

**20 YEARS ARCHIVE DVD** **HUGE SAVINGS!** \$39.90 VALUE! **20<sup>TH</sup> ANNIVERSARY ISSUE!** **ALL 239 ISSUES ON A SEARCHABLE DISC!**

**LINUX**  
MAGAZINE



# LINUX

**MAGAZINE**

ISSUE 240 – NOVEMBER 2020

**20 YEARS**

# IT'S ALIVE!

Breathe life into your coding experiments by creating your own simple OS

**Graphing the Pandemic**  
With freely available data

**Ajenti**  
Cool control panel for Linux servers

**Serial Protocol**  
Still a powerful option for DIY coding



**Tuxedo InfinityBook**  
Light and lively Linux laptop

**usql**  
Manage PostgreSQL, MySQL, and SQLite with a single interface

**Mistborn**  
All-in-one tool for protecting your LAN

# LINUXVOICE

- **SSH in a GUI with Muon/Snowflake**
- **maddog: Build the Future Now**
- **BeeBEEP Office Messenger**



## FOSSPicks

- digiKam 7
- NoiseTorch Noise Remover
- Satellite Imagery with felicette

## Tutorial

- Modify an object with an Inkscape extension

Issue 240  
Nov 2020  
US\$ 19.99  
CAN \$29.99



**LINUX NEW MEDIA**  
The Pulse of Open Source



# Secure and Private

All systems by TUXEDO Computers come ready to start with Linux. In addition to impressive performance, they offer comprehensive protection of your personal data and strong protection of your privacy.



## Privacy+

Intel ME, webcam, audio and WiFi deactivatable



## Fully encrypted

System and data completely protected against unauthorized access



## Zero Spyware

Verifiable security thanks to Open Source software



## Automatic Installation

System reset thanks to web-based, fully automatic installation



100%  
Linux

5

Year  
Warranty



Lifetime  
Support



Built in  
Germany



German  
Privacy



Local  
Support

# TUXEDO COMPUTERS

[tuxedocomputers.com](https://www.tuxedocomputers.com)

# HAPPY BIRTHDAY



Dear Reader,

In case you didn't notice all the fanfare on the cover, it is my privilege to inform you that you are holding Issue 240 of this magazine, and 240 divided by 12 is exactly 20, which means that we are officially 20 years old. I have long since stopped celebrating my own birthdays, but the magazine's birthday always seems like something to celebrate.

I started with Issue 48 and, just for kicks, I went back to look at Issue 48 and see what I was writing about on my first comment page. Without irony, I will tell you that one of the topics I mentioned in my first opinion column was how much I *don't* like opinion columns. It is interesting to consider that I have written 192 more since that day – a number that is difficult to comprehend, although I can safely say I have never run out of topics.

A scan of my first page 3 reveals that some of the things we talked about back then are still important to us now, and some of the things that seemed most pressing have quite fallen out of the headlines. For instance, I mentioned KDE Kontact and PHP, which we still write about. On the opposite end of this contemporary relevance spectrum was SCO, the company that tried to balance its books by claiming it owned the rights to Linux. It was quite scary at the time for Linux, because SCO seemed to have deep pockets and lots of important lawyers, but the open source community held fast. Eventually SCO's efforts fizzled, and they went bankrupt.

Beyond the relics and the relevant was one topic that falls somewhere between and is, thus, an interesting time capsule. That topic was Mozilla XUL. We know that Mozilla is still relevant, because I wrote about them last month. Unfortunately, last month I reported on their layoffs and restructuring, but they had many years of glory between then and now. XUL is still around, although it doesn't get that much attention anymore – even from Mozilla – and it isn't as important since the arrival of HTML5. At the time, it seemed like XUL was undoubtedly cool, but no one was really sure what was going to happen with it.

For the Mozilla project, 2004 was an exciting year. The Mozilla Foundation had launched a year earlier as an independent nonprofit and had received the rights to the old Netscape browser code. Many commentators had already written them off – sure, they still developed the old Mozilla Application Suite, but they seemed like one of those massive codebases that drifted around the open source ocean with no clear sense of purpose.

Then, only three months after Issue 48 appeared in print, Mozilla announced the first release of Firefox – a whole new browser built around existing Mozilla technologies like XUL

and the Gecko engine. Firefox was a big hit – not just with Linux folks but with Windows users, many of whom quickly agreed it was better than the clunky Internet Explorer. Firefox market share rose quickly, eventually surpassing IE and reigning for a time as king of the hill until Chrome arrived in 2008. Mozilla Firefox was a big morale boost to the FOSS community and a shot across the bow to conventional software companies, proving that open source software could compete directly, not just on Linux, but on commercial desktop systems. My first column was written at that curious time when the Mozilla Foundation had already launched, and development tools like XUL were already out there in the world, but before Mozilla started its epic comeback that began with Firefox. The lesson is that things change fast, and you never know what is just around the corner. History emerges from the cloud of maybes and what-ifs that surrounds us in the present.

Magazines come and go all the time in the global publishing industry, and 20 years is actually quite a run. We haven't been around for as long as *The Observer* or *National Geographic*, but we have certainly outlived many of our peers, especially in the chaotic high-tech field. I'm proud that we're still going after 20 years and still prouder that we are tooled up and ready for another 20.

Our well-oiled and finely tuned publishing engine is only part of the equation, though, so keep watching the commercial spaces near you. Support your local bookstore, but keep in mind that some retailers are reducing their shelf space, so if you don't see us, *Linux Magazine* is always available to order online:

Subscribe: <https://bit.ly/Get-Linux-Magazine>

Single Issue: <https://bit.ly/Linux-Newsstand>

Joe

Joe Casad,  
Editor in Chief





**20 YEARS**

# LINUX MAGAZINE

## WHAT'S INSIDE

### Build a whole operating system?

Seriously? Well maybe not a Linux, but if you're interested, you can implement the basic features of an experimental OS using resources available online. We can't show you the whole process, but we'll help you get organized and take your first steps. Also inside:

- **Open Data** – Graph the COVID-19 pandemic using free data in REST APIs (page 34).
- **Mistborn** – Use this one handy tool to secure services on your home network (page 48).

Check out MakerSpace for a study of the serial protocol and how to use it in DIY programming, and turn to this month's LinuxVoice for a tutorial on building an Inkscape extension.

## SERVICE

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

## NEWS

### 08 News

- Lenovo Now Offering Fedora Linux as an Option
- System76 Launches new High-End Laptop
- Mozilla Lays Off Staff, Receives More Cash
- VirtualBox Now Supports Linux Kernel 5.8
- Three Major Threats to Linux Discovered
- Linux Kernel 5.8 is Now Available

### 11 Kernel News

- A Stone So Large ...
- Global Stats-Gathering Interface
- The Kernel Development Process

## REVIEWS

### 20 Tuxedo InfinityBook S 14 v5

Are you in the market for a slim, light, and power-packed laptop? We test an upgradeable machine from Tuxedo.



## COVER STORIES

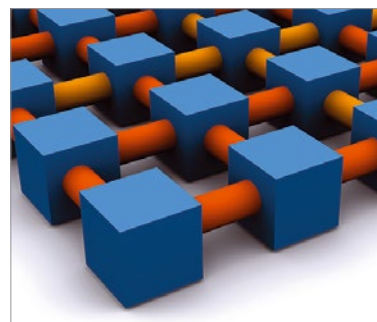
### 14 Building a Hobby OS

Reading and understanding the complete Linux kernel is a challenging project. A hobby kernel lets you implement standard OS features yourself in a few hundred lines of code.



### 22 Distro Walk – Governance

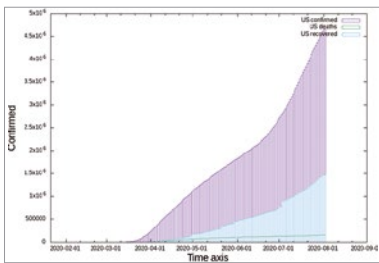
Whether you are a user or a developer, knowing how a distribution governs itself can help you choose a Linux distro.



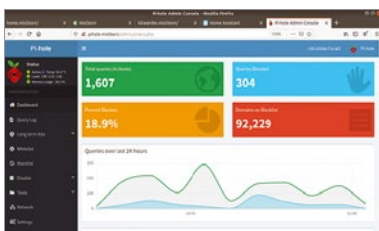


## IN-DEPTH

- 26 **Ajenti**  
Ease the process of managing Linux installations remotely with the web-based Ajenti control panel.
- 30 **Command Line – diffoscope**  
Extend the power of diff beyond the plain text or HTML file.
- 34 **Open Data**  
Lots of COVID-19 data is available through online REST APIs. With a little ingenuity and some open source tools, you can extract and analyze the data yourself.



- 40 **Programming Snapshot – Go Geofuzzer**  
Mike Schilli loves his privacy. That's why he created a Go program that adds a geo-obfuscation layer to cellphone photos before they are published online.
- 44 **Usql**  
Usql is a useful tool that lets you manage many databases from one prompt.
- 48 **Mistborn**  
Mistborn bundles important Internet services on your home network and secures them with a WireGuard VPN tunnel, Pi-hole, iptables rules, and separate containers.



- 53 **Charly – pwquality**  
Regular password changes are a thing of the past: Strong passwords for each individual service provide more protection.

## MAKERSPACE

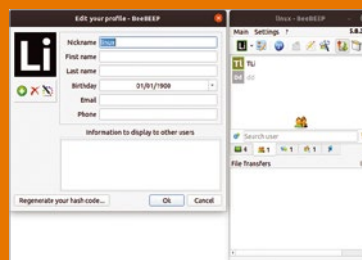
- 54 **Serial Communication Protocol**  
We explore serial communications, from the electrical specs to protocols and libraries, with an example of serial communication with an Arduino.
- 62 **WS2812 RGB LEDs**  
Control a WS2812 LED matrix with the Raspberry Pi.



SEE PAGE 6 FOR DETAILS

# LINUXVOICE

- 67 **Welcome**  
This month in Linux Voice.
- 68 **Doghouse – Future of FOSS**  
To build the future of FOSS, we need to focus on communicating its value – especially to young people.
- 70 **Muon/Snowflake**  
Muon/Snowflake lets you manage SSH access via an easy-to-use GUI with a wealth of useful functions.
- 74 **BeeBEEP**  
This complete chat solution allows you to send messages and share files without relying on the cloud or complicated office infrastructure.
- 78 **FOSSPicks**  
This month Graham explores Carla, digiKam 7, NoiseTorch, diskonaut, Surge 1.7, Trigger Rally, and more.
- 84 **Tutorial – Tremble**  
Writing your own extension for the Inkscape vector graphics app opens up a whole world of possibilities.
- 90 **20 Years of Linux Magazine**  
Editor-in-chief Joe Casad tells our enchanting 20-year story.



# Linux Magazine 20<sup>th</sup> Anniversary Archive DVD

## Linux Magazine Archive DVD

This month's DVD includes the 2020 edition of the Linux Magazine Archive DVD – every previous issue of *Linux Magazine* on a single, searchable disc. Browse the pages of every article we've ever published, and experience our special brand of technical yet accessible how-to insights.

If you want to learn about a common tool in the open source space, search this disc – we've probably written about it more than once over the last 20 years. Catch all the articles you missed – on scripting, system administration, security, containers, cloud computing, Raspberry Pi, and more! Discover desktop applications that will save you time and simplify your life, and relive important moments in the history of Linux: the SCO case, the Novell/Microsoft pact, the birth of Git, the mobile revolution, ...

All the issues on this disc would cost you about \$3,800 in our shop – if they all still existed, but many older issues are out of print: **The only way to experience them is through the Linux Magazine Archive DVD!**

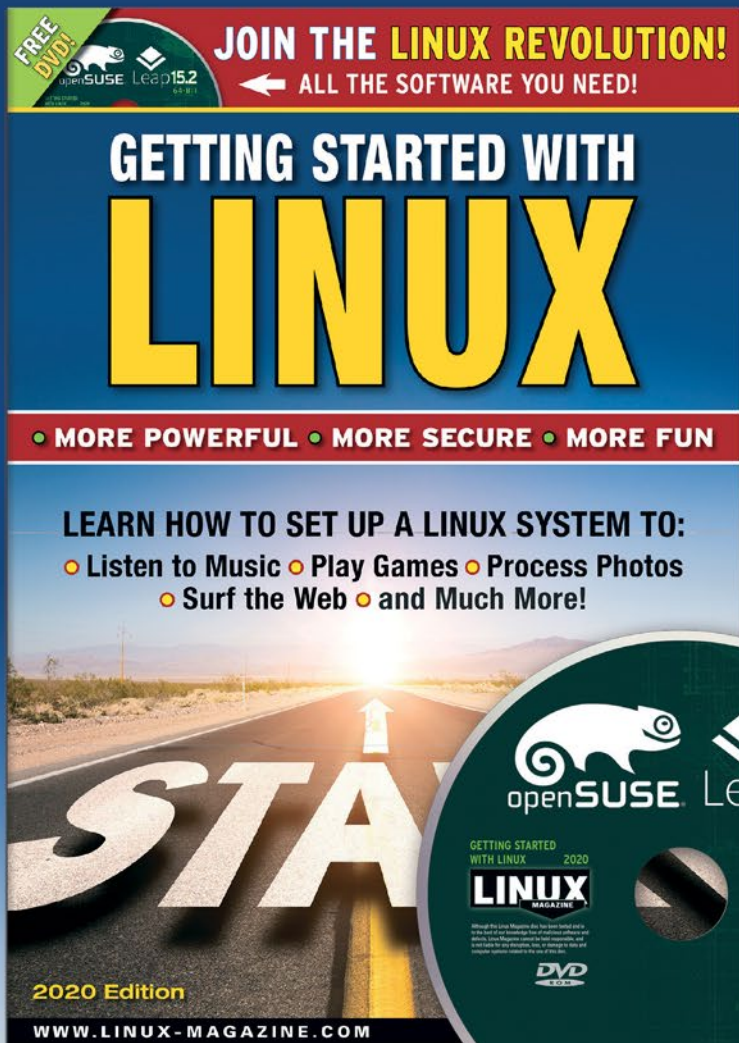


Defective discs will be replaced.  
 Please send an email to [subs@linux-magazine.com](mailto:subs@linux-magazine.com).

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.



# Hit the ground running with Linux



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

This single issue shows beginners how to:

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



**ORDER ONLINE: [shop.linuxnewmedia.com/specials](http://shop.linuxnewmedia.com/specials)**



# NEWS

Updates on technologies, trends, and tools

## THIS MONTH'S NEWS

- 08** • Lenovo Now Offering a Fedora Linux Option
- System76 Launches New High-End Laptop
- 09** • Mozilla Lays Off Staff, Receives More Cash
- VirtualBox Now Supports Linux Kernel 5.8
- More Online
- 10** • Three Major Linux Threats Discovered
- Linux Kernel 5.8 is Now Available

### Lenovo Now Offering Fedora Linux Option

Lenovo kicked off the Summer of 2020 by claiming it would certify both their ThinkPad and ThinkStation hardware lines for Linux. The hardware giant is making good on that promise and is now offering a laptop preloaded with Fedora Linux.

