio then adopts the corresponding entry into the timesheet. At the bottom right of the window, the software totals the time for all entries so that you always have an overview of the time spent on the current project.

In the timesheet, Fanurio summarizes the individual processing times for each task. If there are several time entries with the same task name in the timesheet, they are not displayed individually. Instead, the times are totaled for the entry. In this way, you can quickly gain an overview of which tasks are taking longer than expected.

If you open the *Tasks* or *Timesheet* tab instead of the *Projects* tab, you can see the available data over the entire width of the window. The project tree on the left side of the project view, which groups the projects by customer, is not shown in these views.

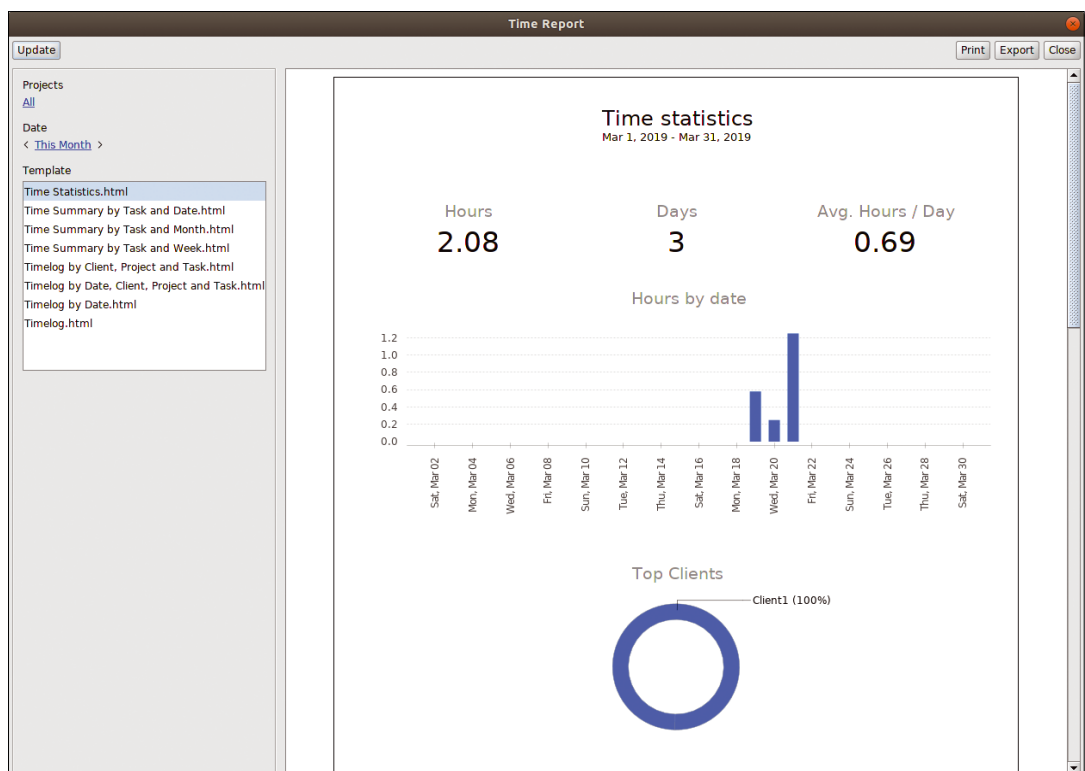


Figure 4: For a better overview and customer presentations, Fanurio creates graphical reports.

However, the *Client*, *Project*, *Task*, *Date*, and *Finish*: links let you filter the task and timesheets, thus improving clarity, especially when working with a team or with many entries in the tables.

Reports

For accounting, Fanurio offers the ability to group activities and their time requirements in reports, print these in a visually appealing way, and save the results in PDF or HTML format so that the reports can be opened across platforms and applications.

In the *Reports* menu, choose one of the available alternatives: *Project Report...*, *Time Report...*, or *Task Report...* In each of these cases, Fanurio opens a new window with a settings bar on the left and the report on the right in the larger window segment (Figure 4).

The application displays the project and task reports in a clearly structured table, while the time report visualizes the collected data graphically in a bar chart and a pie chart. Top right in each report window, there are three buttons that you can use to close the window, print, and export the report.

In the export dialog, you can specify a view option next to the format and location if you have designed your own report templates. By default, Fanurio uses the integrated templates, which can be manually supplemented with customized layouts. Then select the desired template from the *Template*: option's drop-down menu.

Links in the report windows allow Fanurio to limit the selection: Reports can be sorted by project or by date. If there are several templates for a report form, select the desired template from the *Template* list on the left side of the report window.

The Bill Please

Fanurio not only logs the workload during a project, but also supports you in accounting. If you have not enabled the invoice module in the wizard during the initial configuration, you can do this at any time by opening the *Business* menu in the program window and selecting the *My Business Details* entry.

In the *Billing* tab that now opens, check the *Enable billing module* option. Fanurio will then extend the dialog with some configuration options, which you can use to define a payment period and make settings for automatic invoice numbering, for example.

There is also a small table in which you can store items you regularly sell, which can thus flow into the calculations for your projects. The *Projects* tab lets you make further settings. In addition to project numbering, you can define billing units and hourly rates.

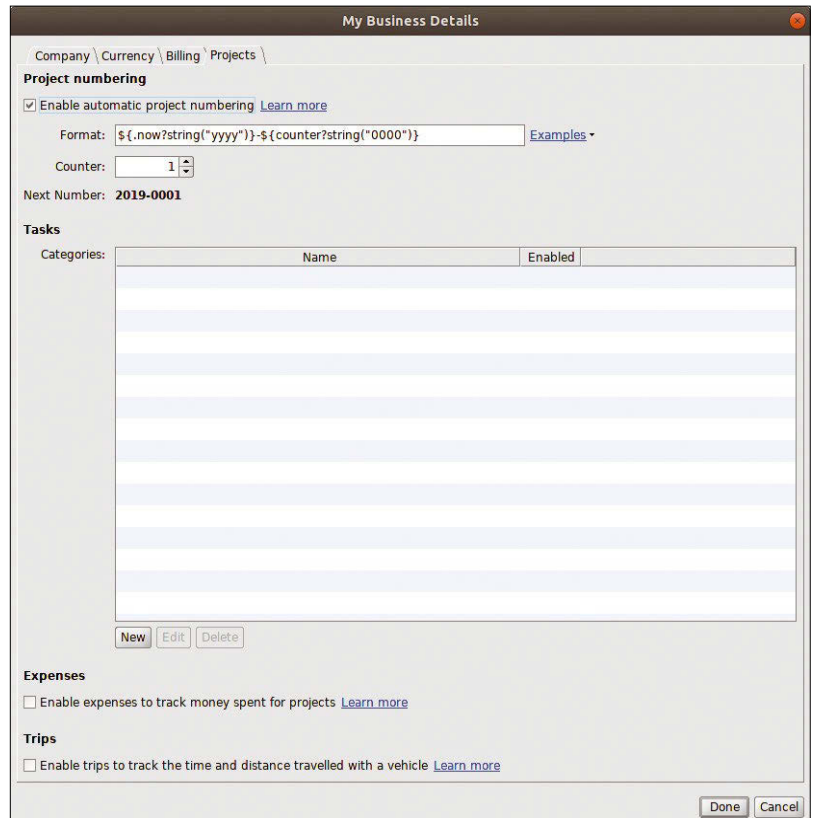


Figure 5: Fanurio offers some options for invoicing customers.

Additionally the *Expenses*, *Trips*, and *Products* groups can be enabled by checking the boxes. This is where you can integrate costs incurred for trips and additional services or goods into the total calculation (Figure 5).

Templates

To print invoices and reports, you have the option of creating your own documents in Fanurio. The application supports numerous formats, with HTML and PDF acting as the standard formats. In addition, the software can also handle XML, CSV, ODT, and ODS files.

If you already use templates in these formats, store them in the corresponding template folder and select them again later. The template folders can be accessed via *File | Open Templates*; Fanurio lists the desired templates to reflect the various report types.

If you have enabled the billing module, you will also find a *Template Editor* item under the *File* menu. You can use this to design an attractive invoice form in a few minutes using a graphical interface (Figure 6). You will find details on this in the software documentation [4].

Data Backup

By default, the software automatically creates regular backups of all the data you create in an application-specific format. You can start a backup manually via *File | New Backup...* Further options for the data backup can be found in the *Restore*

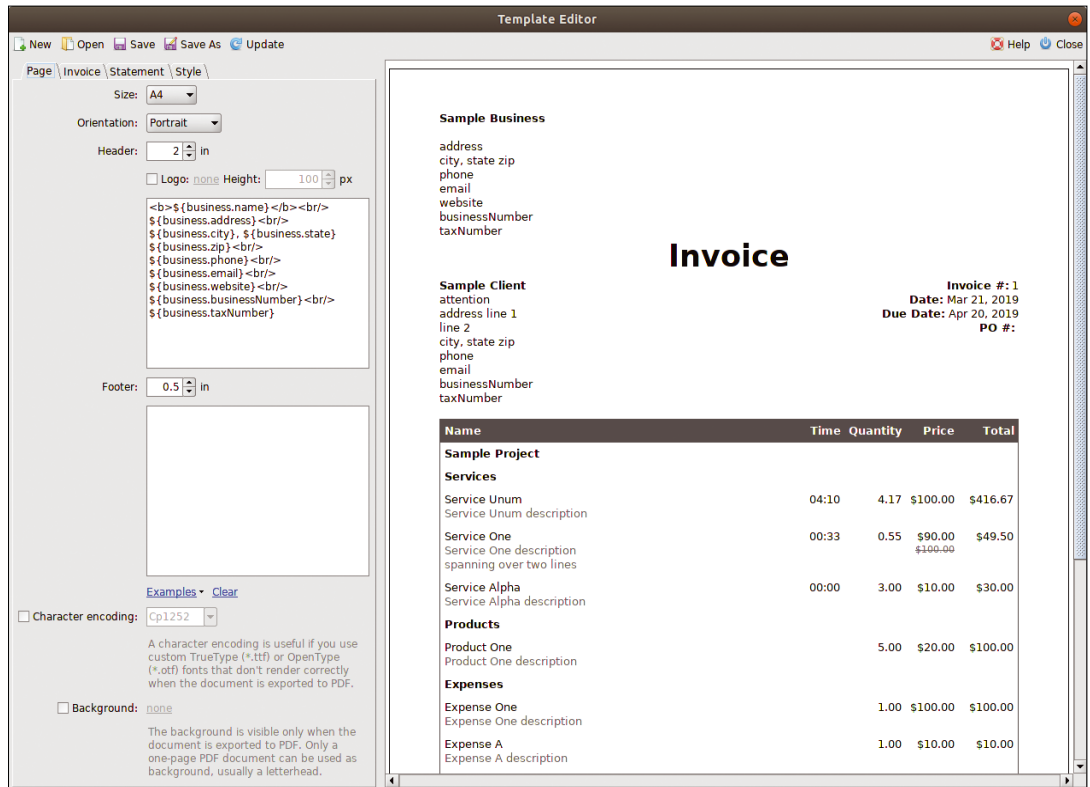


Figure 6: The Template Editor lets you adapt the standard forms to suit your needs.

Backup..., *Open Backup Folder*, and *Change Backup Folder...* menu items.

In the dialog for creating a backup, you can see the automatically created backups. Their file names always start with *auto*, followed by the date and time. Fanurio also saves manual backups with a date and timestamp, but the name of the backup starts with *manual*. This lets you distinguish between automatic and manual backups at a glance.

If you want to change the target folder (e.g., to save the backups to a removable disc) in the future, you can use the *Change backup folder...* option to do so. Enter the new folder path in the dialog that then appears. The best way to tell Fanurio to transfer existing backup files to the new folder is to check the option *Copy files to new folder*.

Import and Export

Some file formats for the transfer of data from other applications have become firmly established across platforms. Fanurio also uses these formats for import and export.

In the *File | Import* menu, you have the possibility to import timesheets and contacts. The option *Import timesheet...* supports CSV as well as Fanurio's own format. Fanurio supports CSV and XLS when exporting customers, tasks, and timesheets, and XML and IIF for timesheets.

Conclusions

Fanurio helps you track your work time more efficiently. The software is appealing due to its

intuitive user interface, functions tailored to the business customer, and simple dialogs and wizards.

The program also handles some small business enterprise resource planning (ERP) functions, which is ideal if you need to add devices and components for projects and tasks for which you would like to bill the customer. However, the focus of the application is clearly on timing and billing for services. With the server version of Fanurio, you can also optimize time management in small workgroups on the intranet by centrally recording and evaluating the times of several employees. This improves the team's workflow and avoids redundancy.

In view of the practical functions and the benefits of the software for freelancers, small companies, or departments, the licensing costs for Fanurio appear to be justified. ■■■

Info

- [1] Fanurio: <https://www.fanuriotimetracking.com>
- [2] Download Fanurio: <https://www.fanuriotimetracking.com/download.html>
- [3] License models: <https://www.fanuriotimetracking.com/purchase.html>
- [4] Documentation: <http://www.fanuriotimetracking.com/files/releases/3.2.1/help/html>

Available Now

* 2018 EDITION *

Linux Magazine Archive DVD

Searchable
Linux Archive:

19,000

Pages of Practical
Know-How

214 issues of **Linux Magazine**
on one handy DVD!

ORDER NOW!

shop.linuxnewmedia.com

Searchable
Linux Archive:

19,000

Pages of Practical
Know-How

18 YEARS OF LINUX!
Issues 1-214

- ▶ All the Tricks
- ▶ All the Hacks
- ▶ All the Apps

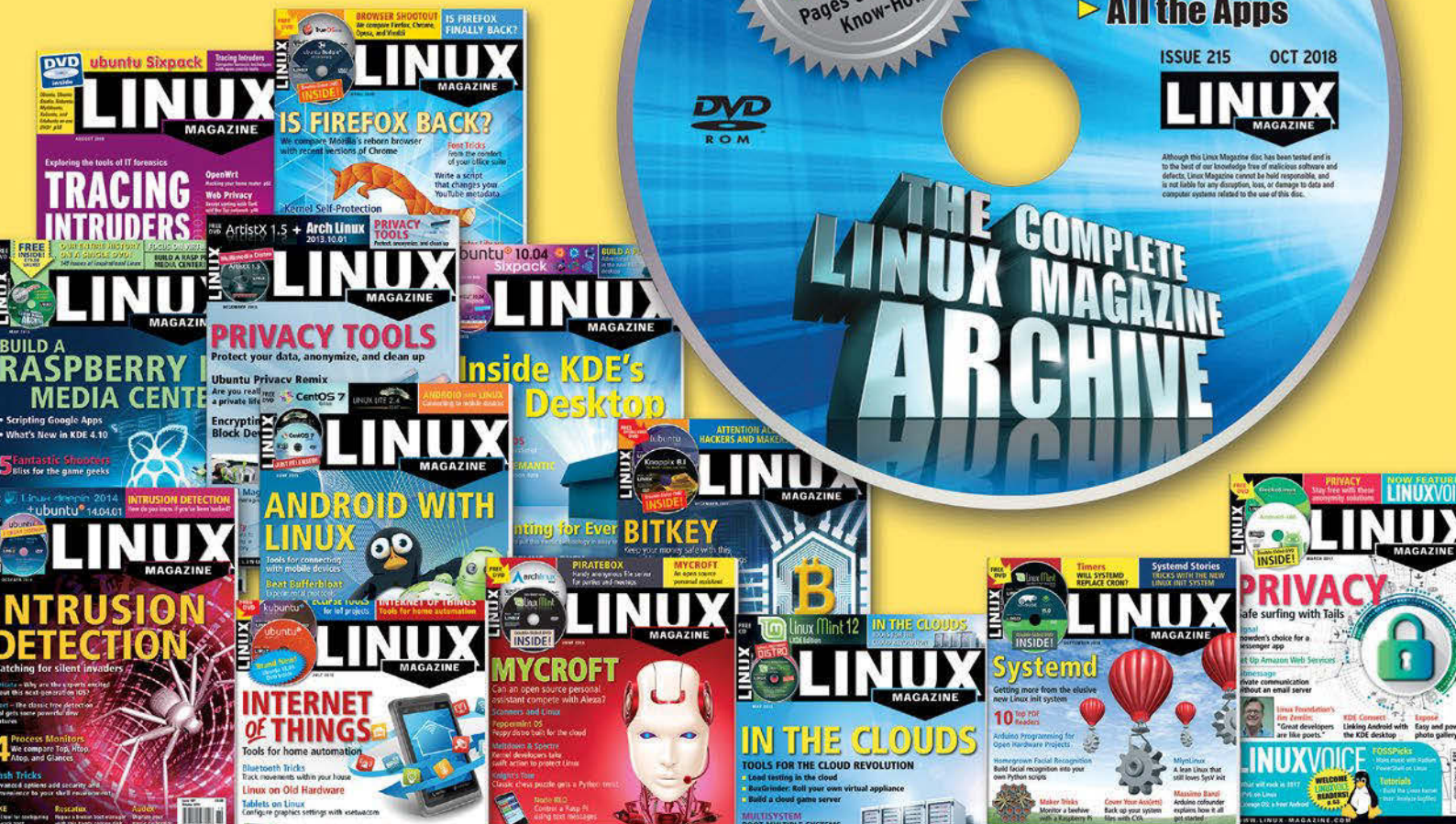
ISSUE 215 OCT 2018

LINUX

MAGAZINE

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

THE COMPLETE LINUX MAGAZINE ARCHIVE



FOSSPicks

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



This month's FOSSPicks was nearly lost in a stream of AES-256 noise, as Graham struggled to get his encrypted Linux installation (using LUKS) to live alongside an encrypted macOS installation (using APFS). **BY GRAHAM MORRISON**

Cloud music player

Olivia

Olivia looks like a standard three-panel music player, with links to albums, artists, and playlists on the left, the player queue on the right, and a context-shifting middle pane. But it's not. Rather than helping you manage and maintain your own music collection, Olivia has been designed to simplify access to music that's typically played and discovered online. It's currently in an alpha testing state, and not all the features shown in the user interface (UI)

are functional, but it's functional enough to be very useful and shows great promise. Type the name of a track into the search field, for example, and a list of image thumbnails for discovered tracks start to load into the middle pane, complete with details about the performer, release date, duration, and album. It's exactly as if the music is sourced from your local storage. A double-click adds the track to your queue from where it can then be played. The actual source for the music seems to be YouTube, from where the music is streamed stripped of its video content.

The UI scales and animates smoothly as you navigate through dif-

ferent search and playback modes, and it can even dynamically theme itself according to your currently playing track's artwork. There's a very neat "widget" mode, which reduces the UI to nothing more than the current track thumbnail and playback controls. This is a great way of removing the distraction of choosing music from the infinite possibilities of online resources. As you play tracks, they're added to your "collection," so you can easily play them back or manage them much like you would local files. Local music is supported too, and there's an excellent song recommendation system. Type in the name of a piece of music you like, and Olivia will come back with a recommendation for something it thinks (or the Internet thinks) is similar. It works surprisingly well.

Another aspect of Olivia we didn't expect to be so good is the Internet radio facility. This is a natural extension to the music search, and there seems to be an immense library of streaming stations to choose among, with the documentation claiming support for over 25,000 stations. It was certainly able to play whatever esoteric French jazz stations we searched for, and you can also list stations by location and language if you don't have a specific name to mind. As with the YouTube music searches, there's always a delay between your selections, when those tracks are added to the queue, and when playback starts, but this is likely to be due to caching. Big chunks of the middle pane seem to be rendered in HTML, making it sometimes feel like a browser. But the output is perfect, and you completely forget you're tapping into online music resources. This gives Olivia a real advantage over using a browser to do the same thing, because a browser (and YouTube itself) has so many other distracting elements that you never really stop and just listen to the music.



1 Search: Find and play music from an online source. **2 Queue:** Search YouTube directly, or via Olivia's curated metadata. **3 Tasks:** As there's usually a delay before playback, see what the engine is doing. **4 Song cache:** Music is cached after playback, so it doesn't need to be streamed again. **5 Recommendations:** Type in a track name and find similar pieces of music. **6 Results:** See variations of a piece of music that only online sources can return. **7 Internet radio:** Search and play thousands of online streaming stations. **8 Desktop widget:** Minimize the application into a simple accessible playback widget.

Project Website

<https://github.com/keshavbhatt/olivia>

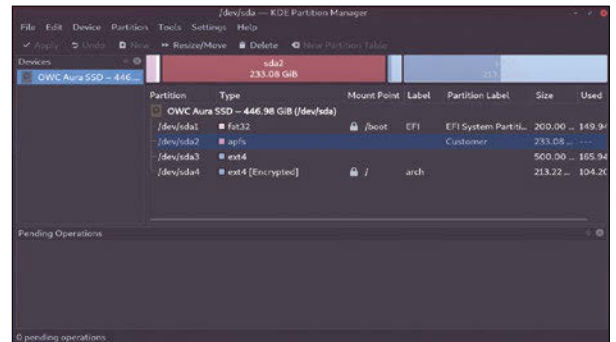
Partition Management

KDE Partition Manager 4

Full disk encryption (or SSD encryption) has become an essential component in many of our Linux installations and for a vital reason. With so much of our lives now conducted from our computers, we can no longer afford to risk the data on our portable devices falling into the wrong hands. Encryption helps mitigate this risk by protecting your data should the worst happen, and you lose your laptop or it gets stolen. But it has the disadvantage of adding another layer of complexity to filesystems and partitions and to the assumed knowledge users already require to run the systems. This puts people off encrypting their drives and, ultimately, puts them at greater risk. Which is why we need great par-

tioning and formatting tools, such as GParted and this excellent new release of KDE's Partition Manager.

KDE Partition Manager v4 is the result of 18 months of development. Apart from the work done to isolate authentication from the app itself, which is something that helps KDE run on Wayland, a big emphasis has been placed on working with encrypted partitions. In particular, there's now better support for LUKS2, and most encrypted partitions can now be resized much like any other partition. Similarly, if you dual boot your Linux machine, Partition Manager can now navigate its processes around both Apple's APFS encryption container and Microsoft's BitLocker. This is important because, especially with



GParted is a brilliant, dependable tool, but a little competition is a good thing – KDE's Partition Manager is getting close.

APFS, partition tables get lost among these new container paradigms, sometimes even blocking the update of macOS without a complete reformat. It then becomes essential to use a partition manager that understands what these partitions are and what they contain, even if they can't manipulate them. And that's exactly what this release does, helping more people encrypt their drives and confidently work with their data, even on Apple or Microsoft hardware.

Project Website

<https://invent.kde.org/kde/partitionmanager>

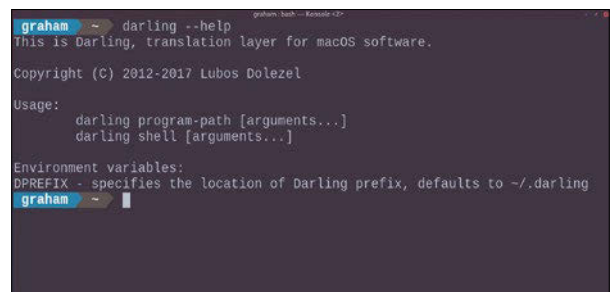
macOS compatibility

Darling

It's understandable why we've long needed Windows compatibility on Linux. Windows was always the most popular desktop operating system and ran many of the proprietary applications desperately needed by Linux users. It also helped that Windows mostly used the same hardware architecture as Linux, and all this commonality led to Wine, which lets you run Windows executables. It had humble beginnings, starting off with a few utilities, applications, games, and developer tools, but it currently helps to run Proton and a huge Steam library of Windows games, complete with DirectX to Vulkan conversion and hardware acceleration.

But Macs, and macOS, have never enjoyed the same atten-

tion. There was little reason when Macs were built on PowerPC CPUs. But that time is long gone, and modern Macs are virtually indistinguishable from their PC cousins inside, even with the many modifications and standards variations that Apple likes to make, as many Hackintosh users have discovered. As well as being a portmanteau for Darwin (the open source core of macOS) and Linux, Darling is a project that feels very much like those early Wine prototypes. Like Wine, it's a translation layer to help bring macOS applications and tools to our favorite operating system. The project has been on and off over the last couple of years and takes some considerable compiling effort to get working, mainly because you're build-



Save yourself hours of compile time by passing Darling the CMake build parameter.

ing kernel modules, but Darling still lets you run simple macOS binaries. You can open a simple shell, for example, and use the `pkg` command line to install simple commands. There's also a build of Midnight Commander that will work. All of this is of course a long way from being able to run Final Cut Pro or Ableton Live, but it's still important work that validates Apple's open source releases and builds a foundation for macOS compatibility for future Linux desktops.

Project Website

<https://wiki.darlinghq.org>

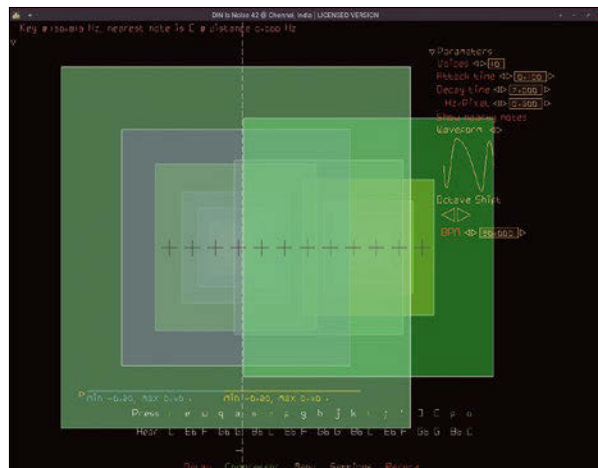
Performance synthesizer

DIN Is Noise

DIN is a sound generator you can play from your computer keyboard, as well as your MIDI keyboard. But it's not a professional re-creation of an analog classic, or a modern reinterpretation of classic subtractive synthesis. Instead, DIN feels more like a cross between an atonal experimental performance tool and the classic game Rez on Sony's PlayStation 2. This is because it's immediately playable with an interactive, vectorized graphical style that illustrates the pitch and volume of each note you play. Hit a few keys on your keyboard, and a large square for each tone appears and immediately starts to shrink, unless you hold the key. You can then control the pitch of whichever notes are playing with

the mouse cursor. This gives you considerable control over the pitch as a note decays, much like you might get by waving your hands over a theremin. It doesn't always sound pleasing, but it's always interesting.