At the moment, there's only one device offered with Fedora. Said device is the ThinkPad X1 Carbon Gen 8 (<https://www.lenovo.com/us/en/laptops/thinkpad/thinkpad-x1/X1-Carbon-Gen-8-p/20U9CTO1WWENUS2/customize>). The base price of this laptop is \$2,145, but with the coupon THINKPROMO that price is cut to \$1,287. The base model includes a 10th Generation Intel Core i5-10210U CPU (1.60GHz) with 4 Cores, 8 Threads, and a 6MB cache, 8GB of LPDDR3 RAM, 256GB PCIe SSD, 14.0-inch FHD (1920x1080) IPS anti-glare display, a 720p HD camera, Intel Wi-Fi 6, and a fingerprint reader. Of course, you can configure up to a 10th Generation Intel Core i7, 16GB of RAM, a 1TB PCIe SSD, and a 4K UHD (3840x2160) IPS display for a total of \$3,451 (using the THINKPROMO coupon drops that price to \$2,070.60).

One very exciting feature about the X1 Carbon is that it promises a massive 19.5 hours of battery life and offers Rapid Charge of up to 80% in an hour.

Lenovo is using a Fedora Workstation image with zero customizations, so it will include the GNOME Desktop Environment.

### System76 Launches New High-End Laptop

System76 is not one to rest on reputation. Instead of being content with an already impressive lineup, they're always looking to up their game. With the release of the Bonobo WS (<https://system76.com/laptops/bonobo>), the Denver, Colorado company has done just that.

The new Bonobo WS is centered around Intel's 10th Generation Comet Lake CPU and is matched with an NVidia GeForce RTX 2080 Super Desktop GPU.

You can spec the Bonobo WS with up to 128GB of RAM and up to 24TB of NVMe storage. As far as display, the Bonobo WS offers a 17-inch matte display with either 1080p or 4K.

As you can probably guess, this isn't your average laptop. It's also a thick beast of a machine weighing in at approximately 8.3 pounds, so the Bonobo WS isn't going to be the laptop you grab when you're jetting about and need the



© System76 <https://system76.com/>

ability to browse the Internet or send an email or two. This machine is all about one thing – power – and it delivers.

The base price for the Bonobo WS is \$2,300, which includes a Core i5 10600K CPU, NVidia RTX 2060 GPU, 17.3-inch 1080p display, 16GB of RAM, and 240GB of NVMe internal storage. If you really want to go crazy, you can spec the Bonobo WS all the way out to \$11,210 (which includes an i9-10900K CPU, massive amounts of RAM, storage, and a 4K display).

Configure and purchase your System76 Bonobo WS now (<https://system76.com/laptops/bonw14/configure>).

## Mozilla Lays Off Staff, Receives More Cash

Mozilla isn't a stranger to struggle. Be it market share or financial issues, the foundation that delivers the most popular open source browser and email client to the Linux platform has always had to fight to keep its head above water.

So it should come as no surprise that last week the foundation laid off almost a quarter of its staff. To this issue, Mitchell Baker (Mozilla CEO) said:

"This will strengthen our ability to build and invest in products and services that will give people alternatives to conventional Big Tech."

Of course, the news doesn't end there. On the same day Mozilla released those employees, they inked a new deal with Google. Said deal is rumored to be worth nine figures (landing them \$400 to \$450 million a year between now and 2023) and ensures Google is the default search engine on the open source browser.

This deal is sure to turn heads, given the timing. However, Mozilla is repositioning themselves such that they won't have to rely so much on Google. The goal is to be more self-sufficient. This means Mozilla will look toward new services and technology to bolster their bottom line.

For more information about the announcement, check out Baker's official statement (<https://blog.mozilla.org/blog/2020/08/11/changing-world-changing-mozilla/>).



© Fernando Gregory, 123RF.com

## VirtualBox Now Supports Linux Kernel 5.8

Linus Torvalds said that the Linux Kernel 5.8 was the largest release of all time. Although the vast majority of the changes to the kernel was code cleanup, there were a number of additions to the supported hardware and even some new features. But due to the size of this new kernel, one would have thought it might have taken considerable time for the likes of VirtualBox to come out with support. That is not the case.

With the release of VirtualBox 6.1.14, the newest kernel is officially supported. This means you can not only run VirtualBox on Linux distribution hosts that use the 5.8 kernel, you can also run those same distributions as guest virtual machines.

But don't think the 6.1.14 maintenance release is only about supporting the new kernel. This latest release also includes plenty of bug fixes for Windows and macOS hosts and other fixes such as:

- Fixing a regression in HDA audio emulation
- Fixing webcam passthrough and audio input issue on macOS Mojave and newer
- Fixing an issue in Windows host serial port implementation
- Fixing an issue when copying HTML data to the shared clipboard

Of course, the big news is the support for the new kernel. For a more complete listing of the changes, read the changelog (<https://www.virtualbox.org/wiki/Changelog>).

Download the latest version of VirtualBox (as well as the Extension Pack and the SDK) from the official download page (<https://www.virtualbox.org/wiki/Downloads>).

## MORE ONLINE

### Linux Magazine

[www.linux-magazine.com](http://www.linux-magazine.com)

### ADMIN HPC

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

#### SMART Devices

- Jeff Layton

Most storage devices have SMART capability, but can it help you predict failure? We look at ways to take advantage of this built-in monitoring technology with the smartctl utility from the Linux smartmontools package.

### ADMIN Online

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

#### Cloud Security with AWS GuardDuty

- Raul Lapaz Valeiras

AWS GuardDuty uses machine learning and threat intelligence to identify malicious activity for continuous security monitoring in the cloud.

#### Goodbye Google Cloud Print

- Carsten Mickleit

With the discontinuation of Google Cloud Print, we look at printing going forward in the enterprise.

#### Secrets and Certificate Management

- Chris Binnie

Vault is a highly secure, trusted place to keep your secrets and certificates.

## Three Major Linux Threats Discovered

In less than a week, it has been reported that Linux has been found to be vulnerable to three different attacks. This should come as no surprise, given the steady rise in popularity Linux has enjoyed over the last year.

The first attack is a cryptomining dedicated denial of service (DDoS) attack, named Lucifer. This hybrid DDoS botnet was first known for infecting Windows machines with Monero cryptomining bots. That attack is now scanning for and infecting Linux servers and desktops. The Linux version of the Lucifer botnet has the same capabilities as the Windows version, but it can also be used in HTTP-based DDoS attacks.



© AlienCat, Fotolia.com

The next attack, dubbed FritzFrog, is another botnet that was discovered breaching SSH servers starting in January 2020. This bot, written in Golang, has been found to target systems within the government, education, and finance sectors. FritzFrog assembles and executes its in-memory payload. Once on a system, FritzFrog communicates, via an encrypted channel, using over 30 commands. The malware then

spawns multiple threads to facilitate replication, deployment, and growth. Guardicore Labs has created a script ([https://github.com/guardicore/labs\\_campaigns/tree/master/FritzFrog](https://github.com/guardicore/labs_campaigns/tree/master/FritzFrog)) that can detect FritzFrog infections.

Finally, Drovorub is a toolset that creates a backdoor on Linux machines that enables file downloads and uploads, as well as the execution of commands as root and port forwarding of network traffic. Worst of all, Drovorub implants a kernel rootkit, which is enhanced with additional capabilities. To mitigate Drovorub, admins are warned to upgrade their Linux systems immediately (including the kernel). If your servers and desktops are running any kernel newer than 3.7, you should be safe. Of course, 3.7 is quite an old kernel, so chances are good you are already free from the effects of this malware.

## Linux Kernel 5.8 Is Now Available

Linus Torvalds (the creator of Linux) has called the 5.8 kernel (<https://cdn.kernel.org/pub/linux/kernel/v5.x/linux-5.8.tar.xz>) the "Biggest release of all time." That kernel is now available for installation.

The majority of the 5.8 kernel is code cleanup, driver support, security improvements, and low-level optimizations, which translates into not a lot of user-facing features. To put this into perspective, Torvalds said this:

*"But again, 5.8 is up there with the best, despite not really having any single thing that stands out. Yes, there's a couple of big driver changes (habanalabs and atomisp) that are certainly part of it, but it's not nearly as one-sided as some of the other historical big releases have been."*

Some of the new features include support for Qualcomm Adreno 405/640/650, AMDGPU TMZ, Tiger Lake SAGV, POWER10 CPUs, Arm SoC, Tiger Lake Thunderbolt (for Intel's Gateway SoCs), as well as a new AMD Energy Driver.

Although you might be tempted to upgrade to the new 5.8 kernel, remember that your distribution of choice may not include their own supported drivers and patches. Because of this, you might not want to jump right in and install the latest kernel on a production machine. Install 5.8 on a test platform and kick the tires before you decide to go ahead on migrating any mission critical machine.

# 5.8



Get the latest news  
in your inbox every  
two weeks

Subscribe FREE  
to Linux Update  
[bit.ly/Linux-Update](https://bit.ly/Linux-Update)



# Zack's Kernel News

## A Stone So Large ...

A lot of kernel patches just fix little things or make something slightly more convenient for system administrators and kernel developers. Recently, Joe Perches pointed out that the Arm architecture could conceivably invoke the Linux kernel with a command line that was longer than what `printk()` was able to output. So you could launch the kernel, but you couldn't then go back and see exactly how you'd done it.

He posted a patch to split long lines up so that `printk()` could output them.

Sergey Senozhatsky liked the overall idea, but pointed out a bug in Joe's code – he noted that `printk()` would also output a bit of prefix text, to let the user know what was being output. In this case, the prefix was “*Kernel command line*”. Sergey reported that Joe's patch neglected to take the length of that prefix text into account when seeing whether the full output string was over the maximum size. So the text could still be slightly too big for `printk()`. Joe agreed and fixed it.

Sergey also commented that it would be good to have some way to let the calling routine know what `printk()`'s maximum line length was, “so that `printk()` will still have sane limits and won't print a 1G string for example.” He suggested exporting a variable that calling routines could see, which simply gave the maximum line length `printk()` supported, such as 256 characters or something around there. Joe agreed with that, saying probably a simple `#define` would do the trick. So there would be the double solution: `printk()` would split lines that were too long, and calling routines would know how small to split their data before calling `printk()`.

Problem solved! Except... Sergey suggested a different approach, in fact one that was already in use in the kernel's `print_modules()` function, which also frequently had to output tons of data. It used `memchr()` and `pr_cont()` to split its input in a loop. The benefit over

Joe's `printk()` implementation being that `pr_cont()` had its own overflow control.

Joe was fine with this – though he did notice that `pr_cont()` also had a maximum size limit of 8,192 characters. But for the purposes of the Arm architecture, he added that kernel command lines would not hit that limit, so it didn't much matter either way.

Andrew Morton sensed controversy on the horizon between the two proposals. He suggested an alternative approach, creating a new `putsk()` function, which would be a kernel-specific version of the `puts()` function from the C library. The main benefit, Andrew said, would be to replace a lot of `printk()` invocations inside the kernel, which would reduce the size of the compiled binary.

Joe didn't care either way, but Sergey was not enthusiastic about this suggestion. He said, “A function that prints the kernel command line is a bit different in the way that we can split command line arguments – they are space separated, which is very convenient – so we would `pr_cont()` parts of command line individually. This has an advantage that we won't `\r\n` in the middle of the parameter.”

In fact, the conversation descended into the intricacies of potential implementations, and it never came up for air. The discussion ended after a few more emails, with no clear decision in any direction.

Obviously, there'll be a solution eventually – the Arm architecture can't be allowed to have a command line so long that even it can't see it. But the interesting thing is the care and attention given to such a minor feature, even when multiple prospective solutions are available.

## Implementing a Global Stats-Gathering Interface

Often the need for a particular feature will only gradually emerge into the collective kernel developer consciousness as a variety of one-off implementations start popping up everywhere. Then finally someone says, “shouldn't all this



**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**.

stuff be in just one place?” And that’s when the fun begins. Or maybe multiple people all start simultaneously implementing their own visions for a glorious unity without saying anything to anyone else. And the fun starts there as well.

Recently, Emanuele Giuseppe Esposito said, “There is currently no common way for Linux kernel subsystems to expose statistics to userspace shared throughout the Linux kernel; subsystems have to take care of gathering and displaying statistics by themselves, for example in the form of files in debugfs. For example KVM has its own code section that takes care of this in `virt/kvm/kvm_main.c`, where it sets up debugfs handlers for displaying values and aggregating them from various subfolders to obtain information about the system state.”

In fact, Emanuele’s purpose was to introduce the `statsfs` filesystem written by Paolo Bonzini in 2019 to replace KVM’s use of debugfs for statistics. In his announcement post on the KVM list in November, Paolo had said, “`statsfs` is a proposal for a new Linux kernel synthetic filesystem, to be mounted in `/sys/kernel/stats`, which exposes subsystem-level statistics in sysfs. Reading need not be particularly lightweight, but writing must be fast. Therefore, statistics are gathered at a fine-grain level in order to avoid locking or atomic operations, and then aggregated by `statsfs` until the desired granularity.”

Paolo’s goal was to create a special-purpose filesystem with a stable API that other parts of the kernel could also use for the same purpose of gathering and processing statistics.

Now, apparently, `statsfs` was ready to be adopted by the wider community of subsystems within the kernel. In Emanuele’s announcement he said, “In this patch series I introduce `statsfs`, a synthetic ram-based virtual filesystem that takes care of gathering and displaying statistics for the Linux kernel subsystems. The file system is mounted on `/sys/kernel/stats` and would be already used by `kvm`.”

He went on to say, “`Statsfs` offers a generic and stable API, allowing any kind of directory/file organization and supporting multiple kind[s] of aggregations (not only sum, but also average, max, min and `count_zero`) and data types (all unsigned and signed types plus boolean). The im-

plementation, which is a generalization of KVM’s debugfs statistics code, takes care of gathering and displaying information at run time; users only need to specify the values to be included in each source.”

David Rientjes was very excited to see this – in fact, he said he’d been looking into doing something similar with Jonathan Adams. So he had some very specific comments and concerns. His main desire was optimization. David saw some dangerous overhead in the way individual values were gathered in preparation for generating statistical data. Values could be as fine-grained as the amount of RAM used by a single data structure. Minimizing the number of operations required for each saved value, David said, would be crucial.

By way of suggestions, David said:

“A couple of ideas:

- *an interface that allows gathering of all stats for a particular interface through a single file that would likely be encoded in binary and the responsibility of userspace to disseminate, or*

- *an interface that extends beyond this proposal and allows the reader to specify which stats they are interested in collecting and then the kernel will only provide these stats in a well formed structure and also be binary encoded.*

“We’ve found that the one-file-per-stat method is pretty much a show stopper from the performance view and we always must execute at least two syscalls to obtain a single stat.”

Emanuele replied that a binary format for holding data in `statsfs` had been considered from the beginning and seemed feasible.

Jim Mattson also stood up in favor of a binary format for encoding raw values. He felt that storing things in ASCII format was simply not scalable.

Paolo, who wrote the original code for the `statsfs` filesystem, said:

“I am totally in favor of having a binary format, but it should be introduced as a separate series on top of this one – and preferably by someone who has already put some thought into the problem (which Emanuele and I have not, beyond ensuring that the `statsfs` concept and API is flexible enough).