The sound engine itself is also equally rich and esoteric with a set of unique features. The basic sound contains a waveform you can change, with some unusual features such as Bézier curves and control points for editing even the sine waveform. These are the sounds you hear when you launch the application and start hitting the keyboard. But there's more to this than playing simple notes. There's a micro-tonal mode, for instance, which generates pitch oscillators that follow a modulation path across



The sounds emanating from DIN are as unique as the visuals that appear and animate as you play.

a grid, creating dense and often ominous clouds of closely pitched sound – great for soundtrack work. Similarly, there's also a mode for generating binaural drones. These are often used in relaxation videos because of the way pitch changes in the left and

right channels subtly phase against each other. This is simpler in harmonic content than the micro-tonal mode and difficult to generate with an ordinary synthesizer.

Project Website
<https://dinisnoise.org>

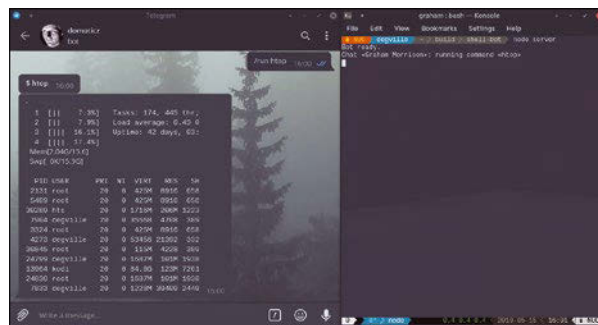
Remote access

shell-bot

The Telegram messenger client has become hugely popular, partly thanks to its open source client and partly thanks to it not (currently) being owned by Facebook. For the security conscious, it's still a risky proposition without independent scrutiny of its server and encryption code, but it's undoubtedly useful. And one of the things that makes it more useful than the average messenger client is its "bot platform." This allows anyone patient enough to step through the lengthy and convoluted API key-generation process to run their own automated Telegram response system, feeding the contents of your own chats with whatever source you can push to the API.

Which is exactly what shell-bot does. With Telegram API access

enabled and your own bot channel configured, you can install shell-bot on your Linux machine or server. This is relatively straightforward after you've worked through the few `npm` dependencies; the script configured itself when first executed. After that, it's as simple as leaving it running on the machine you wish to access. Shell-bot can then receive commands sent from your bot's Telegram channel after typing `/run` followed by the command. Command output is returned to the Telegram channel, and it updates "in-place," which means the output from a command like `htop` will update without scrolling. It's perfect for monitoring your system from wherever you can access a Telegram client. Type `/help` in the channel to see



Execute commands sent from a Telegram channel and see the results, without making any manual connection to a remote server.

which other commands you can use. These include `resize` to change the output width, `upload` to download a file to your Telegram client from your Linux machine, `ctrl` and `meta` for sending control and ALT shortcuts, and `cancel` to interrupt a running process. It works brilliantly and makes a great alternative to remembering a server's IP address and worrying whether you have your SSH keys or passphrase on you for remote administration.

Project Website
<https://github.com/botgram/shell-bot>

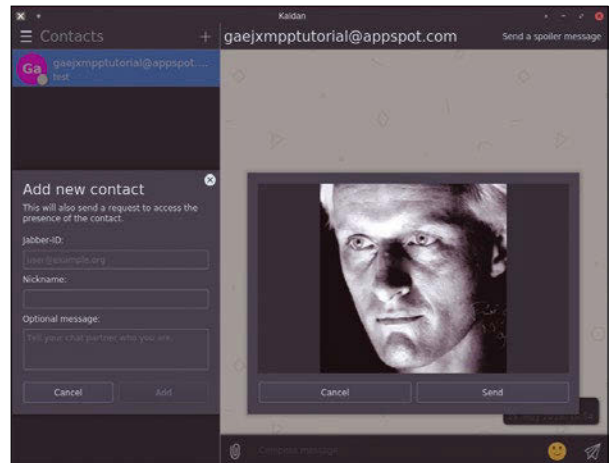
Chat client

Kaidan

You could almost be forgiven for thinking the age of the open messaging client is over. WhatsApp and Telegram and countless other closed services seem to have taken over, and without some clever bridging, none of them can talk to each other. Matrix and Riot.im are promising modern open alternatives, but they don't have the critical mass to connect with people outside of your average open source community. However, even after being dumped by Google, some of the old protocols manage to survive. And Jabber and XMPP are perhaps the most prominent. All you really need is a modern, low resources, graphical client that connects to the Jabber/XMPP network and a few friends that

feel the same way (or perhaps some of the many bots you can run). And that's exactly what this new KDE app, Kaidan, does.

With a name that's going to confuse lots of people with the video editor Kdenlive, Kaidan is a simple to use, quick, and functional Jabber client. You simply sign in, add a few contacts, and you're ready to go. Messages and your replies appear on the right while your contacts are listed on the left. The UI design is clean and minimal and looks very sleek thanks to the Qt components used to build the application. On a KDE desktop, it very much feels like a modern messaging client like Telegram or even Slack. The UI is adaptive, for example, which means it becomes just the conversation view when the window width is re-



You can still create free and relatively anonymous Jabber accounts on sites like <https://jabber.hot-chilli.net>.

duced, and you can easily see images you upload or within the conversation stream. This positions Kaidan brilliantly on both the desktop and any potential mobile platform that Plasma hopefully finds itself on in the future.

Project Website

<https://invent.kde.org/kde/kaidan>

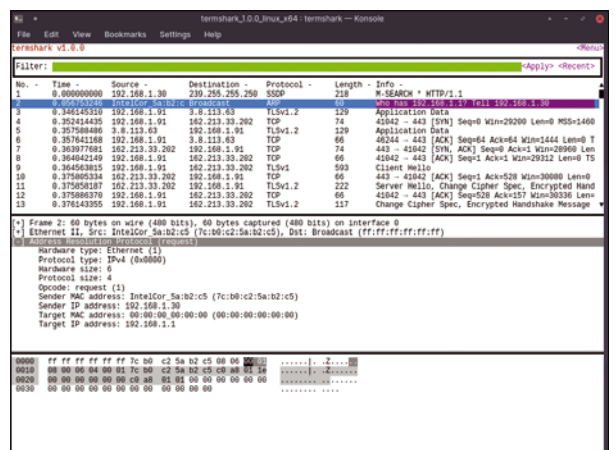
Terminal network tool

termshark

Wireshark is an amazing tool for analyzing and filtering network packets. It's difficult to use, but it allows you to explore the inner depths of protocols and even what goes on beneath protocols. It has become essential for anyone who wants to make sure there's nothing untoward happening on their network or to discover whether that cheap Telnet-enabled webcam is sending packets to your server or live streaming your garden to China. But Wireshark needs a GUI, which means you're somewhat restricted in where you can run the analysis, because the device at least needs to be capable of producing a graphical interface. Wireshark does include `tshark` to help with this, but it's not as intuitive or as discoverable as the GUI version

since it's built for network protocol analysis. What we really need is termshark!

Termshark harnesses and contains the functionality provided by `tshark` and Wireshark, only from the command line. You start it by taking a packet capture of the network interface as arguments, with optional protocol filters. With that done, you see the main view. Remarkably, if you can make your way around the GUI version, you'll be able to find your way around this terminal version, because the layout and interactive elements are very similar. You lose small refinements like the device activity thumbnails, but you gain the ability to capture and filter, export and import captures, and expand packet elements just as you would with the desktop version. It even works



Feel like a real hacker by diving into the depths of your network from the command line.

well with a mouse, and the UI remains quick and responsive through all your captures, expands elements, and views specific sets according to your requirements. There are a few bugs, such as the drop-down filter menu length, but it's still the same old Wireshark, only wonderfully running from the command line.

Project Website

<https://github.com/gcla/termshark>

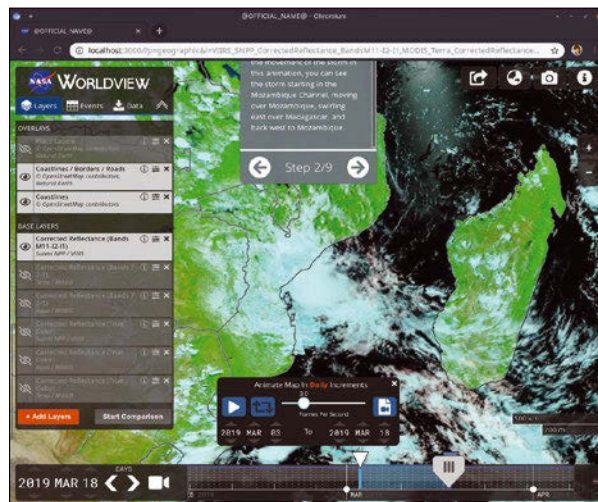
Satellite imagery

Worldview

There are many amazing things about NASA. However, one thing that we don't often consider is that, alongside landing humans on the moon, pioneering reusable orbital spacecraft, and using the Hubble space telescope to calculate that the Milky Way Galaxy weighs approximately 1.5 trillion solar units and has a radius of 129,000 light years, NASA also releases open source software. One of which is Worldview, a portal on its historical Earth imagery. Worldview isn't a standalone application in the normal sense, but it's installed locally via `npm` and accessed from a web browser pointed at `http://localhost:3000`.

The landing page lets you explore specific topical events, such as fires over California or a

tropical cyclone, and selecting one of these is much like selecting a location in Google Earth – the main view map scrolls across a globe and zooms into the area in question. You can then step through some explanatory text for an event, choose to show different imagery layers, such as OpenStreetMap labels, and scroll through an image timeline. Stepping through the explanatory text will often animate the event's imagery, or you can manually control this for any location with an animation tool that lets you define how many frames per second you'd like for a time window. Overlays are the image stacks you can view for a location, and these include rain rates, nighttime imagery, and base layers to show a location's



Worldview can be installed locally, but you can also access it online via a web portal.

color images. As with Google Earth or KDE's Marble, it's an amazing way of exploring the Earth from your desktop, only Worldview is a portal into the Earth's volatility and fragility. It's both incredibly informative and a genuine tool for study and research. Thanks to NASA, anyone can install it on their machines.

Project Website
<https://github.com/nasa-gibs/worldview>

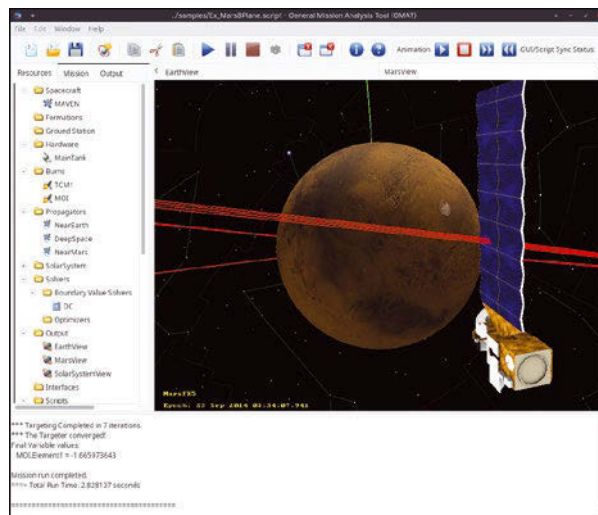
NASA explorer

General Mission Analysis Tool

Another piece of NASA software you can run on your desktop is General Mission Analysis Tool (GMAT). This is a complex desktop application that's been designed to model and analyze specific mission data, such as trajectories for different types of objects in different flight regimes. It's certainly not for the faint of heart, but it's also a fascinating window into the world of space flight. If you play Kerbal Space Program, GMAT is the upgrade tool you should be using when you get a job at NASA and start designing your spacecraft for real and launching it into space. It's complicated and fascinating in equal measure.

GMAT is basically an IDE for the various elements that come

together to map out the feasibility of a mission into space from your desktop. Fortunately, it includes a tutorial mode and several example scripts. These are a great way of getting used to the environment and learning about the various elements required to build a model. One of these example scripts, MAVEN, takes a spacecraft to Mars to study its atmosphere from orbit. With the example loaded, you can play with its three propagators – near Earth, deep space, and near Mars; the transfer scripts; the values for the ship and its engine; and its various burns. These are all elements in a typical mission, and they all interact with each other in much the same way different source files, images, and



If you've ever wanted to go further than Kerbal Space Program, GMAT lets you explore and create real space missions.

libraries connect in a coding project. As with an IDE for programming, clicking on `Run` (or pressing `F5`), builds the mission trajectories and shows them on the output views, which for the MAVEN example is a 3D model of Earth and

Mars showing the trajectory of the spacecraft. You can then scroll and zoom around to explore where the craft goes.

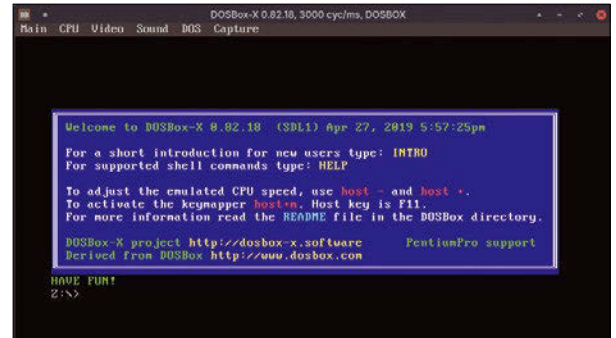
Project Website
<https://sourceforge.net/projects/gmat/>

DOS emulator

DOSBox-X

You've probably already played with the original DOSBox, which is a brilliant piece of software. It enables you to run old PC DOS software on modern PC hardware, solving two problems in a single stroke. The DOS environment that preceded Windows is no longer a thing. Even if you were able to get it to work on modern hardware, it would run far too fast for most software and games to be usable. DOSBox brings compatibility and allows you to throttle the speed, all without remembering a single IRQ register. But for some developers, DOSBox's emulation options are still too limiting, especially if you want emulation at the same exacting standard as some of MAME's emulated hardware reaches. And that's what this fork of DOSBox does.