ASCII stats are necessary for quick userspace consumption and for backwards compatibility with KVM debugfs (which is not an ABI, but it’s damn useful and

*should not be dropped without providing something as handy), so this is what this series starts from.”*

But David was worried. He said that once the new filesystem was merged into the kernel, then `/sys/kernel/stats` could indeed be considered an application binary interface (ABI) and would therefore have much stricter controls on whether and how it could ever be changed.

Linus Torvalds has traditionally been utterly unwilling to accept any changes to an ABI that is already in the kernel, unless such a change is necessary to fix a security hole, or if developers can be reasonably certain that no one – but no one! – is still using that particular ABI.

The reason for Linus’s reluctance is tied to the difference between an ABI and an application programming interface (API). If an API changes, it means that source code using a particular library call will have to be changed in order to use a new library call. This is generally no problem, because, if you have the source code, you can fix it to use the new call and then recompile your program. Presto!

When an ABI changes, however, it means that a compiled binary using a particular internal publicly exposed kernel feature can no longer find that kernel feature. The user program stops working, and it will never work again. The kernel developers can’t make the assumption that users will definitely have access to the source code for a particular binary. Maybe the binary is a closed-source product from a now-defunct company. Maybe the source code has simply been lost in the mists of time.

Linus considers it unacceptable to break user space in this way. If something can run on Linux, then it must continue to be able to run on Linux.

The same problem doesn’t exist for API changes. By definition, if the interface used by your source code changes, you therefore have the source code and can write a patch for it to use the new interface. Because an existing compiled binary is so much harder to patch in that way, it’s basically not reasonable to expect anyone to be able to do it.

This is David’s concern about adopting `statsfs` into the kernel before ironing out the then-permanent ABI issues.

Paolo felt that binary and ASCII data formats should complement each other –

they should each be available. The binary format would be available for highly efficient operations, while the ASCII format would be available for quick-and-easy user operations.

He affirmed that as far as the binary format itself went, he hadn't thought about it and wasn't sure what feature set to aim for. But he agreed that the ASCII format should be an optional item – easy to remove via mount options or during kernel compilation.

Meanwhile, Jonathan, who had been working with David and others at Google to develop metricsfs, a project with similar goals as statsfs, weighed in. Regarding this new project, Jonathan said, "It's designed in a slightly different fashion than statsfs here is, and the statistics exported are mostly fed into our OpenTelemetry-like system. We're motivated by wanting an upstreamed solution, so that we can upstream the metrics we create that are of general interest, and lower the overall rebasing burden for our tree."

He added, "I agree with the folks asking for a binary interface to read statistics, but I also agree that it can be added on later. I'm more concerned with getting the statistics model and capabilities right from the beginning, because those are harder to adjust later."

He offered to collaborate on an overall statsfs design and proposed several ideas for Emanuele and Paolo to consider. These ideas were highly welcome, and the two groups descended into a technical design discussion.

In the midst of that discussion, Jonathan offered an interesting view of Google's metricsfs behavior. He said:

*"Here's a summary of the types of statistics we use in metricsfs in google, to give a little context:*

- integer values (single value per stat, source also a single value); a couple of these are boolean values exported as '0' or '1'.*
  - per-CPU integer values, reported as a <cpuid, value> table*
  - per-CPU integer values, summed and reported as an aggregate*
  - single-value values, keys related to objects:
 
    - many per-device (disk, network, etc) integer stats*
    - some per-device string data (version strings, UUIDs, and occasional statuses.)**
  - a few histograms (usually counts by duration ranges)*
  - the 'function name' to count for the WARN statistic I mentioned.*
  - A single statistic with two keys (for livepatch statistics; the value is the livepatch status as a string)*
- "Most of the stats with keys are 'complete' (every key has a value), but there are several examples of statistics where only some of the possible keys have values, or (e.g. for networking statistics) only the keys visible to the reading process (e.g. in its namespaces) are included."*

The discussion continued without any significant controversy. It seems clear that the two groups of developers want almost


exactly the same thing and will eventually solve the technical implementation details. So we can look forward to a nice, clean, standardized statistics-gathering filesystem at some point in the not-too-distant future.

## The Kernel Development Process

Linus Torvalds has changed his preferred development process many times over the years – in fact so much so that discrete changes can be hard to identify, and the whole process seems to become much more fluid and cultural, rather than formal and rigid.

There was a short exchange recently in which David Howells wanted to submit a few minor bug fixes directly to Linus, but wasn't sure when would be the right time to send them. In recent years, Linus has become stricter about when he wants new features versus fixes. But exactly when and which has not been entirely clear. In this particular case, the next release was going to be Release Candidate 1, and David wasn't sure what Linus's plans for that particular release were.

So David asked. Linus replied, "No, I'll take fixes at any time, and the better shape rc1 is in, the happier everybody will be and the more likely we'll have testers."

And that was that. So, fixes are always welcome at any time during the development cycle. It's only for new features and perhaps more complex changes that submitters need to consider exactly which release candidate is the right one for them. 





### Building a hobby OS with Bochs and Qemu

# Hobby Time

Reading and understanding the complete Linux kernel is a challenging project. A hobby kernel lets you implement standard OS features yourself in a few hundred lines of code. *By Hans-Georg Eßer*

Everyone who works professionally with Linux is used to building software from the source code, perhaps implementing small changes, automating routine work with shell scripts, or developing their own software in one of the many current programming languages. Tinkering with your own, completely new operating system, on the other hand, is a pretty unusual pastime. If you start from scratch, it will take a long time before your system is useful for anything.

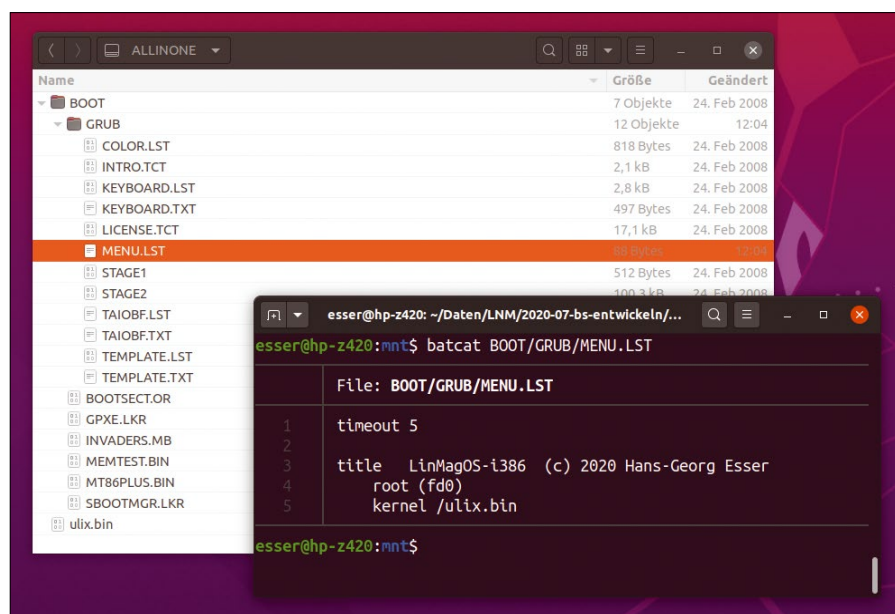
If you are looking for a challenging amateur project (for example, if you are a computer science student) or you want a better understanding of the theoretical basics of interrupts, memory management, scheduling, and other OS features, working on your own kernel can provide valuable insights. Linus Torvalds actually created Linux through a similar tinkering project. In 1991, he posted in a Minix news group, “I’m doing a (free) operating system (just a hobby, won’t be big and professional like gnu) for 386(486) AT clones.” [1]

A development environment for an operating system is more complex than one for an application, because you cannot simply compile the source code and run it on a trial basis. Instead, you need to create a bootable disk that can be used to boot a VM or emulated PC. It makes the work easier if there are debugging possibilities. The Qemu [2] and Bochs [3] emulation tools have proven useful for building a virtual environment.

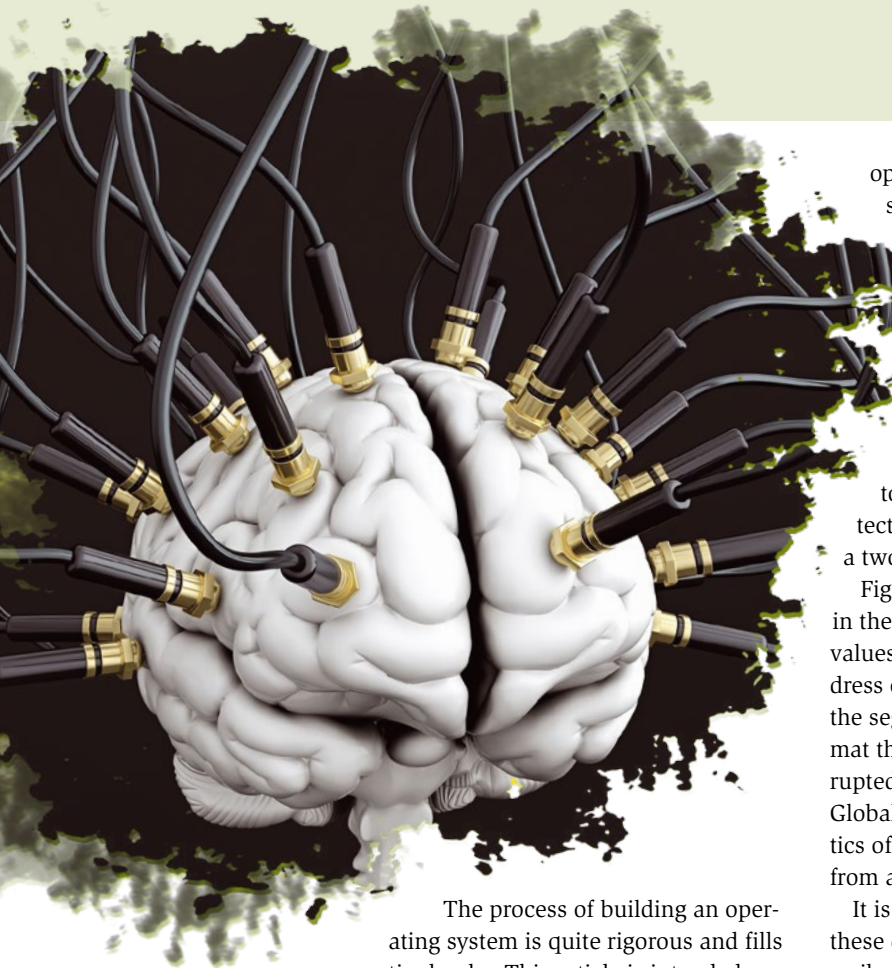
One way to create a bootable disk with as little overhead as possible (and allow it to receive regular updates) is to use a FAT-formatted disk image with the Grub 1 boot manager [4] (Figure 1).

#### Author

**Hans-Georg Eßer** is professor for operating systems at South Westphalia University of Applied Sciences. Prior to his academic career, he worked in magazine publishing, most recently as editor-in-chief of EasyLinux.



**Figure 1:** The easily configurable Grub 1 boot manager doesn’t just boot Linux.



The process of building an operating system is quite rigorous and fills up entire books. This article is intended as a roadmap for taking your first steps. Needless to say: the task of building an operating system is not for beginners. You will need some background in computer science and programming to undertake this kind of a project. But even if you are new to the subject, this article will give you some insights into how developers think about operating systems and the OS programming environment.

If you're ready for the journey, I will introduce some tools that can help you with building your hobby OS. I have also collected online sample files and links to more detailed explanations of the theoretical principles [5].

### Basics

The classic programming language for operating system development is C, not C++ or C#, but the old procedural C. In addition, some parts have to be written in assembler, either in Intel or AT&T syntax [6].

Parallel to the operating system, a collection of test and utility programs must be created to test newly developed OS features; you'll also have to create a shell (at least a rudimentary one). It is *not* a good idea to program a graphical user interface before the most important basic functions around process management and the file system are stable.

"Process management" already implicitly assumes that the operating system will support multitasking – but this is not mandatory; even building an MS-DOS clone can be an informative exercise.

Although you can give free rein to your creativity when designing and implementing your own operating system, there are some places where the devel-

oper has to pedantically implement the descriptions from standards documents. Two examples are storage management and file systems.

### Segments and Pages

In order to implement the modern memory management method of paging [7] on a 32-bit Intel processor, it is mandatory to first activate the older method of segmentation [8]. The Intel documentation describes how to define segments in the Global Descriptor Table (GDT) and then switch the processor into Protected Mode and persuade it to use this table. Only then can a two-level page table be created and activated.

Figure 2 shows the structure of an eight-byte (64-bit) entry in the GDT. In addition to any special options, two important values are stored here: the 32-bit *base address* as the start address of a segment and the 20-bit *limit*, from which the size of the segment is calculated. However, these values are in a format that takes getting used to: *limit* and *base address* are interrupted by other attributes. This has historical reasons: the Global Descriptor Table (GDT) used to define the characteristics of the memory area used in program execution was created from a smaller data structure of older Intel CPUs.

It is crucial that the operating system generates and fills these data structures with content exactly in the form prescribed by Intel. There is no scope for creative deviations or improvements, because the processor only accepts exactly the given format.

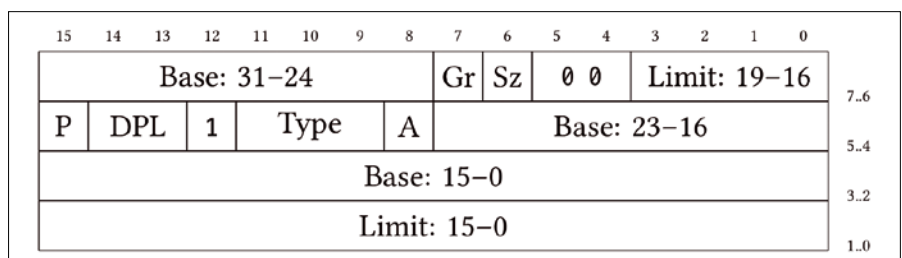
### File System

The developer has more freedom in the conception of a (logical) file system, i.e., in the question as to which blocks on a the storage medium store which administrative and user data and which features the file system offers. For a Unix-like system, at least the features of the Minix file system [9] are required; there should therefore be inodes with Unix access permissions, as well as user and group IDs, directories, symbolic and hard links, and special files (block and character devices, pipes, sockets).

If you are not looking for compatibility with other file systems, there is plenty of scope for new approaches. If, on the other hand, the operating system has to be able to read and write data from another OS, you'll need to pay close attention to the standards.

### Kernel Development

Operating systems form an interface between applications and hardware. Accordingly, it is to be expected that direct hardware



**Figure 2:** Many data structures in the kernel are predefined by the hardware; this figure shows the Intel x86 segment table (GDT).



access often takes place in the kernel. This hardware access is achieved through privileged machine language commands like `in` and `out`. In addition, special processor registers must be filled with contents that determine, for example, in which operating mode the CPU is running.

Kernel binaries do not have the usual program headers (such as Linux ELF or Windows PE files) and cannot be linked to libraries on the fly when loaded. There is no linker at the start.

If you use a debugger, you can check whether newly added code works as desired by setting breakpoints and inspecting register and memory contents in the running system. The CPU changes some memory areas autonomously, i.e., without a memory write instruction in the program. This applies, for example, to entries in the page tables where the processor independently sets access and dirty bits for individual memory pages.