The DOSBox-X project intends to present many more configuration options to the user, to target a much more specific emulating demographic – those who want to play old games and demos as accurately as possible. DOSBox-X has also set itself a hugely ambitious target of emulating all pre-2000 DOS and Windows 9x-based hardware, including peripherals, motherboards, and CPUs. You can see the results of some of the early work in the way DOSBox-X runs old demoscene demos more accurately than Bochs, QEMU, and DOSBox, passing many tests that the original DOSBox fails. This is obviously where the inspiration for the fork comes from, and building a platform to accurately recreate the performance of this old hardware is just as valuable as the



DOSBox-X can also emulate old versions of Windows alongside ancient versions of DOS.

work done to recreate the classic SID chip or the open source re-engineered Commodore 64 ROMs. Without these exacting recreations, we'll likely lose an important part of early technology history after the hardware has failed. But projects like DOSBox-X allow this vital piece of history to be snapshotted and played with by anyone with a Linux box.

Project Website
<http://dosbox-x.com/>

Classic shooter

Snipes

Sticking with the theme of ancient DOS games (see above), Snipes is likely to have been the first computer game many children of the 80s played. It's a classic from 1982 that, despite using text mode and beep sound effects, remains very playable. You can't even tell it's running in text mode, as the simple line graphics and animations feel very similar to games like PAC-MAN. However, the gameplay itself feels more like later arcade shooters like Smash TV, because there are two sets of controls. You use the cursor keys for left, right, up, and down movement, and A, S, D, and F for shooting direction. The game's objective is to simply move around the maze and shoot the 'snipes' and their hives. This

means you can move in one direction while shooting in another, which is quite an advanced control mechanism that adds a lot of depth and timing requirements to the gameplay.

This version prides itself on being 100 percent identical to the original, despite being completely reverse engineered from the original. Sadly, one revolutionary feature of the original hasn't made it into the 21st century. Remarkably, the original had networked play, at least in a later release, which meant that if you were on a network with other players on the same network drive, you could hunt them down in the maze instead of the regular snipes. It's a long way from a modern First Person Shooter game, but it was still crazily ad-



This version of Snipes was reverse engineered from the original to run on modern hardware.

dictive. It would be fantastic if something like networked play could be added in the future, but it's already a very nostalgic and utterly authentic way to waste a few minutes, especially if you played the original.

Project Website
<https://github.com/Davidebyzero/Snipes>

Making your scripts interactive

Bash Communication

Letting your scripts ask complex questions and give user feedback makes them more effective.

BY MARCO FIORETTI

The final installment in my shell script tutorial series focuses on *conversations* between Bash scripts and their human users who interact with them during script execution. Part of what you will learn here can be applied to conversations between shell scripts and other programs, but this is a different problem, which is often best solved with other tools, such as Expect [1].

As with every other programming feature, the most important questions are not “what” or “how,” but “when” and “why” a script would need conversations. For the purpose of this tutorial, I will distinguish three broad cases, some of which may overlap in several ways.

In the first case, there are certain scripts that should not hold conversations at all (reporting outcomes and errors in logfiles is a different issue, of course). Obvious examples are scripts that should run as cron jobs or that must do exactly the same thing every time they run.

The second case involves scripts that must ask *questions* (i.e., receive different arguments every time they start) but require no other input after that.

The third case involves actual conversations, which occur in scripts that regardless of initial arguments must also

carry on some more or less flexible dialog with users or programs over their entire execution.

Before I start, there is one common rule for all three cases: Never trust user input – even if that user is *you!* Due to space constraints, I have omitted any validation code from my examples. Do not, as they say on TV, try this at home! If you ask a user to enter a number in your script, include validation code to verify that it is a number;

if it isn't a number, either ask again for a correct value or abort the script. To do this, use the test operators explained in the fourth installment of this series [2].

Command-Line Arguments

The simplest way to pass a script some data is on the command line when you launch the script. As previously explained in this series, all the arguments passed to a script as follows

```
#> scientists-directory.sh Ada Lovelace
mathematician 1815
```

are available inside the script in the special variables **\$1**, **\$2**, and so on or in one array called **\$@**. In the above example, the `scientists-directory.sh` script should begin by copying all those values into other variables with better, self-documenting names

```
NAME=$1
SURNAME=$2
PROFESSION=$3
YEAR_OF_BIRTH=$4
```

and then only use those other variables as needed. One limit of this approach is that it is easier to make mistakes, for example, by passing arguments in the wrong order. You can greatly mitigate this risk by making your script require command-line switches, which are named parameters that may or may not have an assigned value. In both cases, to distinguish the parameter names from their values, the former always starts with one or two dashes. An argument like `-v` may mean “enable verbose logging,” and a couple of strings like `--newuser marco` would tell the script to assign the value `marco` to the `$NEWUSER` variable.

Listing 1, which is taken from a backup script capable of both total and incremental backups, shows the simplest way to parse such arguments. If the first command-line argument passed to the script (**\$1**) has a value of `-t`, `--total`, `-i`, or `--incremental`, the `case` statement sets the `$BACKUP_TYPE` accordingly, so the rest of the script knows which backup procedure it should execute. If there is no first argument, the script exits. There could, of course, be a default `$BACKUP_TYPE` to run when no arguments are given. Whether that is a good idea or not depends on your individual needs.

Listing 1: Parsing Command-Line Switches

```
key="$1"
case $key in
  -t|--total)
    BACKUP_TYPE="total"
    ;;
  -i|--incremental)
    BACKUP_TYPE="incremental"
    ;;
  *) # missing option!!!
    echo "you forgot to specify the backup
type, quitting!"
    exit
    ;;
esac
# actual backup code would start here...
```

The method shown in Listing 1 can be extended to handle multiple command-line arguments (see Listing 2), each of which may have *any* value and appear in any order.

The special variable `##` in line 1 holds the number of parameters passed on the command line that are still available in the `$@` array. The `while` loop in the same line continues until `##` is equal to zero (i.e., until all arguments have been parsed and removed). The commands in lines 6 or 11 are executed every time `$1` (the *current* first argument saved in `$key`) matches one of the accepted switches (`-f`, `-u`, and so on). When that happens, the *following* parameter (`$2`) is copied to the internal variable corresponding to that switch. If the code in Listing 2 were in a script that makes on-demand backups of only some file types (e.g., images, audio, etc.) for one user, launching it as follows

```
#> custom_backup.sh -u marco -f images
```

would set `##` to 4, and the variables from `$1` to `$4` to `-u`, `marco`, `-f`, and `images` respectively. Consequently, the first execution of the case statement in line 5 would set `$USER` to `marco` and then “shift away” (i.e., remove) the first two elements of `$@` (lines 8 and 9). This would bring `##` to 2, causing one more iteration of the `while` loop. This time, however, `$key` would be equal to `-f`, because the original first two elements of `$@` (`-u` and `marco`) were removed. This would set `$FILETYPE` to `images`. It is easy to see that changing the order of the pairs of arguments

```
#> custom_backup.sh -f images -u marco
```

would yield exactly the same result. Any argument in `$@` after the ones explicitly mentioned in the case statement (or before the first pair) would be dumped into the `$OTHERS` variable.

A more sophisticated option to parse command-line arguments is the `getopts` built-in command, whose main features are shown in Listing 3. The real difference between this `while` loop and the one in Listing 2 are `getopts`' arguments (line 1). If there are only two arguments (as in this example), `getopts` works on the actual command-line switches passed to the script. Optionally, you may pass any other array, as a third argument, to `getopts`.

The second argument in line 1 (`opt`) is the variable that must store the current option. The initial, admittedly cryptic string `":ht:"` contains the recognized (single-character) switches (`h` and `t` in the example) with some qualifiers. Starting this string with a colon tells `getopts` to set `$opt` to `?` whenever a command-line switch other than `-h` or `-t` is found. Of course, this will make a difference to the script only if it contains code to handle such errors (line 11).

As far as the actual command-line switches are concerned, please note the important difference in the processing of `-h` and `-t`. The first switch only works, when present, as an actual on/off switch of some function. The colon after `t` in line 1, instead, means that this option must have a value, which `getopts` will automatically copy in the special `$OPTARG` variable. When that value after `-t` is missing, `$opt` is set to a colon, causing the error message in line 9.

Line 15 has the same purpose as the `shift` commands in Listing 2, because the special variable `$OPTIND` holds the number of options parsed by the last call to `getopts`: Therefore, decreasing it as shown here removes the last argument that was already parsed from `$@`. A short, but clear and complete description of how `getopts` works is available online [3].

Asking Multiple Questions

Command-line arguments are very flexible, but, by definition, you can only use them once and only before the script starts running. When that is not enough, the easiest way to get user input that runs entirely in the terminal without any third-party program is the `read` built-in command discussed in the second, third and sixth installments of this series [4] (second was “Shell arrays”, third was “Shell Flow control” and sixth was “Shell functions”). You can call `read` as many times as you like in a script, and each invocation can set many variables, with several options controlling how the command behaves.

Inside the already mentioned `scien-`
`tists-directory.sh`

Listing 2: Multiple Command-Line Arguments

```
01 while [[ $# -gt 0 ]]
02 do
03 key="$1"
04
05 case $key in
06 -u|--user)
07 USER="$2"
08 shift
09 shift
10 ;;
11 -f|--filetype)
12 FILETYPE="$2"
13 shift
14 shift
15 ;;
16 # many other options here...
17 *)# unknown option
18 OTHERS+=("$1"
19 shift
20 ;;
21 esac
22 done
```

Listing 3: getopts

```
01 while getopts ":ht:" opt; do
02 case ${opt} in
03 h ) # enable option h
04 ;;
05 t )
06 target=$OPTARG
07 ;;
08 : )
09 echo "ERROR! $OPTARG requires an
10 argument!"
11 \? ) echo "ERROR! Usage: cmd [-h] [-t
12 <TARGET>]"
13 ;;
14 esac
15 done
16 shift $((OPTIND -1))
```

script, for example, a single `read` call may load all the data about one scientist:

```
read NAME SURNAME PROFESSION YEAR_OF_BIRTH
```

What happens here is that `read` loads an entire line of input and splits it into words according to the value of the `$IFS` variable [5]. Next, `read` saves each of those words inside the variable in the same position in the argument list it received (`$NAME`, `$SURNAME`, etc.). If there are less words than variables, the extra variables remain empty. In the opposite case, all the excess words are appended to the last variable (`$YEAR_OF_BIRTH` in the command above). If you call `read` without any argument, instead, the entire line is

saved into the special variable `$REPLY`. Alternatively, you may call `read` with the `-a ARRAYNAME` option, and it will save all the words it reads into one array called `$ARRAYNAME`.

Beware! The same power that makes this command very handy can cause problems if you are not careful. To make this point, Listing 4 deliberately uses bad code, which collects voter data inside a generic voter registration script.

Let's see how this works with a few different inputs before explaining the code in detail. If the user is a child, the following appears in the terminal window:

```
Please type your year of birth (4 digits), ↵
followed by [ENTER]:
2010
sorry, you cannot vote because you are only ↵
9 years old. Quitting...
```

So far, so good. But what happens with bogus, unchecked data?

```
Please type your year of birth (4 digits), ↵
followed by [ENTER]:
0
You may vote, because you are 2019 years old.
What is your nationality? [ENTER]:
...
```

Not so good now, right? Even worse is the fact that you would obtain exactly the same result by entering something like `NONE` as your year of birth. A quick and dirty explanation of why this happens is that non-numeric values are interpreted as null (in practice, zero) when the variable that holds them is used in arithmetic expressions. But wait! I saved the best for last. This is what happens with another set of inputs, assuming that the user's nationality is Italian:

```
Please type your year of birth (4 digits), ↵
followed by [ENTER]:
1987
You may vote, because you are 32 years old.
What is your nationality? [ENTER]: i
Please type "t" if you are a terrorist, ↵
followed by [ENTER]:t

you are a citizen of i, 32 years old but you are ↵
a TERRORIST! Please freeze, the police are coming ↵
for you!

Please enter any comment you may have here, ↵
followed by [ENTER]: Rats! How did you know???
```

Rats! How did you know???

#>

Listing 4: Misusing read

```
01 #! /bin/bash
02
03 declare -i year
04
05 echo "Please type your year of birth (4 digits), followed by
[ENTER]:"
06
07 read year
08 age=$(( "2019" - "$year" ))
09
10 if (( "$age" < 18 ))
11 then
12     echo "sorry, you cannot vote because you are only $age years old"
13 else
14     echo "You may vote, because you are $age years old."
15 fi
16
17 echo -n "What is your nationality? [ENTER]: "
18 read -n 1 nationality
19 echo
20
21 echo -n "Please type \"t\" if you are a terrorist, followed by [ENTER]:"
22 read -n 1 goodguy
23 echo
24 echo -n "you are a citizen of $nationality, $age years old "
25
26 if [[ $goodguy == "t" ]]
27 then
28     printf "but you are a TERRORIST! Please freeze, the police are coming
for you!\n\n"
29
30 else
31     printf "and you are not a terrorist. Congratulations, you may vote!\n\n"
32 fi
33
34 read -p "Please enter any comment you may have here, followed by
[ENTER]: " feedback
35 printf "\n\n$feedback\n\n"
36
37 # Actual registration of the voter in a database would start here
```



Figure 1: Who says that scripts are limited to command-line terminals? Use Zenity to make scripts create graphical forms.

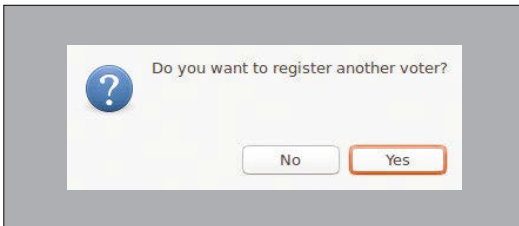


Figure 2: Zenity can create all the common graphical widgets that you may need in a shell script.

When you call `read` with the `-p` option, it prints the corresponding prompt (Listing 4, line 34). Instead, when you use `-n <NUMBER>`, it only grabs the first `n` input characters. Therefore, typing *italian* as the answer to the question on line 17 made *only* the first character (*i*) go into `$nationality` (line 18), and only the second character (*t*) go into `$goodguy` (line 22), thus labeling me as a terrorist. Without input validation between lines 18 and 22, Bash took everything I typed and merrily distributed it among the several commands requesting user input according to those commands' options. Consult `read`'s man page for other options. Consequently, using `read` or any other tool in a Bash script makes it easy to ask questions in order to update many variables quickly, but you need to ask unambiguous questions and *never* use the answers without checking that they at least *look* correct.

Requesting Input in a GUI

Commands like `read`, `echo`, and `printf` are the simplest way to ask questions and receive answers inside a shell script, not to mention the only ones that will work with console connections to remote Unix computers where only the very basic tools are installed. However, you don't have to be limited by these commands. Figures 1 and 2, which were generated with the code in Listing 5, show that you can create pop-up windows that perform the same functions with the Zenity tool [6].

In addition to Figures 1 and 2, Zenity can produce many more window types. Comparing Listing 5 with Figures 1 and 2 also shows how Zenity is simple to use once you have had a quick look at its documentation [6]. Of note in Listing 5, line 9 shows how Zenity makes the answers received by the user available to the script that calls it. The Yes/No

buttons in Figure 2 just produce an exit code, but when the user types something, the data is sent to standard output, which you can redirect to a file like `voter-data.csv`. Inside that file, fields are separated by a pipe (`|`), so the content of the file generated by Figure 1 would be `1981|Italian`.

Talking to Users

Here documents (heredocs) are one way to make a shell script talk to users without a third-party program. They look like this:

```
cat << EMAILADDRESSES
$MANAGER
jane@linux.com
tom@linux.com
EMAILADDRESSES
```

As you can see, a heredoc is just a series of lines of plain text, delimited by two occurrences of the same string (`EMAILADDRESSES` in the above example). The closing occurrence of that string must be the *only* text on its line. If the heredoc body contains calls of variables, like `$MANAGER`, those calls are replaced by the current values of the same variables. In the format above, the heredoc content is just printed to the standard output with the `cat` command. However, it may also be saved into a variable for later use:

Listing 5: Zenity Dialogs

```
01 #! /bin/bash
02
03 cd $HOME
04
05 zenity --forms --title="Voter Registration form" \
06 --text="All fields are mandatory" \
07 --add-entry="Year of Birth (4 digits)" \
08 --add-entry="Nationality" \
09 > voter-data.csv
10
11 case $? in
12 0)
13     echo "Data loaded, thanks"
14     ;;
15 1)
16     echo "Data not loaded"
17     exit
18     ;;
19 -1)
20     echo "Zenity error"
21     exit
22     ;;
23 esac
24
25 zenity --question --text "Do you want to register another voter?"
26 ANOTHER_VOTER=$?
```

```
MYADDRESSBOOK=$(cat <<'EMAILADDRESSES'
$MANAGER
jane@linux.com
tom@linux.com
EMAILADDRESSES
)
```

Heredocs are a very convenient system to define and embed generic templates inside a script. Many heredocs are created to dump data into some file or database. In the context of this article, however, their main use is to interact with others. The reason is that heredoc templates may be completely static, but they also can be generated on the fly (and often are) for many different purposes.

A heredoc's content may be shown to the user to explain what the script is doing or to summarize all the provided data and ask for confirmation before using the data.

Heredocs can also allow a script to send multiple input commands to the standard input of programs that should otherwise be executed manually, typing text one line at a time. In that case, however, you should really be careful. Conversations are well and good, but only if both parties can talk and listen at the same speed! You really don't want to throw two or more commands in one shot at a program unless you are *absolutely* sure that this will not cause any of those com-

mands to be missed or to fail, because the program depends on the complete, successful execution of all previous commands!

Graphic Feedback

Imagine a process that takes some time (possibly a long time) to complete. It is good practice to let the user know how far that task is from completion; you can do this with a simple text-based graphic or animation. The simplest solution, which works in any shell without installing anything else, is a progress bar drawn with `echo` commands:

```
echo -ne '### (33%)\r'
sleep 1
echo -ne '#### (66%)\r'
sleep 1
echo -ne '##### (100%)\r'
echo -ne '\n'
```

Each command prints a number of hash characters that is proportional to the percentage of the task already completed. Then the carriage return (`\r`) at the end of each string places the cursor back at the beginning of the line. This makes the next `echo` overwrite the previous one. Note that the `sleep` instructions in the above example are just placeholders for the real code that you want to run between each call. This method's main limitation is that it only works if you know what percentage of the total running time each phase takes.

In general, to draw more precise progress bars, in or outside a terminal, you must know the entire task's size. This may be expressed with sufficient accuracy by metrics like the total number of files to compress, text lines to parse, or digital photographs to index. Once you have that number, you can use tools like `pv` or `dialog`, which are available in the standard repositories of all major Linux distributions, to draw progress bars.

`pv` must be placed in the middle of a pipeline to observe the flow of data being processed. Then, it can print to the terminal both an ASCII progress bar and information such as the elapsed time, the percentage of work already completed, and other data. Figure 3 shows the result of measuring the time taken to make a tar archive of a entire directory of size `$DIRSIZE`:

```
DIRSIZE=`du -sb . | awk '{print $1}'`; 2
tar c . | pv -s $DIRSIZE > 2
~/all-my-articles.tar
```

You can draw precise progress bars (or more accurately gauges) with the `dialog` tool, which is the console, text-only version of Zenity. It is the same tool that is used in the text-based installation procedures of many Linux distributions. In addition to drawing progress bars, it can also col-



Figure 3: The `pv` program creates the most complete progress bars you can display in a Linux console.

Listing 6: Two Types of dialog Progress Gauges

```
01 #! /bin/bash
02
03 files=$(HOME/tarfiles/*.tar)
04 dialog --gauge "Compressing tar archives... (Process Substitution
    Gauge)" 20 75 < <(
05 n=${#files[@]} i=0
06 for f in "${files[@]}"; do
07     sleep 3
08     bzip2 $f
09     echo $((100*(++i)/n))
10 done
11 )
12 bunzip2 $HOME/tarfiles/*.tar.bz2 ; files=$(HOME/tarfiles/*.tar)
13 n=${#files[@]} i=0
14 for f in "${files[@]}"; do
15     sleep 3
16     echo $((100*(++i)/n))
17 done | dialog --gauge "Compressing tar archives... (Pipe Gauge)" 10 75
18 exit
```


Listing 7: Rotating Spinner

```

01 #! /bin/bash
02
03 i=0
04 sp='/-\|'
05 n=${#sp}
06 printf ' '
07 sleep 0.1
08 while true; do
09   printf '\b%s' "${sp:i++%n:1}"
10   sleep 0.1
11 done

```

lect user input with text boxes, radio buttons, and other systems.

Using `dialog` to draw the gauge is relatively straightforward. Listing 6 shows two types of progress bars that can be created with `dialog`. Lines 4 and 17 show how to tell it what to draw (`--gauge`), along with the corresponding caption and window size (`20*75` or `10*75`). Even the code that actually compresses the archives and echoes (lines 9 and 16) the percentage of work already done is not complex, if you have read the fifth installment of this series “Shell Math” [7]. Listing 6 is important, because it shows two different ways to pass data to `dialog` (or any other command for that matter) from a whole bunch of shell commands. The first call (line 4) uses process substitution, a general Bash technique that allows you to bundle into one stream the output of *multiple* commands [8] (Figure 4). The second call to `dialog` (line 17) receives data from the `for` loop through a standard shell pipe (Figure 5).

Instead of progress bars, you can also use a rotating spinner (Listing 7), which I found online [9], to reassure users that the script is alive and running. Line 5 saves in `$n` the number of characters of the `$sp` string. The `while` loop starting in line 8 runs forever, because its condition is constantly true. Every time it runs, it prints one of the characters in the `$sp` string, then sleeps for 0.1 seconds. The character that is printed is the one in the `i` th

The Author

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

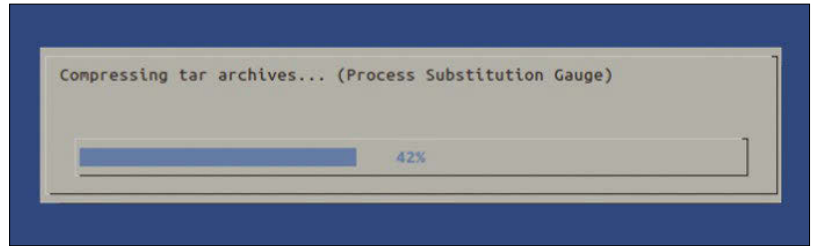


Figure 4: Depending on your preferences, you can use `dialog` to create a process substitution gauge ...

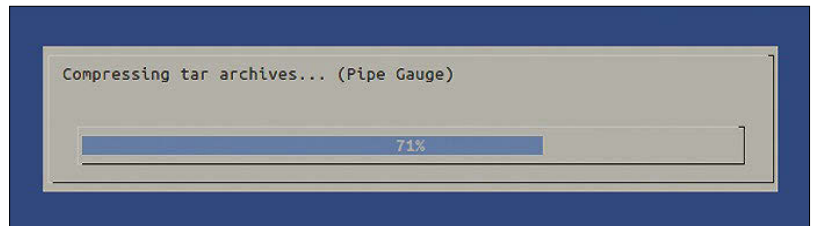


Figure 5: ... or a pipe gauge.

position in the `$sp` string. But in line 9, `$i` is incremented every time `printf` is used to select a character. This is why a different character of the `$sp` string is printed every time, giving the illusion of a rotating spinner.

Share Your Scripts!

Throughout this series, I have shown how the average Linux user can use Bash scripts to automate many different time-consuming tasks, in ways that may either be impossible or much more complex with other tools. Take advantage of what you have learned here. Above all else, please share with us your best shell tricks that you’ve discovered in your Linux scripting adventures! ■■■

Info

- [1] Expect: <http://core.tcl.tk/expect/index>
- [2] “Tutorials – Shell Test Conditions and Exit Codes” by Marco Fioretti, *Linux Magazine*, issue 222, May 2019, pp. 84-88
- [3] getopt: <https://sookocheff.com/post/bash/parsing-bash-script-arguments-with-shopts/>
- [4] “Tutorials – Bash Arrays” by Marco Fioretti, *Linux Magazine*, issue 220, March 2019, pp. 84-89
- [5] “Tutorial – Custom Shell Scripts” by Marco Fioretti, *Linux Magazine*, issue 219, February 2019, pp. 84-88
- [6] Zenity: <https://help.gnome.org/users/zenity/>
- [7] “Tutorials – Shell Math” by Marco Fioretti, *Linux Magazine*, issue 223, June 2019, pp. 84-89
- [8] Process substitution: <http://tldp.org/LDP/abs/html/process-sub.html>
- [9] “Can I do a spinner in Bash?": <http://mywiki.woledge.org/BashFAQ/034>

Preparing an object for 3D printing

Print Ready

Having covered several ways to design your 3D object, let's cover how we prepare your design for printing.

BY PAUL BROWN

We have been going over the process of designing an object, be it something decorative or a piece for a larger machine, but you have to know that you can't actually print an STL object directly on a 3D printer.

If you have been following this series, it is very likely you will have already bought, received, assembled, calibrated, and tested your 3D printer following the manufacturer's instructions. You may have even made some of your own designs using a computer assisted design program (like OpenSCAD [1] or FreeCAD [2]).