Current Linux distributions are mostly 64-bit, and so linkers, compilers, and other tools run in 64-bit mode by default. In order to build a 32-bit operating system kernel, you therefore need to call the tools with options that switch to 32-bit operation. Listing 1 shows the defaults for `gcc`, `nasm`, and `ld`.

An easy-to-understand and compact introduction to x86 kernel development is provided by “Bran’s Kernel Development Tutorial,” which you can find at [Archive.org](http://Archive.org) [10]. In several sessions, the tutorial guides you to a mini kernel booting in an emulator, which activates the protected mode of the Intel processor, creates a segment table for memory access, contains two interrupt handlers (for the keyboard and timer module), generates the interrupt handler list, and switches on interrupts.

If you want to add further operating system features, you can, for example, work through the documents of the “Operating System Development” course offered by TH Nuremberg [11]. Some of the necessary files, further links, and a preconfigured VM with all development tools are available on the web [5].

## Tool Setup

In addition to the usual tools of a development environment, you also need an assembler, because the start routine

### Listing 1: 32-bit Operation

```
CFLAGS=-O0 -m32
ASMFLAGS=-f eleven
LDFLAGS=-m elf_i386
```

of the operating system is not a usual `main()` function. For testing the kernel, an emulator like

### Listing 2: Installation

```
$ sudo apt install build-essential ...
$ apt install build-essential nasm qemu-system-x86 bochs bochs-x
$ sudo ln -s $(which qemu-system-i386) /usr/bin/qemu
```

### Listing 3: Solving the Bochs 2.6.11 Problem

```
$ sudo apt install libtinfo5
$ wget http://de.archive.ubuntu.com/ubuntu/pool/main/r/readline/\
libreadline7_7.0-3_amd64.deb
$ for pkg in 396601834/bochsbios_2.6.9+dfsg-2_all.deb \
396601839/bochs-x_2.6.9+dfsg-2_amd64.deb \
396601840/bochs_2.6.9+dfsg-2_amd64.deb; \
do wget https://launchpadlibrarian.net/$pkg; done
$ sudo dpkg -i bochs*.deb libreadline*.deb
```

## Problems with Bochs 2.6.11

All attempts to display system tables like the GDT or the IDT with the current Bochs version 2.6.11 under Ubuntu 20.04 caused the emulator and debug windows to freeze. Also, several websites make reference to problems with version 2.6.11. For Ubuntu 20.04, the problem was solved by removing (`apt remove ...`) the three packages `bochs`, `bochs-x`, and `bochsbios`, and reinstalling version 2.6.9. In addition, you have to import `libtinfo5` and `libreadline7` (Listing 3). Instead of typing paths, you will also find the packages via Launchpad [12] or a web search for `bochs 2.6.9 ubuntu`.

Qemu or Bochs is a good choice. I used a computer with Ubuntu 20.04 as a test system, on which the necessary packages were quickly installed with the commands from Listing 2. However, the graphical user interface of the debugger integrated in Bochs did not work properly until I downgraded to an older Bochs version (see the box entitled “Problems with Bochs 2.6.11”).

## Boot Manager

A kernel image is created after compiling, assembling, and linking. The computer has to load and activate this image at power-up. From the BIOS or UEFI, the computer loads a boot sector on the selected medium, which contains further instructions for reloading the kernel.

Linux relied on the LILO (Linux Loader) bootloader in early versions; its successor Grub [4] is more flexible and is also excellent for starting your own kernel. A prepared disk image in FAT format with a Grub installation is available from Christian Steinrücken’s GitHub page [13]. You only have to unzip the `bootgrub.gz` with `Gunzip` and can then mount it or manipulate it with the `Mtools`.

The floppy image in FAT format lets you modify files in the image without mounting. Using the `Mcopy` program from the `Mtools` [14], the kernel file can be updated without root privileges with the following command (including the two colons at the end)

```
$ mcopy -o -i kernel.img MyKernel.bin ::
```

Alternatively, you can use a minix-formatted Grub floppy image, but then you have to make changes to the image with a three step process consisting of `mount`, `cp`, and `umount`.

## Debugging

Troubleshooting an operating system is more difficult than troubleshooting an application running as a process. For the simplest form of debugging, insert informative output into the code that displays memory addresses, variable contents, and other state information on the screen.

In order not to mix the regular screen output with debug information all the time, you can write to a virtual serial or parallel port in the emulator and redirect it to a file (or a terminal window).





For Qemu, the additional `-serial mon:stdio` option ensures that output appears on the serial port in the terminal from which the emulator was started. Bochs lets you redirect serial output to a file, if you add a matching line in the Bochs configuration (`.bochsrc`) (Listing 4). Instead of a file, the target can also be a terminal window via `mode=term`, `dev=/dev/pts/5`, whose device file the developer queries in advance with `tty`.

Listing 5 shows a simple implementation of the `uartputc` function, which sends a single character to the first serial port (COM1): It uses the `0x3f8` port for this purpose. The functions used here, `inportb` and `inportb`, execute the x86 instructions out and in via inline assembler.

Bochs has an integrated text mode debugger, which is made more convenient by a graphical frontend (in the `bochs-x` package). At the bottom (above the status line), there is an input line in which you can set a breakpoint, for example, using `lb 0xc0101496`. When the emulated computer jumps to this address, Bochs interrupts the execution.

By pressing F11, the developer can now execute one machine-language command after another, step by step; Bochs displays the disassembled code in the middle column (Figure 3). You can find the relevant address up front by looking

at the symbol table, which the command from Listing 6 generates from the kernel binary.

The kernel can also be debugged with Qemu. If Qemu is started with the option `-s`, it listens locally on TCP port 1234 and can be remotely controlled with Gdb. Next call the debugger with the kernel binary as a parameter – it is now aware of the symbolic names, i.e., functions and variables – and then establish a connection to the kernel from within

### Listing 4: Redirecting serial output

```
com1: enabled=1, mode=file, dev=serial.out
```

### Listing 5: Debug Output on COM1

```
#define COM1_BASE 0x3f8 // first serial port

void uartputc (char c) {
    while (inportb (COM1_BASE+5) & 0x20) == 0) ;
    outportb (COM1_BASE+0, c);
}

char inportb (short port) {
    char wert;
    asm ("inb %1, %0" : "=a" (rv) : "dN" (port));
    return wert;
}

void outportb (short port, char value) {
    asm ("outb %1, %0" : : "dN" (port), "a" (value));
}
```

## Shop the Shop

[shop.linuxnewmedia.com](http://shop.linuxnewmedia.com)

## Missed an issue?

You're in luck.

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

[shop.linuxnewmedia.com](http://shop.linuxnewmedia.com)



**GET IT NOW!**  
 SAVE TIME ON DELIVERY WITH OUR ALTERNATIVE PDF EDITIONS



### Listing 6: Create Symbol Table

```
$ objdump -M intel -D kernel.bin | grep -e '^[^']*<' | sed -e 's/<///' -e 's/>:/'
```

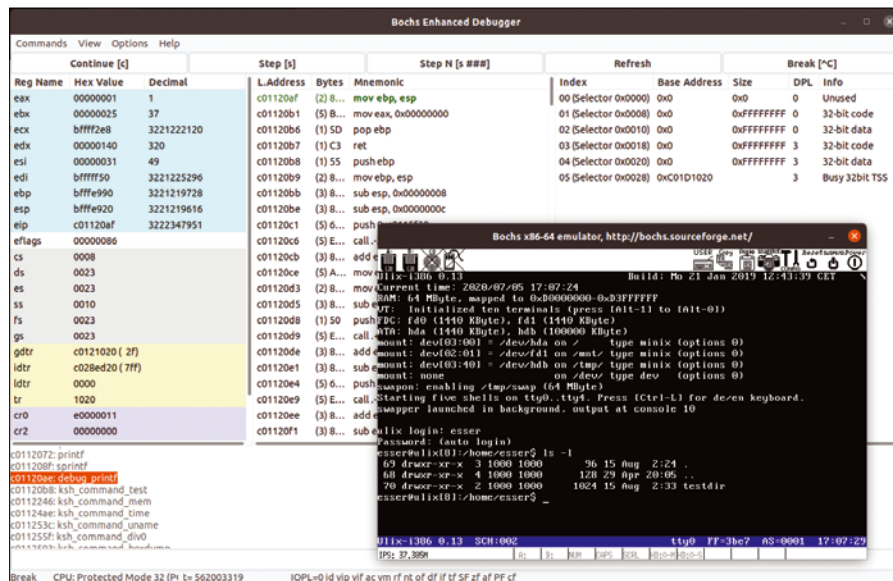


Figure 3: The Bochs GUI displays register contents, disassembled OS code, and selected contents (here the GDT).

Gdb by typing target remote tcp::1234. You can then stop the kernel in Gdb and display the assembler code of individual functions, using disassemble, for example (Figure 4).

### Outlook

By experimenting with the example files for this article [5], you can easily add additional features. If you want to turn this project into a real Unix system with multitasking and some standard shell tools, take a look at Ulix [15]: I developed the Unix-style 32-bit system for use in computer science studies and documented the complete source code in a 700-page book [16]. The book also introduces readers to the theoretical basics of operating systems. An amazingly long overview of other hobby operating systems is available at OSDev.org [17]. If you want to build your own kernel, you're not alone. ■■■

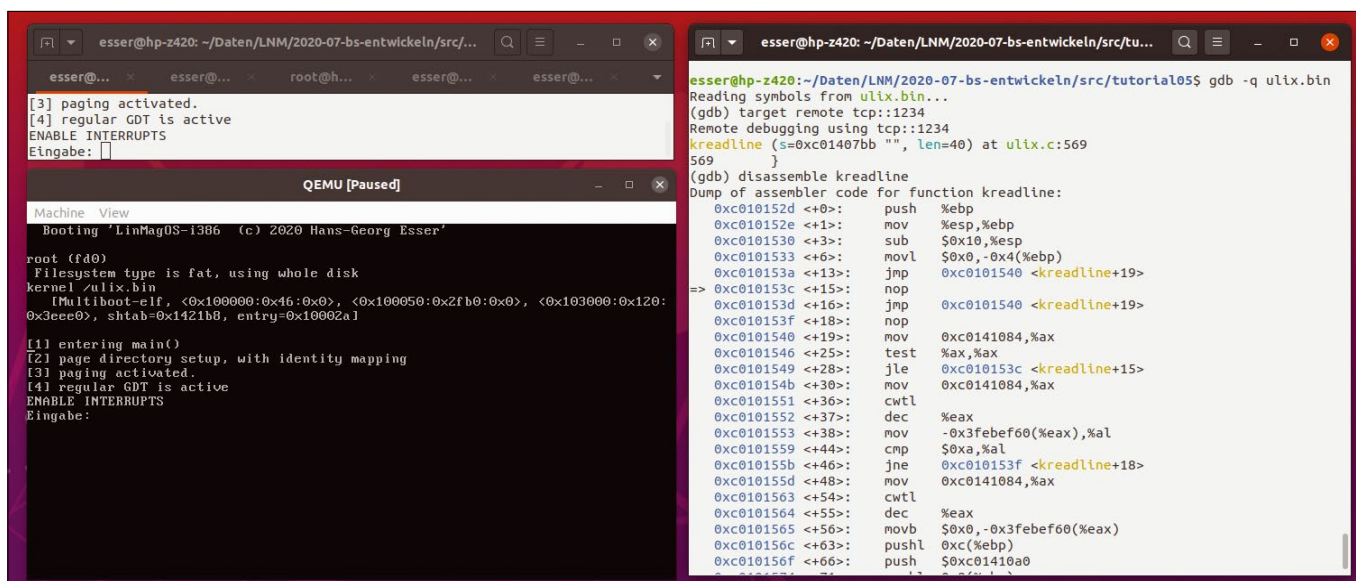


Figure 4: Qemu does not come with its own debugger, but you can use it with Gdb.

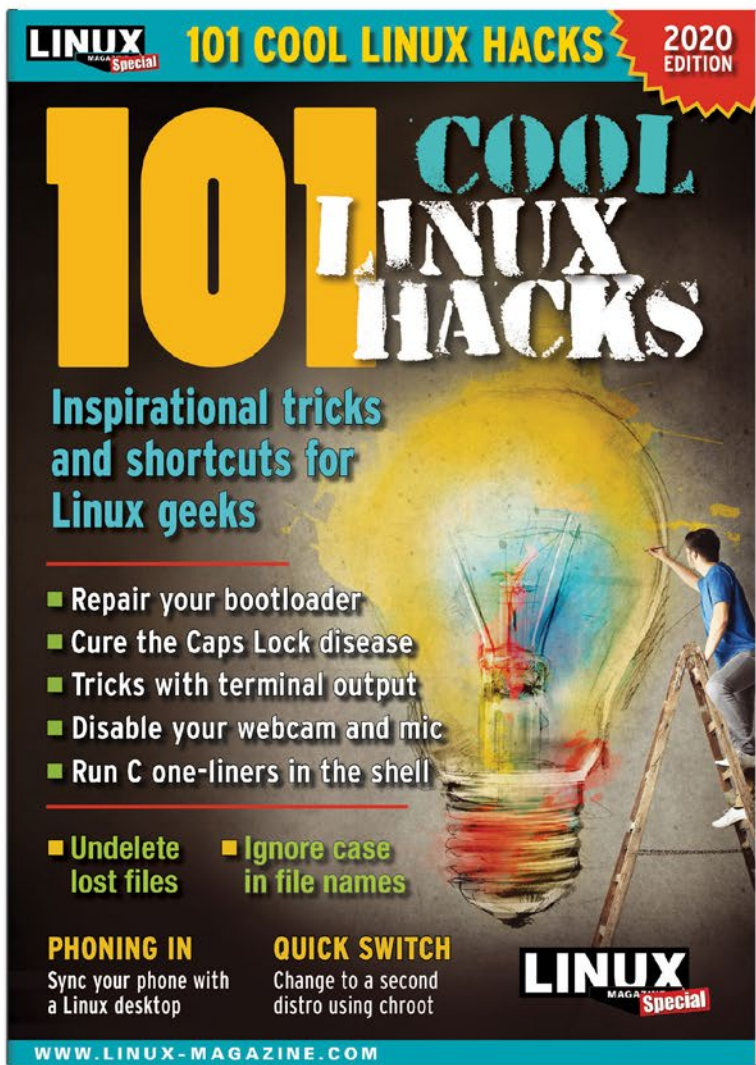
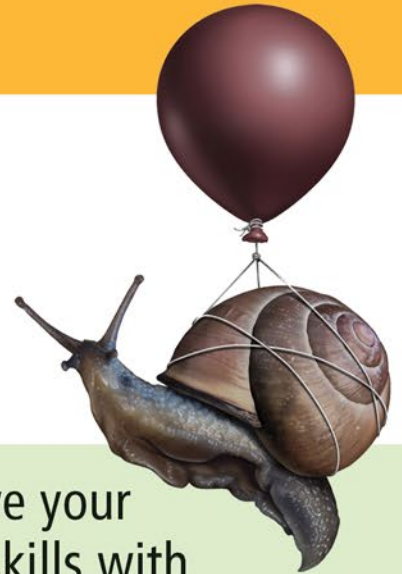
### Info

- [1] Posts by Linus Torvalds: [https://en.wikiquote.org/wiki/Linus\\_Torvalds](https://en.wikiquote.org/wiki/Linus_Torvalds)
- [2] Qemu: <https://www.qemu.org>
- [3] Bochs: <http://bochs.sourceforge.net>
- [4] Grub: <https://www.gnu.org/software/grub/>
- [5] Example files: <http://swf.hgessler.de/d/kernel/en/>
- [6] x86 assembler language: [https://en.wikipedia.org/wiki/X86\\_assembly\\_language](https://en.wikipedia.org/wiki/X86_assembly_language)
- [7] Paging: <https://en.wikipedia.org/wiki/Paging>
- [8] Segmentation: [https://en.wikipedia.org/wiki/Memory\\_segmentation](https://en.wikipedia.org/wiki/Memory_segmentation)
- [9] Minix: <http://www.minix3.org/>
- [10] Kernel Tutorial: <https://web.archive.org/web/20191123051335/http://www.osdever.net/tutorials/view/brans-kernel-development-tutorial>
- [11] Operating System Development, TH Nuremberg: <http://ohm.hgessler.de/be-ws2015/en/>
- [12] Bochs 2.6.9 for Ubuntu: <https://launchpad.net/ubuntu/+source/bochs/2.6.9+dfsg-2/+build/15625627>
- [13] Grub FAT boot disk: <https://q4.github.io/bootgrub.html>
- [14] Mtools: <https://www.gnu.org/software/mtools/>
- [15] Ulix: <http://ulixos.org>
- [16] "The Design and Implementation of the ULIX Operating System": <http://ulixos.org/doc/ulix-book-0.13.pdf>
- [17] OSDev.org wiki: <https://wiki.osdev.org/Projects>



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

## Tuxedo InfinityBook S 14 v5

# Slimline

Are you in the market for a slim, light, and power-packed laptop? We test an upgradeable machine from Tuxedo. *By Mike Saunders*

Some Windows and macOS fans joke that Linux users have no need for svelte, lightweight, and powerful laptops. After all, we like taking things apart and poking around inside, right? A typical Linux laptop should be a 3kg beast with replaceable everything, two parallel ports (ever tried PLIP networking?), and five hard drive bays to store all of our favorite distros. And it doesn't need a fast chip, because we can just run Puppy Linux on it.