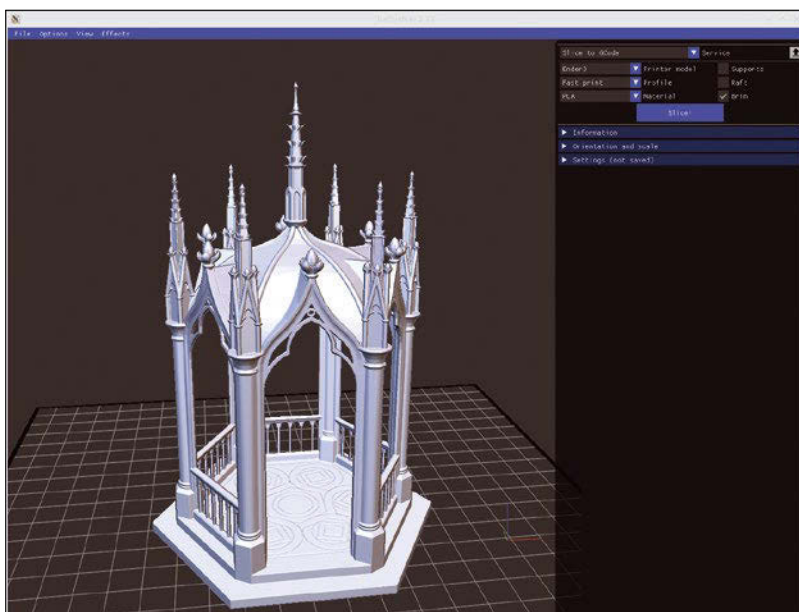
The next step is getting cozy with something called a *slicer* .

Slice & Dice

Before we get started on which slicer is best for you, there are three basic concepts that need explaining:

1 *Infill* is the cross-hatched pattern you often see inside 3D printed objects. Printing an object solid all the way through would waste a lot of filament and also make it very heavy. Printing a hollow object would make it as fragile as an egg. You can reach a happy medium using infill. You

Figure 1: This folly requires no supports thanks to the way its arches and roof are designed.



usually set the infill as a percentage in your slicer, zero percent being no infill (a hollow object) and 100 percent being solid fill. A tighter infill will make your objects stronger, but will also use up more filament and take longer to print.

2 *Supports* are like the scaffolding for overhanging things in your object. You use supports because ... well, 3D printing technology in general still hasn't reached the point where it can print on thin air. Slicers will generally insert supports by default for you. If you use the Cura slicer (see below), any face that shows up red in the preview of your object is getting a support.

There is a problem with supports, though: They are often difficult to remove. You often have to cut them out, and it can lead to breaking delicate parts of your print. They will often leave a mark where they were stuck on your print. There are two things you can do to minimize using supports: The first is not to worry too much about very small overhangs. I did say that 3D printers cannot print on thin air above ... except if the overhang is, say, three millimeters or less wide. Usually the filament's viscosity can help sustain narrow overhangs, although this will depend on the material you use.

The second thing is to learn from architects of yore. You may have noticed all those ogive (that's a real word) or pointed arches in medieval churches. They are so common, because they are structurally strong and *didn't need* scaffolding – or at least not much (Figure 1). The same applies to 3D printing objects: If your overhang is, say, less than 45 degrees from vertical, then you are probably safe. You may even get away with printing a semicircular arch without supports. Again, the viscosity of the material you use will be a factor. Experiment.

3 *Skirts, brims, and rafts*: No, we have not shifted into a sewing tutorial; these are all elements that help ensure your object sticks properly to the bed or helps you see whether the bed is properly leveled and the nozzle is extruding correctly.

A skirt is an outline of the printed object, which is *drawn* onto the bed with one layer of

filament (Figure 2). It is a reference for you, the user, and will tell you if there is any problem with the adhesion (it won't stick to the bed), leveling of the bed (some parts will be too pale or, again, some parts won't stick), or whether the extruder is having trouble (like if it is clogged). Pay special attention to the skirt, and cancel the print if you notice any problems.

Brim is a special kind of skirt that are joined to the object. A brim is designed to improve the adhesion of your object to the bed. Say you have an object with a very narrow base. You can make your slicer include a brim in your print, and your printer will create a wide, flat, one-layer surface in the shape of your print and print your object on that. When you finish printing, you cut the brim away.

A raft is similar to a brim, but instead of being solid, it is a lattice, and it is also taller. It is like a stand on which the printer will print your object. You may want to use a raft when printing with ABS, since a raft will often help avoid warping, a common problem with this material. As with skirts, rafts also help objects stick better to the bed.

Cura

A slicer does what it says on the box: It splits up your 3D object into layers (the *slices*), which your printer then prints one on top of (or under) the other.

There are quite a few slicers out there, and many are free software. We managed to narrow down the selection to three. The ones we are not going to mention we rejected because they were either proprietary, required some sort of registration, were nightmarish to configure, didn't work on Linux, or didn't work at all without requiring beyond a reasonable amount of tweaking. Let's not even bother with them. Life is short, and 3D print times are long.

Probably one of the most popular slicers out there is Cura [3]. Originally created by Ultimaker for their own line of printers, it has since developed further to support a much wider range of printers.

It is popular for a good reason: It is simple. The default interface is clean and straightforward, with big, friendly buttons and hints when you roll over the options (Figure 3).

In the upper left corner, the folder icon will open a file explorer that lets you search for the STL file you want to slice. You can load more than one object into Cura and arrange them on the virtual bed. In that manner, you can print several pieces at the same time.

Continuing across the top, to the right of the load button you have a drop-down that lets you choose the printer that will use the slicer. You can add to the list by choosing *Add Printer* and then

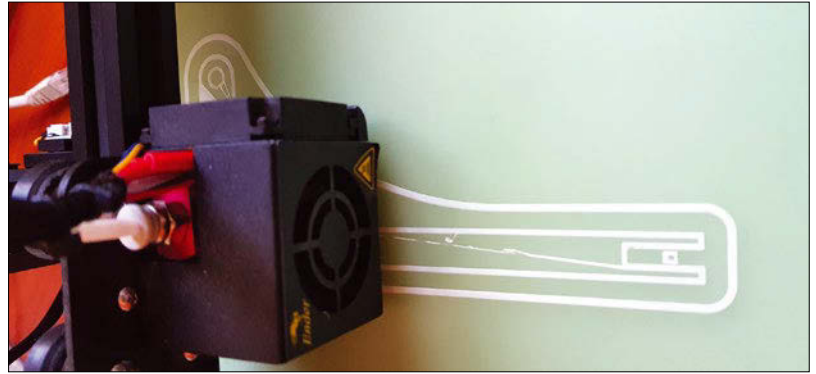


Figure 2: A skirt is an outline of the object drawn on the bed that allows you to check whether everything is working properly.

picking from a list of printer profiles. Or you can define your own printer by inputting the bed size, nozzle diameter, maximum nozzle temperature, bed temperature, etc. – this is where having your printer's specs comes in handy.

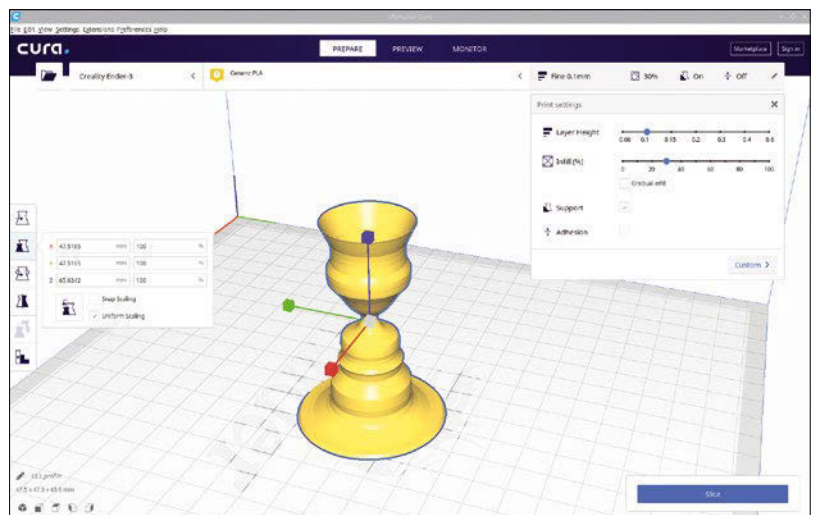
Finally, to the right, you have the print settings panel. Clicking on that will open the options that let you configure the layer height (lower gives you a smoother and more detailed print, but will take longer), infill, whether to use supports or not, and whether to use adhesion devices (like brims, skirts, and rafts).

If you click *Custom* at the bottom of the panel, you can fine-tune all these options and a few more. You can set things like whether the fan should be active all the time (something you may not want if you are using ABS) or the print speed, the shape of the infill, and all sorts of other bits and bobs.

To the left of the 3D visualization area, you have a graphical toolbar with the things you can do to any of the objects you have loaded. Click on an object and the toolbar will become active. The tools let you move, scale, rotate, and mirror the selected object, as well as configure specific print parameters for the object.

The last option allows you to place *support blockers*. Click the option and then click the object where you want to place the blocker, and a

Figure 3: Cura's interface is simple and straightforward – ideal for getting to grips with slicing.



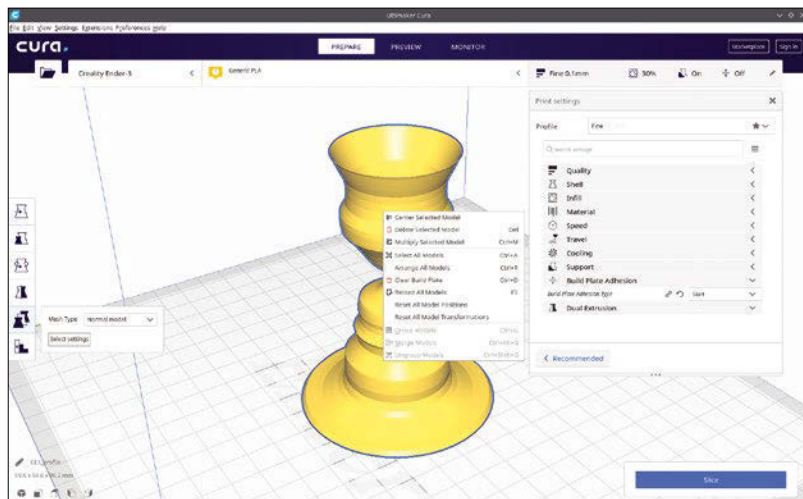


Figure 4: Underneath its veneer of simplicity, Cura hides all of the options you need to fine-tune your slicing.

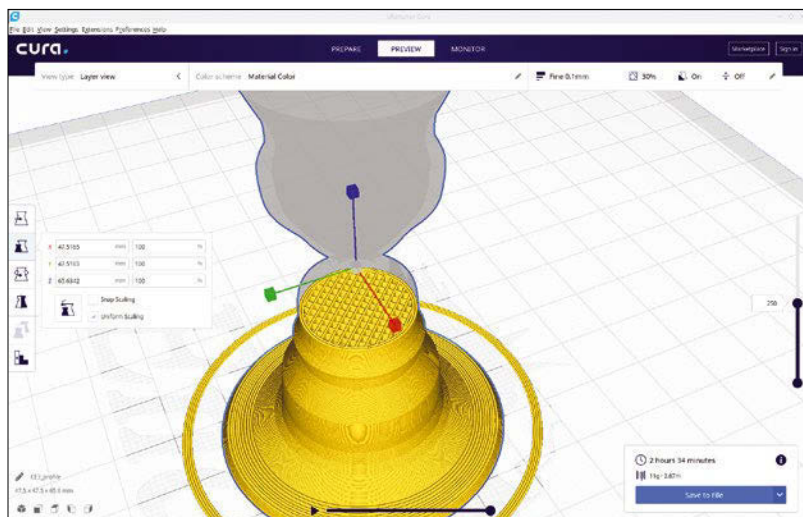
little cube will appear. You use blockers to stop the slicer from creating supports where you don't want or need them (even though the slicer thinks you do).

You can do more things with an object by right-clicking on it. That brings up a menu that lets you delete, center, clone (*multiply*), group, and do a lot of other things with your objects. So, even though Cura seems simple on the surface, really it has plenty of tools to tweak your slicing to a very precise degree (Figure 4).

Once you are done configuring the slicing, press the *Slice* button in the bottom right-hand corner of the window. When the slicing process is done, Cura will show you some important information, such as how long the print will take and how much filament it will need. These values are estimations, so take them with a pinch of salt.

Another interesting thing you can do is see a preview of the print, with supports, skirts, brims – the whole lot. At the top of the window, in the center of a dark blue bar, you have the choices of *Prepare* (the tab you have been using up until now), *Preview*, and *Monitor*. Click on *Preview* to see what the actual printed object will look like,

Figure 5: A layer in the lower half of the print showing the lattice infill as seen in Cura's Preview pane. Also notice the skirt around the base of the object.



with each layer defined (supports, infilling, etc.). Be warned that this is a lot of 3D data, and it can slow down the Cura application (or even your whole desktop) a lot.

On the right of the preview window, you will see a black vertical bar with two circular handles. Drag the handles up and down to see sections of the print (Figure 5). Along the bottom of the window is another bar with a circular handle and a “play” button. Press the button to see an animation that shows how the nozzle will move to print the selected layer.

You can use the Preview window to check how the print will go and anticipate any problems that may occur.

Once you are happy with the sliced object, the button in the lower right-hand corner now offers to save it to a G-code file on your hard disk, or, if you have an SD inserted into your computer, onto the card. You can then transfer the card to your printer and start printing.

You could also print directly from Cura via the *Monitor* tab. However, we will be seeing how to control and monitor a print from your laptop in the next issue, because that is whole different kettle of fish from slicing, which is this article's topic.

Slic3r

Slic3r [4] has slicing capabilities similar to those of Cura, but you will probably have to do more adjustments by hand, since it comes with no default settings.

When you first open the program, for some reason, the virtual bed is way off in the distance. Use the mouse wheel to zoom in (Figure 6). You can also use the left mouse button to rotate the view and the middle button to pan. Although right-clicking, holding, and dragging on an empty area also pans, it is best to save right-clicks for bringing up more options for the active object.