Except, of course, that is far from the truth. Sure, we Linux users like to get the most out of our hardware and value upgradeability and longevity over shaving off every single millimeter where possible. But we shouldn't have to compromise. Why can't we have a slim, powerful, and upgradeable laptop that also runs the very latest Linux distros?

This is what Tuxedo aims to offer with the InfinityBook S 14 v5 (Figure 1). Tuxedo Computers GmbH, based in

Augsburg, Germany, has been in the Linux hardware business for over 15 years. This particular model – the InfinityBook S 14 v5 – is “made in China, Assembled in Germany,” and the company describes it as lightweight, slim, elegant and very persistent. It's available in various configurations; the one we tested for this review costs EUR1,329 (see Table 1).

## Design, Dimensions, and Ports

The first thing you notice with the InfinityBook is how light it is. For a machine with a 14-inch display, you expect a bit more heft, but at 1.1kg (2.4lb) it feels fantastically light and portable. Indeed, it's so light that when it's flat on a table, you can't fully open the screen by solely lifting it with a finger – you need to prise the machine apart.

Still, the hinge feels sturdy, as does the magnesium chassis, with virtually

no flexing on the base. The screen permits a bit more bending, but it is still solid. Overall, the InfinityBook's build quality and finish is excellent – and there are no annoying stickers on the palmrests to remove.

Size-wise, the laptop is 322x217x16.5mm. When placed on a flat surface, taking into account the rubber feet on the underside, it rises 2cm above the surface at its thickest point. There are thinner ultrabooks out there, but the InfinityBook is still decently slim.

The left and right sides of the laptop house various ports: a USB 3.1 Gen 1 (type A), a USB 3.1 Gen 2 (type A), a USB 3.1 Gen 2 (type C), an HDMI 1.4b with HDCP (max. 2560x1600@60Hz; 3840x2160@30Hz), a 2-in-1 audio port (headphones/microphone), a microSD card reader, and a Kensington lock port. There's a physical power switch on the right.

But what about the aforementioned maintainability and upgradeability? The InfinityBook does pretty well here: Remove 12 regular crosshead screws from the panel on the underside of the screen, and you get access to the replaceable Samsung RAM, replaceable Samsung 970 EVO Plus SSD, and the replaceable (albeit with soldered cables) Intel AX200 WiFi card (Figure 2). With the RAM being expandable up to 40GB, the machine should be usable for many years down the line.

## Screen, Keyboard, and Trackpad

Let's turn to the usability aspects now. The 14-inch 1920x1080 matte/non-glare screen provides a luminance of



**Figure 1:** Weighing just 1.1kg and measuring 1.65cm deep, the InfinityBook is especially light and slim.

**Table 1:** Specs at a Glance

CPU	Intel i7-10510U @ 1.8GHz
GPU	Intel UHD Graphics 620
RAM	8GB DDR4 2666MHz
Screen	14" 1920x1080 matte
Storage	500GB Samsung 970 EVO Plus SSD
Networking	Intel AX200, WiFi 6 and Bluetooth 5.1
Ports	3x USB, HDMI, MicroSD, head/microphone, Kensington
Size	322x217x16.5mm
Weight	1.1kg (2.4lb)
Price	EUR1,329





**Figure 2:** Pop off the back by removing a few screws, and you can swap out the RAM, SSD, and network card.

350 nits and a contrast ratio of 1200:1. At its very brightest, it still looks a bit dim compared to a 500-nits display on other high-end machines (such as a MacBook Pro that we used as a comparison). But for most use cases, this isn't an issue.

Meanwhile, the InfinityBook's chiclet keyboard has decent travel and feels solid and satisfying to type on, without generating much noise. We'd prefer a bit less space between the individual keys. On our review unit, we noticed a glitch: duplicated pipe-and-backslash keys, one on the right (next to the Enter key) and one beside the left Shift key. The whole keyboard is supported by two levels of back lighting (enabled using Fn + Space). And there's a special Tux key – not just a sticker over a Windows logo!

Underneath lies a spacious and responsive 10.4x6.4cm trackpad, in “diving board” style. In other words, it's eas-

### The Software Side

The InfinityBook ships with TUXEDO OS, which is basically Ubuntu 20.04 with the Budgie Desktop and a few other tweaks. It's also possible to order the machine with openSUSE.

By and large, Ubuntu runs well on the laptop, but we encountered a few annoying gremlins. For instance, when suspending and resuming, the desktop flashes briefly (and brightly) before the lock screen appears, which is rather off-putting during regular use. It doesn't

only disappointment with the machine, which is otherwise superb. In a busy office or home, it's not a huge deal. However, in quiet environments, the trackpad noises can be quite distracting.

Stereo speakers are mounted on the underside, towards the front. They generate an adequate amount of volume at their max, but are lacking in bass. You wouldn't want to use it to watch concert streams, but, for casual video watching or gaming, it's good enough. Above the screen is a 1.0 megapixel webcam (attached to a white LED that shines when it's enabled), accompanied by an IR camera.

### Performance and Battery

As mentioned earlier, the laptop is available in several configurations – our review unit shipped with a quad-core Intel Core i7-10510U chip from the “Comet Lake” family, running at 1.8GHz, with 8MB cache. This is

have a massive impact on usability, but, for a high-end machine, we'd hope for a slicker overall experience.

Also, the keyboard backlight level isn't maintained after suspending and resuming. In one instance, we had the display settings opened and closed the lid. Upon reopening, the CPU spiked, ramping up the fans to jet engine levels, and we had to force quit the Settings app. Otherwise, Ubuntu runs well on the machine.

ier to click on the bottom parts of the pad than on the top. Clicking is rather loud, however. Even if you opt for the light tap approach, just patting your fingertip on the pad rather than depressing to generate a mechanical click, it produces a slight vibration and some noise – especially at the bottom of the pad. In terms of hardware, this was our

backed by 8GB of DDR4 2666MHz RAM. Communication is provided by an Intel AX200 chip, with dual-band WiFi 6 and Bluetooth 5.1.

In our test, we could watch 1080p 60fps YouTube videos without the InfinityBook breaking a sweat – the fan didn't ramp up. With some other high loads, the fan noise was noticeable, but not annoying. We found that the fan changed speeds quite often though, adapting to the loads, which always caught our attention. A smoother transition between fan speeds would be preferable.

Airflow is handled entirely through the hinge area, which is a setup we really like; you can use it on a bed or sofa without worrying about blocking vents on the underside and triggering heating problems. The laptop gets moderately hot under a high load, but never unpleasant to the touch.

In terms of battery, the InfinityBook ships with an integrated 73Wh polymer smart lithium-ion battery. Tuxedo claims a whopping 24 hours of battery life, albeit when the machine is left idle with minimum screen brightness – and we can confirm that. To recreate a more real-world usage scenario, however, we tested playing 720p YouTube videos with medium brightness and achieved almost 14 hours. When suspended, the laptop loses around one percent of battery every hour.

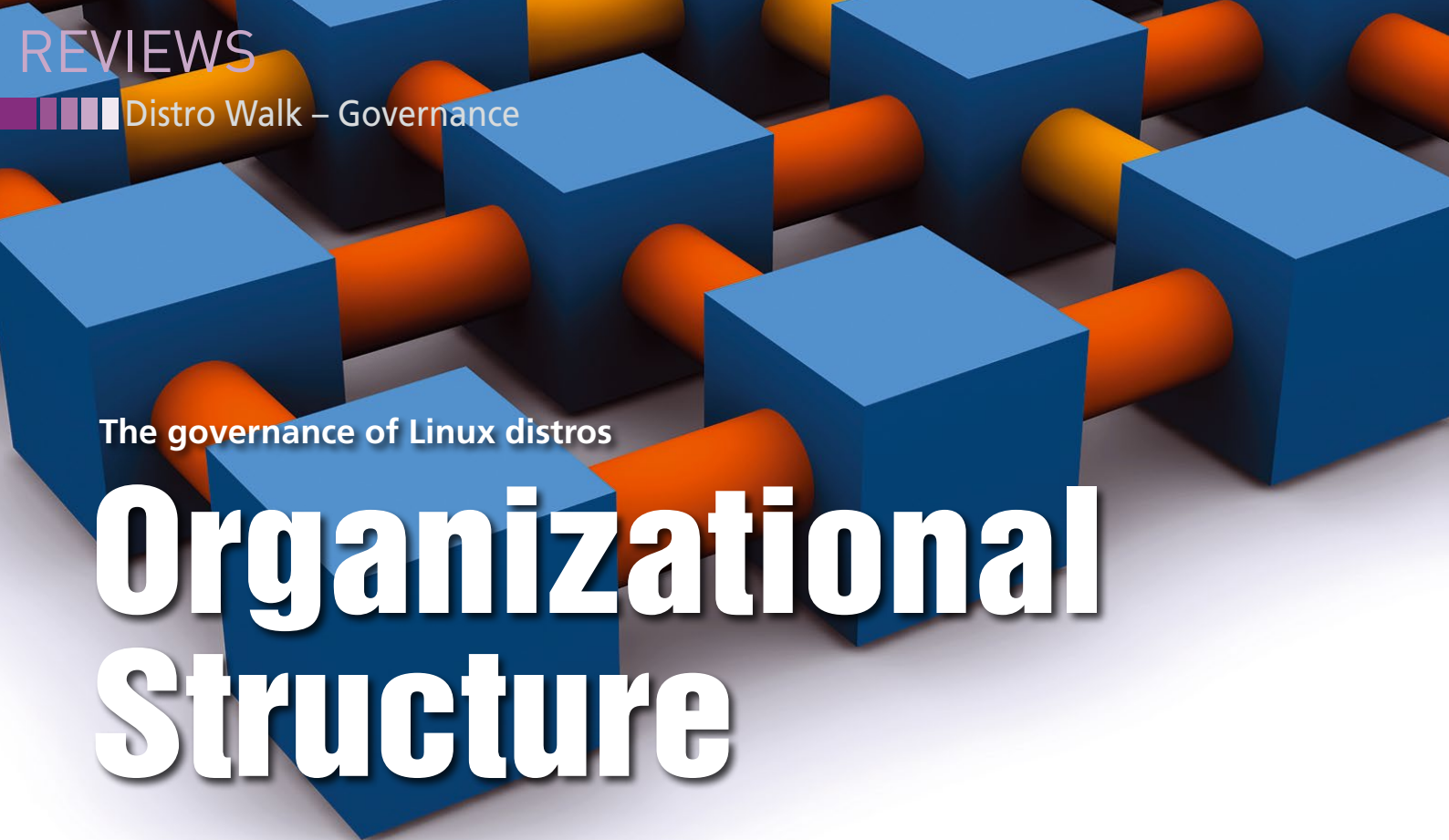
The laptop's charger is a slim 10x5x1.5cm mini-brick, which brings the battery back up to 100 percent in 2 hours and 15 minutes. The battery gets quite hot during charging, though.

### Conclusion

With its excellent build quality, slim and lightweight chassis, beefy battery, good keyboard and screen, and possibility to update the RAM and SSD in the future, the InfinityBook S 14 v5 is a great machine for the price. It loses a couple of points due to the noisy trackpad and a few software glitches (see “The Software Side” box), but in our two weeks of testing, we were largely very happy with it. ■■■

### Author

**Mike Saunders** has been using, advocating, and writing about Linux since 1999. He was one of the founders of *Linux Voice* and gets nostalgic thinking about it.



The governance of Linux distros

# Organizational Structure

Whether you are a user or a developer, knowing how a distribution governs itself can help you choose a Linux distro. *By Bruce Byfield*

**H**ow Linux distributions govern themselves may be the last aspect you look at when choosing a distribution. Often information about governance is buried several levels down on a project’s website, yet the information is worth uncovering. Even though open source is usually considered as a business advantage, idealism still runs strong in the community, and it is often reflected most clearly in organizational structure. Administratively, Linux distributions run the spectrum from town-hall meetings on online forums and chats to dictatorships to progressive democracies. If you are a user, governance may be a clue to whether a distribution suits your preferences. If you are a developer, governance can become even more important. If you become a contributor, you will be dealing with the organization on a daily basis.

This month, I’ll look at the governance of seven popular distros. In this sampling, you most likely will be able to find an approach to Linux that suits you.

## Arch Linux

Although Arch Linux [1] is one of the more influential distributions, it is organized to operate with a minimum of structure. Its structure has two basic te-

nets: First, “Anyone should be free to contribute to any aspect of the distribution;” second, “Decisions concerning a particular project should be made by people actively involved in that project.” Conflicts are resolved through discussion until consensus is reached.

If consensus cannot be reached, the Project Leader steps in to make the final decision. The Project Leader also represents Arch in legal and publicity matters. For 13 years, the Project Leader was Aaron Griffin (aka “phraktur”), but in 2020 Griffin stepped down and was replaced in an election by Levente Polyak (aka “anthraxx”). At the same time, a structural change limited the Project Leader’s term to two years and also defined the role as whatever “is determined by a vote among eligible members of the Arch Linux Team.”