The settings for the print, filament, and printer itself are on the right. Click on the gear button next to the *Printer*: drop-down to open a new tab in the work area that lets you create printer profiles and save them with your printer's name. Again, you will need to have the specs of your printer handy.

The *Filament*: setting allows you to define things like heat of the nozzle and printing bed for different materials. On my Ender 3, for example, PLA works best with a nozzle heat of 200 degrees Celsius and with a bed temperature of 70 degrees Celsius for the first layer and 60 degrees Celsius for the subsequent layers. When you have defined several materials, you can pick them from the drop-down, and the G-code files will contain the correct data when you send them to the printer.

The gear next to the *Print Settings*: drop-down opens a tab where you can adjust things like supports, infill, print speed, and others.

The toolbar across the top lets you add more objects to the print, create more copies of existing objects, rotate a selected object, scale it, split a group of objects into its components, cut an object up, adjust its settings, and modify its layers' heights.

Along the bottom, there are several tabs. The 3D view is what you see in Figure 6, and 2D shows a top down view of the bed.

The *Preview* tab shows what the slicing will look like (Figure 7). In a similar fashion to Cura, down the right side of the pane, you have a slider with a circular handle. Drag the handle up and down to see what a cross section of the object will look like at different phases of the print.

The *Layers* tab shows you each layer as a 2D cross section. Again you have a slider on the right, which will let you move up and down through the object.

Moving back to the column on the right, under the configuration drop-downs, when you click the *Export G-code* button, you generate and save the *code* file that you can then load into the printer.

In theory, as with Cura, Slic3r can talk directly to the printer, but this didn't work for us. Every time we tried to *Test* the connection in the *Print Settings* tab, Slic3r crashed. This needs further research.

IceSL

IceSL [5] is probably the most intimidating of the applications you are seeing today. It also has the steepest learning curve. However, it is worth getting to know this slicer, because it comes with some features that make it unique.

You make all the adjustments for your print in the column on the left. You can define the type of printer you have in the *Printer model* drop-down and the size and orientation of your object under the *Orientation and scale* fold-down. Click on *Settings* (Figure 8) to see all the options that let you fine-tune every aspect of the print.

There are no toolbars, so you reach all the options from the menus at the top left of the window. Speaking of which, some of the most obvious features that makes IceSL different are the options hiding under the *Effects* menu. Among tools that are nifty but of dubious use (such as melting bits away from your object or adding "snow" or bumps), you'll find one that lets you paint different colors onto your model: the *Paint brushes* effect. This means that, if you have a 3D printer with multiple extruders, each extruder loaded with different types of filaments, you can actually print multicolored objects.

Pressing the *Slice!* button in the column on the right starts the slicing process and saves a G-code file to your system. When the process is over, IceSL will show the preview of the sliced object in the main window, along with a box with in-

formation on the print and sliders that let you see a cross section of the sliced object superimposed on your model (Figure 9).

You can also create objects with two or more materials by loading STL files into different

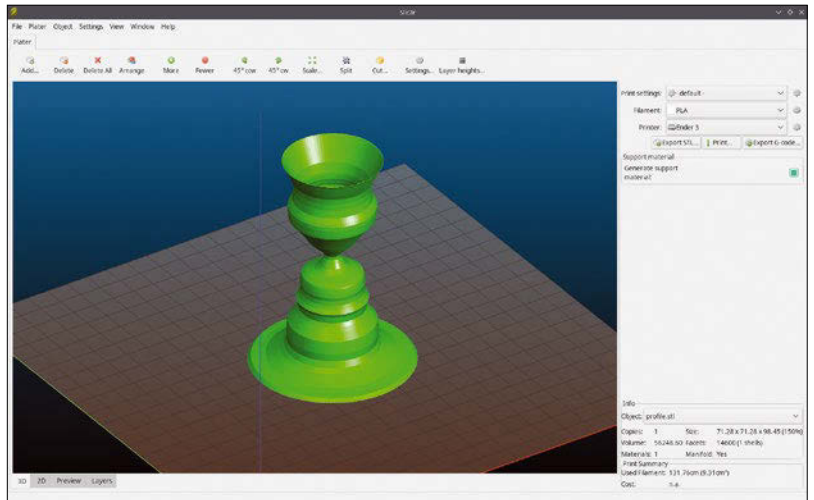


Figure 6: Slic3r offers similar features to Cura, but with less predefined defaults.

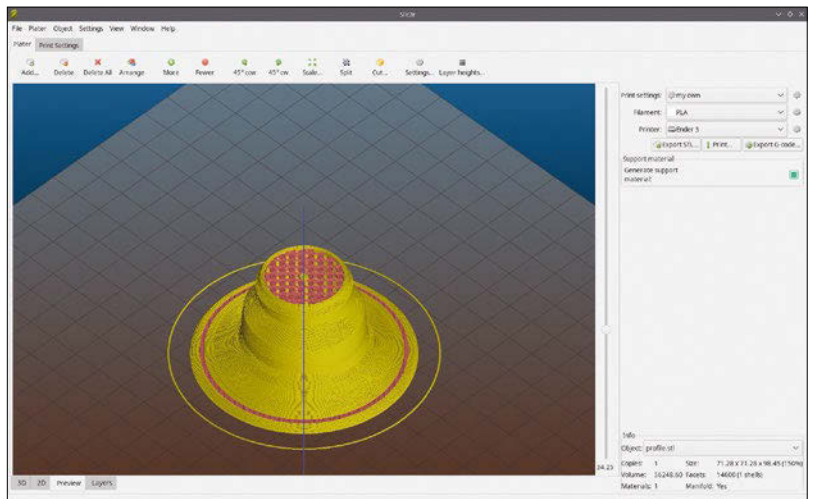


Figure 7: The *Preview* tab lets you see a cross section of the print.

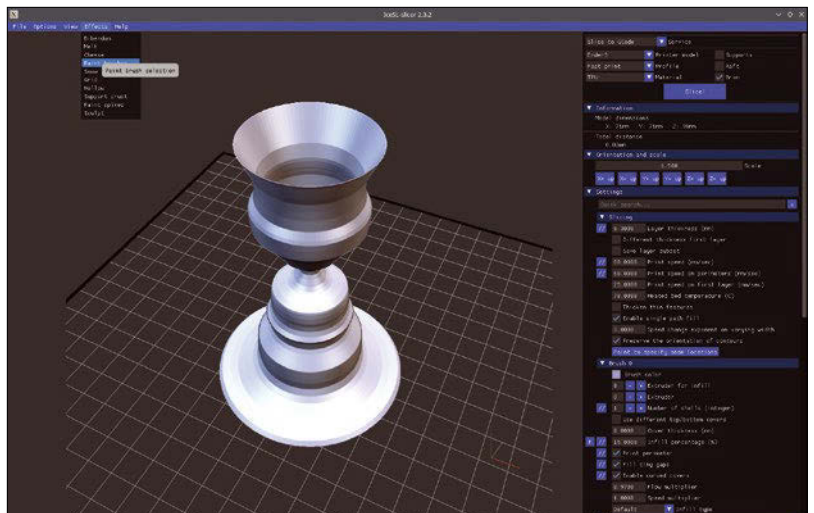


Figure 8: IceSL makes up for a lack of a friendly interface with options galore.

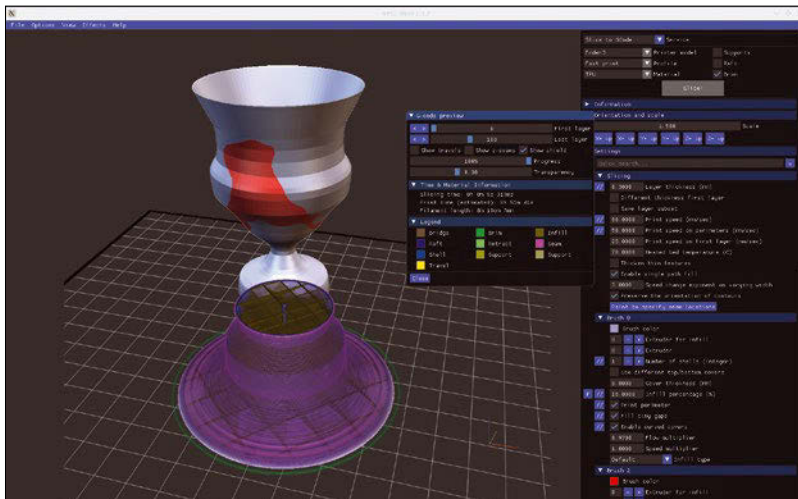


Figure 9: A sliced object with two materials is superimposed over your model.

brushes. Click on *File | Load stl on... | Brush 0*. Navigate to the STL file containing the parts that needed printing in one material and load that. Then do the same choosing *Brush 1* and loading a second STL containing the parts that need to be printed using the second material (Figure 10).

This allows you to not only print in a variety of colors, but also use filaments with different properties, like rigid and flexible filaments, or conducting and non-conducting materials. You can also assign different densities and shapes of the infill for each part.

We're going to stop here, but IceSL has much more going for it. In fact, to cover even a tenth of the features that IceSL brings to the game, we would probably need another couple of articles. Did you know, for example, that you could

model your 3D objects directly in IceSL using scripts similar to those of OpenSCAD, but written in Lua? Well, you can. You can also configure the printing options in obsessive detail using the same language.

However, at this stage and at the end of the day, what you are trying to achieve is to get the object sliced and into a G-code file that won't collapse during the print. With what you have learned about IceSL here, you have enough to do that.

Conclusion

We are not totally done yet! Next time, we will look at the final steps in the printing process and what tools you can use to control the print while it is happening.

Until then, happy printing! ■■■

Info

- [1] "Using OpenSCAD to build custom 3D pieces" by Paul Brown, *Linux Magazine*, issue 223, June 2019, pp. 90-94, <http://www.linux-magazine.com/Issues/2019/223/Tutorials-OpenSCAD>
- [2] "Technical 3D design using FreeCAD" by Paul Brown, *Linux Magazine*, issue 224, July 2019, pp. 90-95, <http://www.linux-magazine.com/Issues/2019/224/Mother-of-Invention>
- [3] Cura: <https://ultimaker.com/en/products/ultimaker-cura-software>
- [4] Slic3r: <https://slic3r.org/>
- [5] IceSL: <https://icesl.loria.fr/>

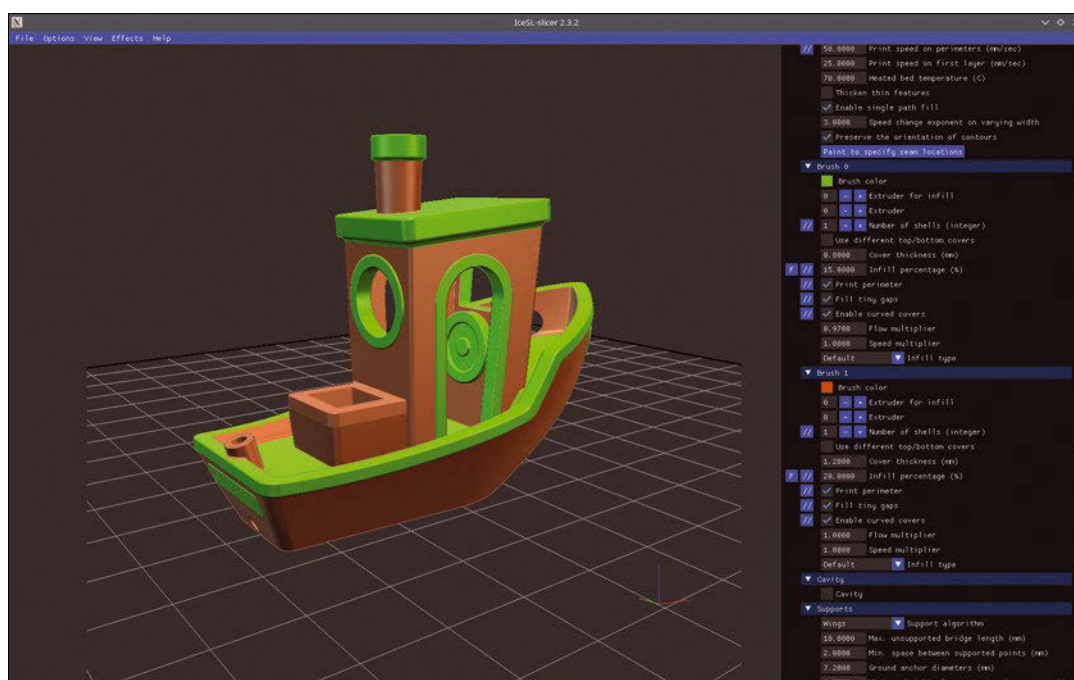
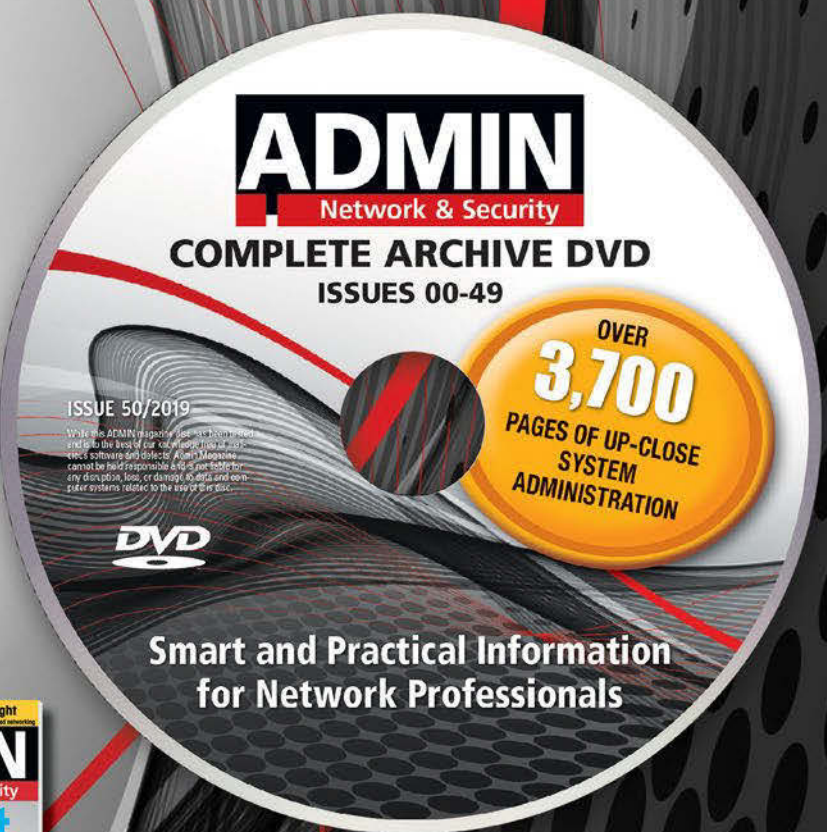


Figure 10: A classic benchy that will be printed with two different materials.