Other official contributors sometimes overlap, but this group includes Developers, Trusted Users (managers of the community repositories), and Support Staff (managers of forums, wikis, IRC channels, and bug ports.) People in all three categories are eligible to vote for Project Leader, but Developers may veto the results within 14 days after they are announced, in which case the candidate who finished second becomes Project

Leader. Despite this new structure, Arch remains a simply governed distro, with similar principles to Debian, but with a less formal structure.

## Bodhi Linux

Bodhi [2] is the sort of small distro many people still imagine when they think of Linux. Currently, the project is led by Robert Wiley. “In a way,” Wiley says in a private email, “[I] have the freedom to make what changes and choices I want but I would be reluctant to do so without full consensus and support from all active team members.” The team currently consists of one other developer and another four who handle system administration, web development, and the forums. In addition, there are several regular forum members who sometimes contribute suggestions.

With such a small group, there is “hardly any need for formal governance or commit reviews. If we need to make a decision that is important enough we will talk it out via email or on Bodhi’s Discord channel until we have full consensus,” Wiley says. “I have often wondered what we would do if our community grew to such an extent we had to establish some formal rules for things like commits and decisions. I suppose that is

Lead Image © Peter Galbraith, Fotolia.com



something we will deal with if and when it ever becomes an issue.”

### Debian GNU/Linux

With over 1,300 maintainers and many more uncounted contributors, Debian [3] is probably the distribution that looks most like a corporation – although an unusual one in many aspects. It seems no accident that several Debian Leaders have gone on to become executives in major tech corporations.

Debian’s Project Leader, who is elected each April as the official representative of the project, helps to coordinate development internally. However, the Devuan fork of Debian – and many others – argue that real authority in the project lies in the unelected Technical Committee and the FTP Masters, who are responsible for what is distributed. The Technical Committee, for example, is responsible for the adoption of systemd, while the FTP Masters can exercise considerable authority over when releases are made and such issues as which architectures are supported. In

addition, numerous other teams oversee such tasks as publicity and the processing of new members.

Official Debian Maintainers and Debian Developers can vote on the Project Leader, as well as general resolutions about the direction of the project, which occur on an average of every 15 months or so. All elections use the Condorcet method [4] of tallying votes, in which each possible choice is compared against every other one. All this structure is detailed at length online and conforms to the principles of the Debian Social Contract [5], which prioritizes free software. The Debian repositories include sections for software that depends on proprietary software (contrib) and for proprietary software (non-free), but these are not enabled by default.

### Fedora

Although the Fedora project [6] is the open source basis for Red Hat Enterprise Linux and CentOS, it is also popular in its own right. Its membership is unstated, but rivals Debian’s in size. Unsurpris-

ingly, it has a complexity of structure that rivals that of both Debian and Ubuntu.

Fedora is managed by the Fedora Council, which consists of four members appointed by Red Hat: one to represent the Engineering team, one to represent the Mindshare (marketing and communications) team, and two community-elected representatives. In addition, the Fedora Project Leader is appointed to chair the council. The Fedora Council may vote on decisions in the hopes of reaching consensus, but its charter also mentions “‘lazy approval’, in which general consent is assumed unless valid objections are raised within a period of time – generally three to seven days, although the timeframe should be stated each time and should be proportionate to the impact of the action. This process is used for decisions with short-term consequences and which can be easily reversed. Any project member can ask for the deadline to be extended or the decision escalated to require full consensus.” Numerous specialized projects are organized under the Fedora Council and report to it.

Shop the Shop → shop.linuxnewmedia.com

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

Want to subscribe?

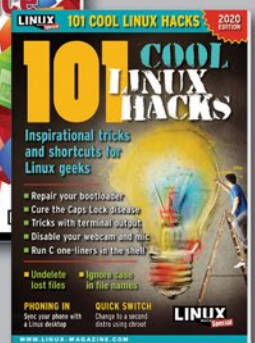
Searching for that back issue you really wish you’d picked up at the newsstand?

➤ shop.linuxnewmedia.com

DIGITAL & PRINT SUBSCRIPTIONS



SPECIAL EDITIONS



On the one hand, the composition of the Fedora Council means that Fedora is not officially a community-based distribution. On the other hand, the specialized projects appear to act independently. Even more importantly, of all the major distributions, Fedora appears to be the least technocratic, being careful to give non-programmers a role and a voice.

## Linux Mint

When Linux Mint [7] started in 2006, development decisions were made by founder Clement Lefebvre and a handful of active developers. As Lefebvre told me in 2013, “strong leadership is important and benefits Linux Mint, [because] the decisions we take remain consistent and are coherent with our overall vision.” At the same time, Lefebvre added that “We always try to gather as much feedback as possible, and to get a good appreciation of what people want.” In fact, Linux Mint has a reputation for listening to user feedback far more than many distributions; it first became popular by forking Gnome 2 after Gnome ceased to develop it. Soon after, Mint started the development of Cinnamon, a desktop environment whose development continues to be heavily influenced by user feedback. The Linux Mint blog continues to be a major means of communication with the distro’s user base, as well as the forums.

Today, Mint is more structured than it once was. In addition to Lefebvre, Mint has an Administration and Moderation team, as well as teams for development, design, translation, and quality assurance. Its philosophy remains unchanged. The project website’s FAQ explains: “Linux Mint does not support any political or ideological stance against any software programs or editors no matter what license they use. With that said, most if not all (depending on the edition) software used in Linux Mint is Free and Open Source. We believe in Open Source as a choice, not as a constraint.”

## openSUSE

Just as Fedora is a community project associated with Red Hat, so openSUSE [8] is a community project associated with SUSE. However, while Red Hat appointees dominate the Fedora Council, the five members of the openSUSE Board are elected by the community. Historically,

the openSUSE Board tends to consist of a majority of Germans and other Europeans. Far from being dominated by SUSE, the openSUSE Board rarely hears from SUSE about matters of policy, aside from the appointment of the board chair. In fact, in early 2020, the openSUSE Board was discussing ways to interact more closely with SUSE.

The openSUSE Board then appoints people as tasks arise. Currently the only appointed board position is the treasurer, who oversees the Travel Support Program and interacts with SUSE and the project’s other sponsors on all financial matters. Appointees remain in their position until the next board election. Although openSUSE has ongoing teams like other large distros, its bureaucratic structure appears lighter than most.

Another similarity to Fedora is openSUSE’s insistence on using only free software in its releases. If anything, SUSE is even more specific about this policy, with the openSUSE Board wiki devoting several hundred words to the topic before describing the governance of the project.

## Ubuntu

Ubuntu’s governance page [9] declares, “This is not a democracy, it’s a meritocracy. We try to operate more on consensus than on votes, seeking agreement from the people who will have to do the work.” Mark Shuttleworth, Ubuntu’s founder, is the “self-appointed benevolent dictator for life (SABDFL)” and reserves the right to make the final decision on any issues that may arise. Although the governance page describes this situation as “happily undemocratic,” in practice this organizational structure has caused problems once or twice in the past, particularly during the development of Ubuntu’s Unity desktop, when decisions by Shuttleworth or his employees at Canonical have over-ruled community volunteers. However, in the last few years, such conflicts have become rarer, as Canonical has turned its attention to servers and away from the operating system.

In ordinary affairs, Ubuntu is governed by the Community Council and the Technical Board. Both groups consist of both community representatives and Mark Shuttleworth. Shuttleworth also appoints members of the Community Council and nominates members for the Technical

Board to be accepted by Ubuntu developers. Like the rest of Ubuntu, these groups are governed by the Code of Conduct, which outlines how project members should interact with one another. I have heard Ubuntu described as governed by discussion and consensus in everyday matters and governed by Shuttleworth and his appointees on project direction and strategy.

## A Distro for Everyone

Judging distributions by their governance is not always easy. A distribution with a democratic structure can still be dominated by veteran contributors. Conversely, a distribution with no structure beyond consensus might give a newcomer a larger say in decisions than one with regular offices. And in any distribution, an eager volunteer may quickly find a place, no matter what the structure.

Whatever your reason for distro shopping, remember the basic tenet of corporate structure: The formal structure is often compensated for with an informal structure. The same is true for distributions. Hang out in the forums and IRC channels for a while, and you should soon see how a distribution is actually run – and who runs it, officially or unofficially. How a distribution runs on a daily basis could be just as important to you as a user or contributor as its technical aspects. ■■■

## Info

- [1] Arch Linux governance: [https://wiki.archlinux.org/index.php/DeveloperWiki:Governance\\_And\\_Decision\\_Making](https://wiki.archlinux.org/index.php/DeveloperWiki:Governance_And_Decision_Making)
- [2] Bodhi Linux: <https://www.bodhilinux.com/>
- [3] Debian organization: <https://www.debian.org/intro/organization>
- [4] Condorcet voting method: [https://en.wikipedia.org/wiki/Condorcet\\_method](https://en.wikipedia.org/wiki/Condorcet_method)
- [5] Debian Social Contract: [https://www.debian.org/social\\_contract](https://www.debian.org/social_contract)
- [6] Fedora leadership: <https://docs.fedoraproject.org/en-US/council/>
- [7] Linux Mint teams: <https://linuxmint.com/teams.php>
- [8] openSUSE guiding principles: [https://en.opensuse.org/openSUSE:Guiding\\_principles](https://en.opensuse.org/openSUSE:Guiding_principles)
- [9] Ubuntu governance: <https://ubuntu.com/community/governance>

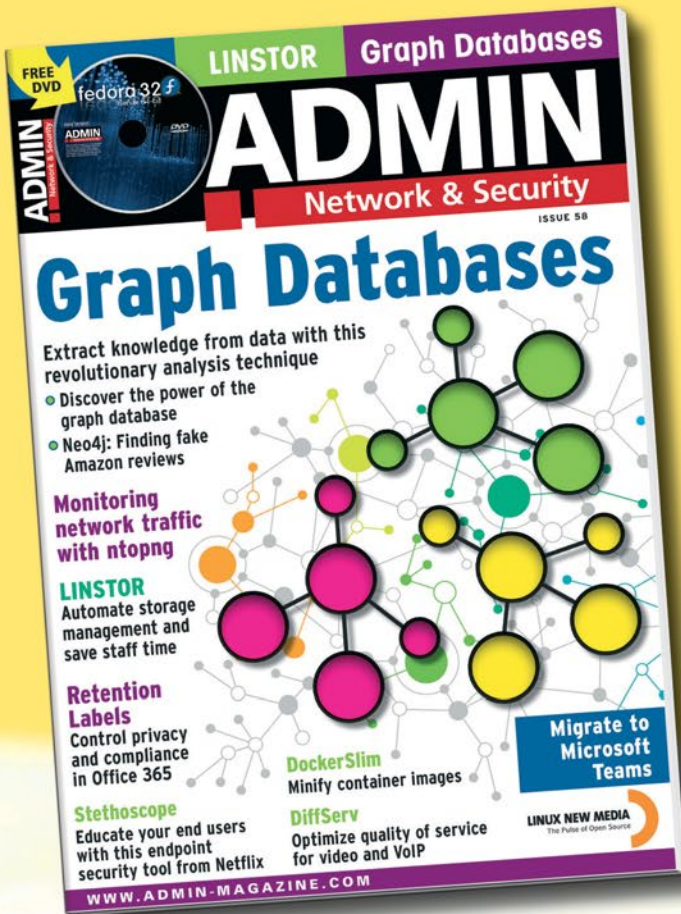


# REAL SOLUTIONS *for* REAL NETWORKS

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

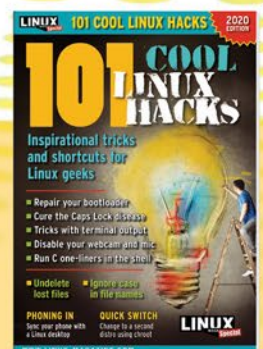
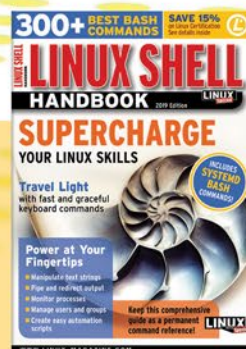
Improve your admin skills with practical articles on:

- Security
- Cloud computing
- DevOps
- HPC
- Storage and more!



**SUBSCRIBE NOW!**  
[shop.linuxnewmedia.com](http://shop.linuxnewmedia.com)

Check out our full catalog:  
[shop.linuxnewmedia.com](http://shop.linuxnewmedia.com)





A web-based server management tool

# Remote Control

Ease the process of managing Linux installations remotely with the web-based Ajenti control panel. *By Mayank Sharma*

**A**genti [1], an open source, web-based system control panel, lets you manage several important server management tasks with very little overhead. Think of it as a lightweight version of Webmin [2], the popular web-based administration tool.

You can use Ajenti for basic server monitoring, installing and removing packages, as well as managing services, the network, and more. It'll also help you remotely manage files on your server without firing up a terminal. In fact, if you want to get to the console, Ajenti will also give you a command-line interface inside the web browser itself.

The two Ajenti branches currently under development are Ajenti 2 and Ajenti V. Ajenti 2 offers a new lightweight interface developed with Python 3 and AngularJS. Ajenti V is a plugin that adds

web hosting features to Ajenti, but currently only works on top of the previous version, Ajenti 1 (see the “Ajenti V Plugin” box).

## Installing Ajenti 2

Ajenti 2, the latest version of the control panel, officially supports the latest releases of deb-based distributions like Debian and Ubuntu, as well as RPM-based distros including CentOS, Fedora, and RHEL. The process for both doesn't differ much, but in this example I'll assume the IP address of the Linux server is 192.168.0.1.

On deb-based installations like Ubuntu, first enable the Universe repository:

```
$ sudo add-apt-repository universe
```

And then install the dependencies with:

```
$ sudo apt-get install build-essential \
python3-pip python3-dev python3-xml \
libssl-dev python3-dbus python3-augeas \
python3-apt ntpdate
```

Similarly, in RPM-based distributions like Fedora and CentOS, not all the required dependencies are in the main repository, so you will need to install the EPEL repository:

```
$ sudo dnf install epel-release
```

Then install the dependencies with:

```
$ sudo dnf install \
-y gcc python3-devel python3-pip \
python3-pillow python3-augeas \
python3-dbus chrony openssl-devel
```

With the dependencies in place, irrespective of your underlying distribution, you can now fetch and install Ajenti using the following script:

```
$ curl \
https://raw.githubusercontent.com/ajenti/ajenti/master/scripts/install.sh | sudo bash -s -
```



## Ajenti V Plugin

Ajenti V, an optional plugin for Ajenti 1, lets you manage multiple websites using the same control panel and carry out some additional website-related tasks, as well as configure email accounts.

Ajenti V can only be installed atop Ajenti 1, so you'll first have to install Ajenti 1 on your server before installing the Ajenti V plugin. Of note, Ajenti 1 depends on certain packages that have been replaced on the latest versions of the supported distributions. While you can work around the limitation by manually installing the dependencies from third-party repositories, it's best to install a still-supported, long-term support (LTS) release from a supported distribution. For instance, I was able to successfully install Ajenti 1 without any issues on the latest Ubuntu 16.04.7 LTS release that reaches end of life in 2024.

To set up Ajenti 1, follow the installation instructions for Ubuntu [3] on Ajenti's website. Once that's done, you can install Ajenti V. Since the plugin requires PHP v5.6 support, you'll have to enable Ondřej Surý's PHP PPA, with:

```
$ sudo apt-get install \
-y software-properties-common
$ sudo add-apt-repository ppa:ondrej/php
$ sudo apt-get update
```

When that's done, you can again follow the instructions on Ajenti's website and install the basic Linux, NGINX, MySQL, PHP (LNMP) stack with:

```
$ sudo apt install ajenti-v \
ajenti-v-nginx \
ajenti-v-mysql \
ajenti-v-php7.0-fpm php7.0-mysql
```

Once that's done, restart Ajenti with:

```
$ sudo systemctl restart ajenti.service
```

Now fire up a web browser and bring up the dashboard on port 8000. Ajenti 1's dashboard offers more options than Ajenti 2's dashboard, including the *Websites* option in the navigation panel. You can now refer to Ajenti's official documentation to configure PHP and set up a website using Ajenti V [4].

The script will pull packages and a handful of the commonly used Ajenti plugins. There are a few official and community-supported plugins that Ajenti doesn't install by default, but these can be pulled in from inside the administration panel.

Since Ajenti is written in Python, you can install it on other distros in addition to the officially supported ones. The website documents the process of manu-

ally installing Ajenti 2 [5] using Python's pip package management system.

Once the installation is complete, you can control the Ajenti service with:

```
$ sudo systemctl restart ajenti.service
```

## Dashboard and Tools

Fire up a web browser and head to <https://192.168.0.1:8000> from another machine on the same local network. You'll have to accept the self-signed certificate before you can continue to the login page.

In Ajenti 2, you can log in using the base installation's credentials. Since most of the administrative tasks require root user authentication, it's a

good idea to log in with the root user's credentials.

Alternatively, you can also log in as a standard user and then elevate yourself to a superuser from inside the console. Before you do that however, you must tweak Ajenti's configuration file (see the "Becoming a Superuser" box).

Ajenti's landing page, the Dashboard (Figure 1), displays the hostname, CPU usage, memory usage, load average, and the computer's uptime. These details are fetched via widgets, and you have the option of adding more widgets.

Click the *Add Widget* button in the right corner to reveal a pull-down list of all the supported widgets. Scroll through the list and click the widget you wish to

## Becoming a Superuser

Before you can elevate your standard Linux user to the root user, you'll have to add details about this user to Ajenti's configuration file.

Fire up a terminal and edit Ajenti's simple configuration file with:

```
$ sudo nano /etc/ajenti/config.yml
```

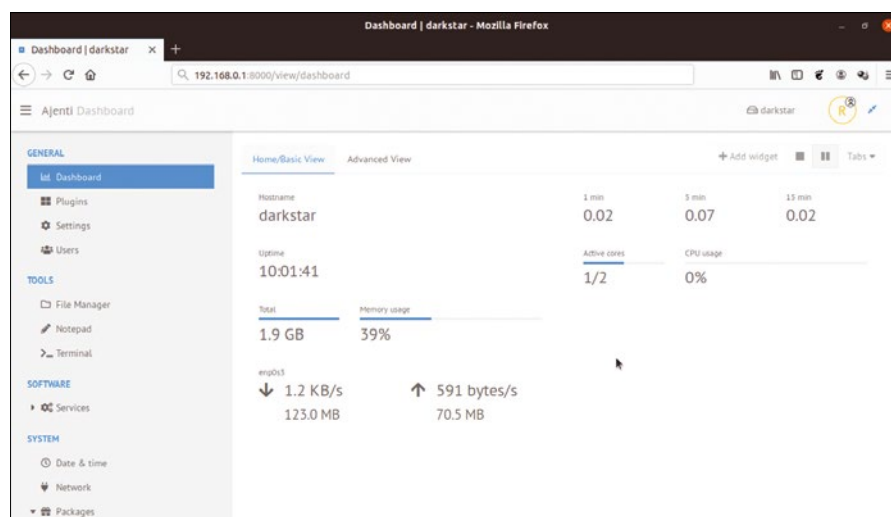
Now scroll down to the end of the file and enter the following line:

```
restricted_user: bodhi
```

Make sure you replace *bodhi* with the username of your installation's standard user. Now save the file and bring the changes online by restarting the service with:

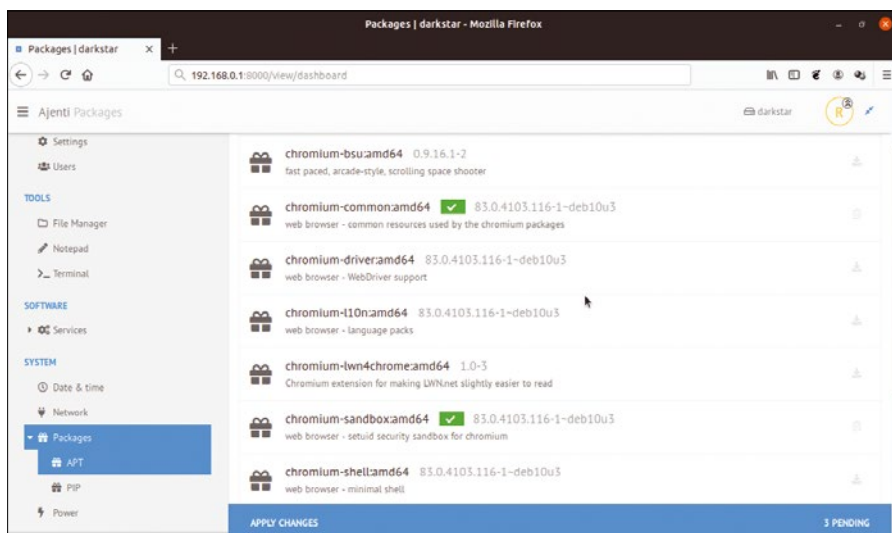
```
$ sudo systemctl restart \
ajenti.service
```

You can now log in as the standard user. Whenever you run into a section that requires superuser permissions, just click on your username in the top-right corner and select *Elevate*. Enter the password for the user, and Ajenti will give you superuser privileges.

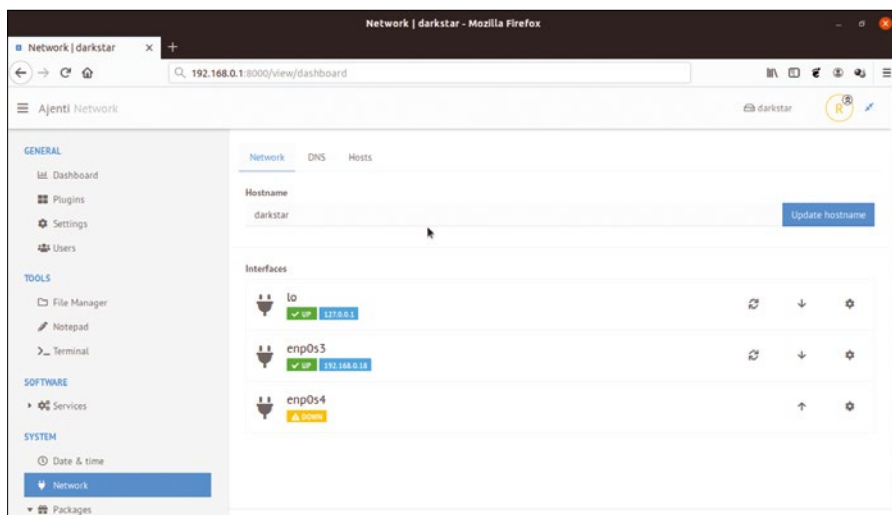


**Figure 1:** By default, the Ajenti session times out after 60 minutes. You can increase or decrease this time by heading to *General | Settings*.

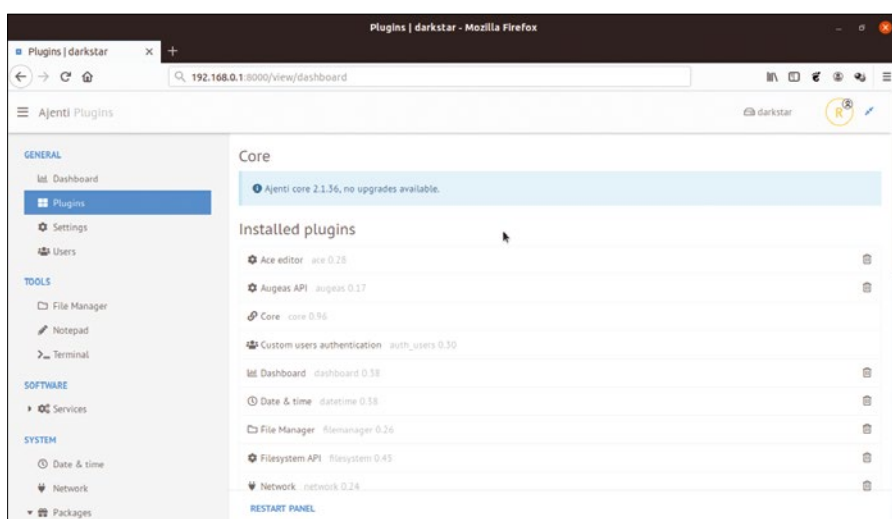




**Figure 2:** You can also search for installed packages and mark them for removal; Ajenti will remove them using the underlying package management system.



**Figure 3:** You can also customize and configure the `/etc/hosts` file from the Network interface.



**Figure 4:** Ajenti 2 has some interesting plugins that aren't enabled by default. Make sure you scroll down to the Available Plugins section and check them out.

add; it'll be placed in the first available space on the Dashboard.

To customize the Dashboard, mouse over any widget to reveal buttons to either delete or rearrange the widget. You can even add a completely new tab on the dashboard and place widgets there.

One way you could use this arrangement, for instance, is to separate all the available system information into a standard view (that lists the default widgets) and an advanced view (that lists all the available widgets).

Ajenti's interface is fairly intuitive and self-explanatory. The Tools section lists the utilities available to help administer your installation.

File Manager lets you edit files in your installation. When you click on a file, Ajenti will display its metadata including the size and ownership details. You also have the ability to change the file's permissions from here using a point-and-click graphical interface, which is helpful for new Linux users who aren't yet familiar with Unix's numeric notations. You can do the same for directories as well.

File Manager also lets you create new files and directories. You can also upload content into a directory. This is especially useful if you wish to upload content into the Ajenti-managed machine from a remote machine on your network.

If the file you are interested in is a text file, Ajenti will offer to open it in the built-in text editor called Notepad, which lacks rich-text editing tools and is designed for editing configuration files. You can also use it to create a new file and save it anywhere in the File Manager.

Finally, the Terminal tool offers a web-based console that'll help you get a command-line interface into your Ajenti-equipped machine. This is especially useful when accessed from a remote computer as it saves you the effort required to set up and configure an SSH server. If you only need to run a single command, you can do this from the *Terminal* tab without launching a full-fledged terminal.

## Administrative Tasks

To manage the services running on the machine, head to *Software | Services*.

This brings up a list of installed services in alphabetical order along with their

current status. Running services are represented by a *Play* button, while stopped services have a *Pause* icon.

To control a service, mouse over it to reveal icons to reload and stop/start the service. This is useful for users who are new to Linux and aren't well-versed in systemd operations.

To remotely install packages on the Ajenti-equipped server, go to *System | Packages* (Figure 2). Ajenti 2 has two package installation options: Apt/RPM or pip. Both of them have the option to refresh the package list to sync with the online repositories.

Once the repositories have been refreshed, you can use the search bar to search for apps you wish to install. Then pick a package from the displayed list and mouse over it to reveal the option to download and install the package.

When you click on the install icon, Ajenti will fire up the terminal and download and install the app using the underlying package management system of the machine running Ajenti.

Besides packages, the System section also helps with a couple of other admin-

istrative tasks. Navigate to *System | Network* (Figure 3) to view all the network interfaces connected to the machine along with their IP addresses. You can, just as with Services, restart and deactivate any of the interfaces.

More importantly, you can also click on the gear icon adjacent to a network interface to configure it. You can, for instance, change its addressing scheme to DHCP, manual, or a static address. Similarly, you'll also be able to tweak the DNS settings and even update the hostname of the machine running Ajenti without firing up a terminal.

In addition to these functions, you can add more functionality to the base Ajenti installation by heading to *General | Plugins* (Figure 4).

## Conclusion

While some server management tasks can be performed from inside many modern desktop environments, Ajenti is specifically designed for remote management. It can save you time and effort in dabbling with various configuration files (assuming you have the know-how).

Ajenti also makes sense for managing hosts that have a headless installation.

Give Ajenti a shot, and you'll see how it will make conducting regular remote administration tasks significantly easier. ■■■

## Info

- [1] Ajenti: <https://ajenti.org>
- [2] Webmin: <http://www.webmin.com>
- [3] Installing Ajenti 1 on Ubuntu: <https://support.ajenti.org/knowledge-bases/5/articles/1121-installing-on-ubuntu>
- [4] Setting up a PHP website with Ajenti V: <https://support.ajenti.org/knowledge-bases/5/articles/1134-setting-up-a-php-website-with-ajenti-v-wordpress-example>
- [5] Installing Ajenti 2 with pip: <http://docs.ajenti.org/en/latest/man/install.html#install-ajenti-2>

## Author

**Mayank Sharma** is a technology writer. You can read his scribbblings in various geeky magazines on both sides of the pond.



# What?!

I can get my issues SOONER?



## Available anywhere, anytime!

Sign up for a digital subscription and enjoy the latest articles on trending topics, reviews, cool projects and more...

**Subscribe to the PDF edition:** [shop.linuxnewmedia.com/digisub](http://shop.linuxnewmedia.com/digisub)

**Now available on ZINIO:** [bit.ly/Linux-Pro-ZINIO](http://bit.ly/Linux-Pro-ZINIO)







## A modern diff utility

# File Comparison

With support for more than 60 file formats, diffoscope extends the power of diff beyond the plain text or HTML file. *By Bruce Byfield*

The first command in Unix-like systems for comparing files and directories was `diff`. Originally written by Douglas McIlroy and first appearing in Unix 5th Edition in 1974, `diff` rapidly became an essential programming tool. Today, the original command is still available, and most programming languages have their own versions of `diff`. However, `diff` and its derivatives generally have one limitation: With few exceptions, most of them work only with plain text or markup languages like HTML. A new variation called `diffoscope` [1], which was released in mid-2020, brings a new level of functionality to file comparison.

Diffoscope is developed primarily by Debian’s Reproducible Builds project [2], which aims to increase the robustness

### Author

**Bruce Byfield** is a computer journalist and a freelance writer and editor specializing in free and open source software. In addition to his writing projects, he also teaches live and e-learning courses. In his spare time, Bruce writes about Northwest coast art (<http://brucebyfield.wordpress.com>). He is also co-founder of Prentice Pieces, a blog about writing and fantasy at <https://prenticepieces.com/>.

and security of Debian packages by ensuring that they always build the same way. Given Debian’s nearly 60,000 packages and the variety of hardware available, this is no small task, especially considering that small errors in code can be hard to trace. Diffoscope was written to make this task easier by quickly tracking down differences between two files that are supposed to be identical but perform differently. As a side effect, diffoscope provides a modern `diff` utility that works across most programming languages and brings the power of `diff` to desktop users and non-programmers, especially writers who wish to compare drafts. Already, diffoscope supports over 60 binary formats that range from files and filesystems to audio and text files, including MS Word, LibreOffice Writer, and PDF (Table 1). And more seem likely to follow.