9 Years of ADMIN on One DVD



Smart and Practical Information for Network Professionals

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

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

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

ORDER NOW!
shop.linuxnewmedia.com

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.



GUADEC 2019

Date: August 23-28, 2019

Location: Thessaloniki, Greece

Website: <https://2019.guadec.org/>

The GNOME Conference: GUADEC brings together Free Software enthusiasts and professionals from all over the world. Register now for six days of talks, demos, discussion, parties, games, and more.

Akademy 2019

Date: September 7-13, 2019

Location: Milan, Italy

Website: <https://akademy.kde.org/2019>

Akademy 2019 is expected to draw hundreds of attendees from the global KDE community to discuss and plan the future of the community and its technology. Many participants from the Free and Open Source software community, local organizations, and software companies will attend.

Storage Developer Conference

Date: September 23-26, 2019

Location: Santa Clara, California

Website: <https://www.snia.org/events/storage-developer>

Attending SDC is a cost-effective way to acquire training in a host of different areas through tutorials, sessions, keynote speakers, and the opportunity to participate in the co-located plugfests. Register now, and learn from the experts all in one place.

Events

28th USENIX Security Symposium	August 14-16	Santa Clara, California	https://www.usenix.org/conference/usenixsecurity19
Open Source Summit NA	August 21-23	San Diego, California	https://events.linuxfoundation.org/events/open-source-summit-north-america-2019/
Guadec 2019	August 23-28	Thessaloniki, Greece	https://2019.guadec.org/
Akademy 2019	September 7-13	Milan, Italy	https://akademy.kde.org/2019
Linux Plumbers Conference	September 8-10	Lissabon, Portugal	https://linuxplumbersconf.org/
Cloud Foundry Summit Europe	September 11-12	The Hague, Netherlands	https://www.cloudfoundry.org/event/summit/
HPC on Wall Street	September 11-12	New York, New York	https://www.hpconwallstreet.com/
The Linux Kernel Maintainer Summit	September 12	Lisbon, Portugal	https://events.linuxfoundation.org/events/linux-kernel-maintainer-summit-2019/
All System Go!	September 20-22	Berlin, Germany	https://all-systems-go.io/
Open Networking Summit Europe 2019	September 23-25	Antwerp, Belgium	https://events.linuxfoundation.org/events/open-networking-summit-europe-2019/
Storage Developer Conference	September 23-26	Santa Clara, California	https://www.snia.org/events/storage-developer
JAX London 2019	October 7-10	London, United Kingdom	https://jaxlondon.com/
All Things Open	October 13-15	Raleigh, North Carolina	https://allthingsopen.org/
Open Source Summit + Embedded Linux Conference Europe	October 28-30	Lyon, France	https://events.linuxfoundation.org/events/open-source-summit-europe-2019/
LISA19	October 28-30	Portland, Oregon	https://www.usenix.org/conference/lisa19
DrupalCon Amsterdam 2019	October 28-31	Amsterdam, Netherlands	https://events.drupal.org/amsterdam2019

CALL FOR PAPERS

We are always looking for good articles on Linux and the tools of the Linux environment. Although we will consider any topic, the following themes are of special interest:

- System administration
- Useful tips and tools
- Security, both news and techniques
- Product reviews, especially from real-world experience
- Community news and projects

If you have an idea, send a proposal with an outline, an estimate of the length, a description of your background, and contact information to edit@linux-magazine.com.



The technical level of the article should be consistent with what you normally read in *Linux Magazine*. Remember that *Linux Magazine* is read in many countries, and your article may be translated into one of our sister publications. Therefore, it is best to avoid using slang and idioms that might not be understood by all readers.

Be careful when referring to dates or events in the future. Many weeks could pass between your manuscript submission and the final copy reaching the reader's hands. When submitting proposals or manuscripts, please use a subject line in your email message that helps us identify your message as an article proposal. Screenshots and other supporting materials are always welcome.

Additional information is available at:

http://www.linux-magazine.com/contact/write_for_us.

NOW PRINTED ON recycled paper from 100% post-consumer waste; no chlorine bleach is used in the production process.

Authors

Erik Bärwaldt	36, 72
Swapnil Bhartiya	8
Chris Binnie	22
Paul Brown	90
Zack Brown	12
Bruce Byfield	26, 58
Joe Casad	3
Răzvan T. Coloja	14
Mark Crutch	65
Nate Drake	68
Marco Fioretti	84
Jon "maddog" Hall	67
Charly Kühnast	42
Andrew Malcolm	52
Vincent Mealing	65
Pete Metcalfe	62
Graham Morrison	78
Mike Schilli	44
Mayank Sharma	30
Tim Schürmann	48

Contact Info

Editor in Chief

Joe Casad, jcasad@linux-magazine.com

Copy Editor

Amy Pettle

News Editor

Swapnil Bhartiya

Editor Emerita Nomadica

Rita L Sooby

Managing Editor

Lori White

Localization & Translation

Ian Travis

Layout

Dena Friesen, Lori White

Cover Design

Lori White

Cover Image

© andreysuslov, 123RF.com

Advertising

Brian Osborn, bosborn@linuxnewmedia.com
phone +49 89 3090 5128

Marketing Communications

Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
2721 W 6th St, Ste D
Lawrence, KS 66049 USA

Publisher

Brian Osborn

Customer Service / Subscription

For USA and Canada:
Email: cs@linuxpromagazine.com
Phone: 1-866-247-2802
(Toll Free from the US and Canada)

For all other countries:
Email: subs@linux-magazine.com

www.linuxpromagazine.com – North America

www.linux-magazine.com – Worldwide

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 © 2019 Linux New Media USA, LLC.

No material may be reproduced in any form whatsoever in whole or in part without the written permission of the publishers. It is assumed that all correspondence sent, for example, letters, email, faxes, photographs, articles, drawings, are supplied for publication or license to third parties on a non-exclusive worldwide basis by Linux New Media USA, LLC, unless otherwise stated in writing. Linux is a trademark of Linus Torvalds.

All brand or product names are trademarks of their respective owners. Contact us if we haven't credited your copyright; we will always correct any oversight.

Printed in Nuremberg, Germany by hofmann infocom GmbH on recycled paper from 100% post-consumer waste; no chlorine bleach is used in the production process.

Distributed by Seymour Distribution Ltd, United Kingdom

LINUX PRO MAGAZINE (ISSN 1752-9050) is published monthly by Linux New Media USA, LLC, 2721 W 6th St, Ste D, Lawrence, KS, 66049, USA. Periodicals Postage paid at Lawrence, KS and additional mailing offices. Ride-Along Enclosed. POSTMASTER: Please send address changes to Linux Pro Magazine, 2721 W 6th St, Ste D, Lawrence, KS 66049, USA.

Published monthly in Europe as Linux Magazine (ISSN 1471-5678) by: Sparkhaus Media GmbH, Ziebländstr. 1, 80799 Munich, Germany.

Issue 226 / September 2019

Linux Shopping

Approximate
UK / Europe Aug 03
USA / Canada Aug 30
Australia Sep 30
On Sale Date

Ubuntu, openSUSE, and Fedora all claim to be simple, intuitive, and suitable for both beginners and veterans. We help you sort through the details and choose the right Linux for your environment.

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

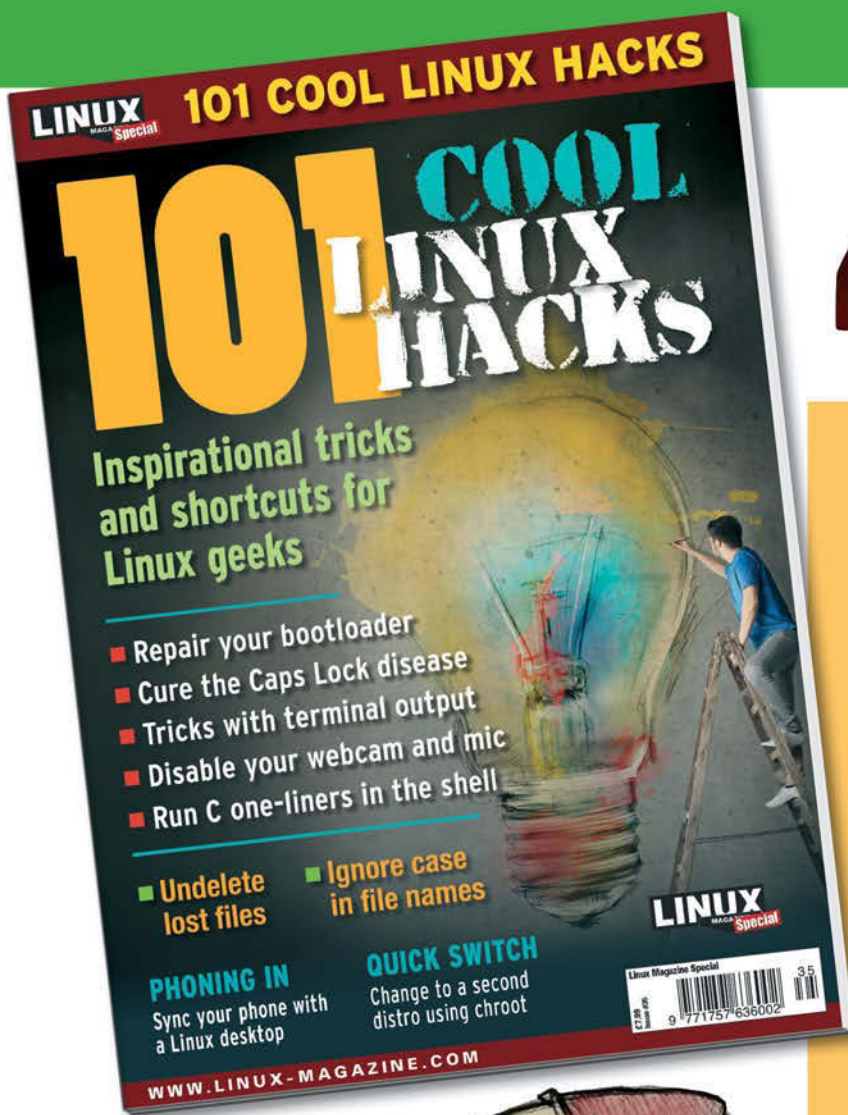


Lead Image © Golkin Oleg, 123RF.com



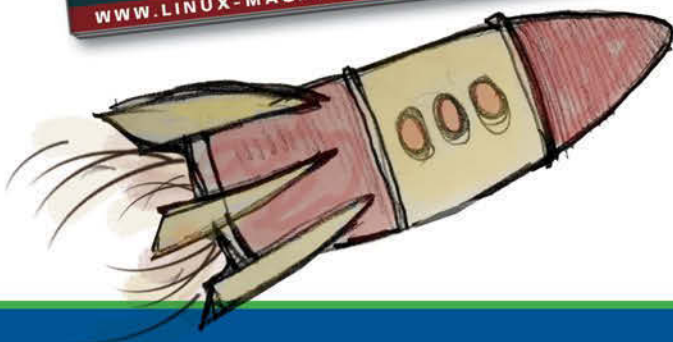
SHOP THE SHOP
shop.linuxnewmedia.com

GET PRODUCTIVE WITH 101 LINUX HACKS



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

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



ORDER ONLINE:
shop.linuxnewmedia.com/specials

SUPERMICRO

Better. Faster. Greener.

Expect Better Data Center Performance, TCO & Impact on the Environment



35% Faster | Up to 50% TCO Reduction | Reduce E-Waste
Featuring 2nd Generation Intel® Xeon® Scalable Processors



Learn More at www.supermicro.com/X11

© Supermicro and Supermicro logo are trademarks of Super Micro Computer, Inc. in the U.S. and/or other countries.