Diffoscope’s basic command structure is:

```
diffoscope FILE1 FILE2
```

If only one file or directory is given, then diffoscope attempts to compare the given file with the last file compared – a desperate act that will only occasionally be useful. For convenience, the com-

mand can be piped through less or more. You might also add the `--progress` option for large files like DVD images. If you are dealing with large files, you might also run up against the built-in limits for output. Rather than resetting them, you can cancel all of them with the option `--no-default-limits`.

Output is to standard output by default, but you can also save to file. The output shows the content of the first file in red text, with each line prefaced by a minus sign, and the content of the second file in white text prefaced by a plus sign. At the top of the output, you’ll find statistics that vary with the file type. For example, in Figure 1, the files share LibreOffice’s `.odt` format, and the statistics are the file names, the amount of text in each file that differs, and the number of total words in each file. By contrast, in Figure 2, a directory `diff` is prefaced by file listings, file permissions, and other attributes. The output is driven by context, ensuring that it is useful for more than the `diff` itself.

### Output Formatting Options

Besides standard input, diffoscope’s output can be saved to several file formats. To write output to a text file, add the op-



**Table 1: Supported Formats**

Android APK files	LLVM IR bitcode files
Android boot images	LZ4 compressed files
ar(1) archives	macOS binaries
Berkeley DB database files	Microsoft Windows icon files
bzip2 archives	Microsoft Word .docx files
Character/block devices	Mono Portable Executable files
ColorSync color profiles (.icc)	Multimedia metadata
coreboot CBFS filesystem images	OCaml interface files
cpio archives	Ogg Vorbis audio files
Dalvik .dex files	OpenOffice/LibreOffice .odt files
Directories	OpenSSH public keys
Debian buildinfo files	OpenWRT package archives (.ipk)
Debian .changes files	PDF documents
Debian source packages (.dsc)	PGP signatures
Device Tree Compiler blob files	PGP signed/encrypted messages
ELF binaries	PNG images
ext2/ext3/ext4/Btrfs/FAT filesystems	PostScript documents
freedesktop.org fontconfig cache files	RPM archives
Free Pascal files (.ppu)	Rust object files (.deflate)
gettext message catalogs	SQLite databases
GHC Haskell .hi files	SquashFS filesystems
GIF image files	Statically linked binaries
Git repositories	Symlinks
GNU R database files (.rdb)	Tape archives (.tar)
GNU R Rscript files (.rds)	tcpdump capture files (.pcap)
Gnumeric spreadsheets	Text files
Gzipped files	TrueType font files
ISO 9660 CD images	WebAssembly binary module
Java .class files	XML binary schemas (.xsb)
JavaScript files	XML files
JPEG images	XZ compressed files
JSON files	

tion `--text OUTPUT-FILE`, giving the full path. You can also color-code an output text file with `--text-color WHEN`, replacing *when* with *never*, *auto*, or *always*. Color is enabled automatically in standard output, but disabled by default when you write to a file. Similarly, an HTML file is named with `--html OUTPUT-FILE`. Color is not supported for HTML files, but you can write a multi-HTML file using `--html OUTPUT-DIRECTORY`, so you can absorb the output in small chunks, and `--css URL` to format the output as desired. If you are using JavaScript, both text and HTML output can be formatted using `--jquery URL`. Other supported file format options are `--json OUTPUT-FILE`, `--markdown OUTPUT-FILE`, and `--restructured-text OUTPUT_FILE`, all three of which can be used for either files or for standard output. In all these formats, `--output-empty` can be used to write a file to report no differences.

## Output Limit Options

Coming from an era of memory limitations, `diff` is economical, by default writing just a few lines so that the context of a difference can be read. By contrast, `diffoscope`, written in mid-2020 has limits that are so high that, for all practical purposes, it often has no limits. Instead, if you want to limit `diffoscope`'s output – perhaps to make the output more manageable – you have to deliberately add limits. The number of bytes in an output report is unlimited by default, but you can use `--max-text-report-size BYTES` to define a limit. Alternatively, you can use `--max-text-report-size BYTES` to change the default of 409,600, or, if using `--html OUTPUT-DIRECTORY`, you can use `--max-page-size-child BYTES` to change the size of the separate pages of an HTML report from the default of 204,800. Still another alternative is to change the default 1,024 lines for a unified-diff block – that is, for separate chunks of the report. These options are primarily for comparisons of long files, such as .iso images, and are generally irrelevant when dealing with files in MS Word or LibreOffice format unless you are comparing complete manuscripts.

## Difference Calculation Options

A number of options modify how `diffoscope` makes its comparisons. `--exclude`

```
bb@nanday:~/test$ diffoscope draft1.odt draft2.odt
-- draft1.odt
++ draft2.odt
  odt2txt
  @@ -1,419 +1,451 @@
-3003words
+3333words

Chapter 23: Siblings at War

-They got as far as the ferns before they dove to the ground.
-Moss dampened on Talson's knees and elbows, and the smell of
-cedars was as strong as smoke. The murmurs and inconsistent
-feet closed in from behind, and he eased up to his knees and
-froze like a rabbit, hoping enough dark was left to cloak him.
-
-The militia members did not march like housecarls. They ambled
-along in twos and threes, bags tied to their belts, and bread
-twisted around the spears that rested across their shoulders.
+The militia did not march like housecarls. They ambled along in
+twos and threes, bags tied to their belts, and bread twisted
+around the spears that rested on their shoulders. "Who's like
+us?" a cadence-caller cried, and a few replied, "Only the
+dead!" Their voices were already weary of the repetition.
+Maybe two hundred were strung along the road. An owl standard
```

**Figure 1: File comparison includes stats useful for the format.**

```
bb@nanday:~/test$ diffoscope ./draft1/ ./draft2/
-- ./draft1/
+++ ./draft2/
-- file list
@@ -1,1,2 @@
-draft1.odt
+draft1.odt
+draft2.odt
-- stat {}
@@ -1,8 +1,8 @@

Size: 4096          Blocks: 8          IO Block: 4096   directory
Links: 2
Access: (0755/drwxr-xr-x)  Uid: ( 1000/      bb)   Gid: ( 1000/      bb)

-Modify: 2020-08-20 19:54:10.640801760 +0000
+Modify: 2020-08-20 19:54:25.121100600 +0000

Birth: -
```

**Figure 2:** Diffoscope can also compare directory contents and structure.

GLOB\_PATTERN and --exclude-command REGEX\_PATTERN are different names for the same option and can be used with either files or directories. When working with directories, you can set whether permissions and other file attributes are used with --exclude-directory-metadata SETTING, which can be completed with *auto*, *yes*, *no*, or *recursive*. In addition, you can opt to enable fuzzy logic, controlling how minor differences are handled. A setting of 0 means that all matches must be exact; however, the meaning of the default of 60 or the maximum of 400 has to be discovered through trial and error, since it is currently undocumented.

Other options are reminiscent of diff itself setting the number of lines to com-

pare. Use --max-diff-input-lines LINES to compare the number of lines (the maximum is 4,194,304). You can also set the maximum number of lines per diff block with --max-diff-block-lines-saved LINES.

## Information Options

For those not installing from a distribution, diffoscope includes --list-tools DISTRO, which lists dependencies, and --list-missing-tools DISTRO, which lists the external tools not currently installed on a system. You can get a sense of which distributions are contributing to diffoscope from some of the information options. In both commands, the distro must be specified as either Arch, Debian, or FreeBSD. For Debian, the option --list-debian-substvars displays

dependencies that are required or simply recommended. All three options list what is required for full functionality. Depending on how you use diffoscope, it may function without all the packages listed (Figure 3).

## A Work in Progress

As I write, diffoscope is still in rapid development. As a result, some features are missing. Unlike diff, diffoscope has yet to support a two-column view, a feature that makes comparison handy. Moreover, while the list of supported formats is already impressive, it would be useful to support all the LibreOffice modules.

However, it seems clear that diffoscope has quickly expanded beyond its original purpose. In particular, as a writer, I was pleased to find that diffoscope is well-suited to comparing drafts, something that I do regularly when writing fiction or long non-fiction. After using it for a couple of weeks, already I wonder how I got along without it. It has become a tool that I use daily and have come to depend on. ■■■

## Info

- [1] diffoscope: <https://diffoscope.org/>
- [2] Reproducible Builds: <https://wiki.debian.org/ReproducibleBuilds>

```
bb@nanday:~$ diffoscope --list-tools debian
External-Tools-Required: Rscript, abootimg, apktool, bsdtar, bzip2, cbfstool, cd-iccDump, cmp, compare, convert, db_dump, diff, docx2txt, dumpxsb, enjarify, ftdump, ffprobe, getfacl, ghc, gifbuild, gpg, gzip, identify, img2txt, isoinfo, javap, js-beautify, lipo, llvm-bcanalyzer, llvm-dis, lsattr, lz4, msgunfmt, nm, objcopy, objdump, ocamlobjinfo, odt2txt, oggDump, otool, pdftotext, pedump, pgpdump, ppudump, procyon, ps2ascii, readelf, showtff, sng, sqlite3, sconvert, ssh-keygen, stat, tcpdump, unsquashfs, wasm2wat, xxd, xz, zipinfo, zipnote
Available-in-Debian-packages: abootimg, acl, apktool, binutils-multiarch, bzip2, caca-utils, colord, coreutils, db-util, default-jdk-headless | default-jdk | java-sdk, device-tree-compiler, diffutils, docx2txt, e2fsprogs, enjarify, ffmpeg, fontforge-extras, fp-utils, genisoimage, gettext, ghc, ghostscript, giflib-tools, gnumeric, gnupg, gzip, imagemagick, jsbeautifier, libarchive-tools, llvm, lz4 | liblz4-tool, mono-utils, ocaml-nox, odt2txt, oggvideotools, openssh-client, pgpdump, poppler-utils, procyon-decompiler, r-base-core, sng, sqlite3, squashfs-tools, tcpdump, unzip, xmlbeans, xxd | vim-common, xz-utils, zip
```

**Figure 3:** For those building diffoscope from scratch, the command includes lists of packages needed for full functionality.





# CLEAR OFF YOUR BOOKSHELF WITH DIGITAL ARCHIVES

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

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

**2019**  
Archives  
Available  
Now!

[bit.ly/2019-Digital-Archives](https://bit.ly/2019-Digital-Archives)





## Graphing the pandemic with open data

# Visualize

Lots of COVID-19 data is available through online REST APIs. With a little ingenuity and some open source tools, you can extract and analyze the data yourself. *By Chris Dock*

**T**ravel is broadening. You experience different cultures, see issues from different angles, and meet fun and unique people. I love living abroad, but I have to admit that I have less access to the news from home. Unfortunately, news outlets today spend more time on opinion than facts, and sometimes I just want the unvarnished truth. During the pandemic era, I am especially anxious to learn about the challenges faced by my family back home.

The good news is that lots of open data on COVID-19 is available on the Internet via REST API calls. This data might be too dry for some, but if you want to get your own impressions of the COVID-19 crisis, without the sometimes intrusive “analysis” of newscasters and commentators, this free Internet data is a valuable resource. This article describes how to access and display freely available COVID-19 data using

open source tools. And, if you’ve already had your fill of COVID-19 information, the techniques I’ll describe in this article will also help you with other kinds of government and academic data available through REST APIs.

### CovidAPI

The CovidAPI project [1] provides COVID data based on the well respected Johns Hopkins University dataset [2]. The original Johns Hopkins data is available in CSV form. Argentine software developer Rodrigo Pomba converted the data to JSON time series format. According to the documentation [3], the goal of the project is to make the data “queryable in a manner in which it could be easily consumed to build public dashboards.”

The CovidAPI data is organized by country using the list of ISO country codes [4]. Use the `curl` command in a terminal window to send a URL that will access the data for a specific country and date:

```
curl https://covidapi.info/api/v1/?country/USA/2020-06-15
```

Calling this API, in this case with `curl`, will return the following JSON object.

```
{
  "count": 1,
  "result": {
    "2020-06-15": {
      "confirmed": 2114026,
      "deaths": 116127,
      "recovered": 576334
    }
  }
}
```

This command is an easy way to get the daily report for a specific country and date, but if you want to visualize and analyze the data yourself, you might prefer to request the values for all dates. If you leave off the date, you’ll get the data for all available dates:



```
curl https://covidapi.info/api/v1/
country/USA
```

This command returns one giant JSON message containing the records for every day in the dataset. However, I ran into problems parsing out the individual days due to the dashes that were part of the date. As an alternative approach, I chose to write a small Bash script to fetch the count of the day records then iterate through the list of days to retrieve the COVID-19 information for each day (Listing 1). Most of the steps are self-explanatory if you are familiar with Bash

scripts, but see the comment lines for additional information.

One part of the script that might not be obvious is how I calculate the date.

```
DATE=`date --date="12:00 today"
-$IDX days" +%Y-%m-%d`
```

The date command subtracts a given number of days from the current date and formats the output as a YYYY-MM-DD string.

Of course, it would be inefficient to download hundreds of days worth of data each time if I just want yester-

day's data. Because of this, the script verifies if the data has been retrieved before making the REST API call to retrieve the data. The first time you run the script, you get all the data, and on subsequent runs, you only get the new data.

## Data by State

Retrieving COVID-19 figures for a whole country is useful for comparing one country against another, but it is less than helpful if you want to know what is really happening locally. The Covid Tracking Project [5] provides COVID-19

Listing 1: covid19.sh

```
001 #!/bin/bash
002
003 get_count()
004 {
005     GATHERCOUNTRY=$1
006
007     # get the count of days since the start
008     CNT=`curl https://covidapi.info/api/v1/
009         country/${GATHERCOUNTRY} 2>/dev/null | jq '.count'`
010     echo $CNT
011 }
012 gather_state()
013 {
014     GATHERSTATE=$1
015     cnt=$2
016
017     echo gather state $GATHERSTATE $cnt days
018     DATAFILE=covid19_${GATHERSTATE}.Data
019
020     # from beginning until yesterday
021     IDX=$cnt
022
023     # absolute values, followed by daily delta
024     if [ ! -f $DATAFILE ]
025     then
026         echo "date positive hospitalized deaths " >
027             $DATAFILE
028     fi
029     while [ $IDX -gt 0 ]
030     do
031         #DATE=`date --date="$IDX days ago" +%Y%m%d`
032         DATE=`date --date="12:00 today -$IDX days" +%Y%m%d`
033         FILEDATE=`date --date="12:00 today -$IDX days"
034             +%Y-%m-%d`
035         CMD="curl https://api.covidtracking.com/v1/
036             states/${GATHERSTATE}/${DATE}.json"
037         grep $FILEDATE $DATAFILE >/dev/null
038         if [ $? -eq 1 ]
039         then
040             SINGLE=`CMD 2>/dev/null `
041             error=`echo $SINGLE | jq ".error"`
042             if [ $error == "true" ]
043             then
044                 # nothing to output
045                 # echo oops looks bad $DATE
046
047                 positive=0
048                 hospitalized=0
049                 deaths=0
050             else
051                 positive=`echo $SINGLE | jq ".positive"`
052                 deaths=`echo $SINGLE | jq ".death"`
053                 hospitalized=`echo $SINGLE | jq
054                     ".hospitalizedCurrently"`
055                 if [ $positive == "null" ]; then positive=0; fi
056                 if [ $deaths == "null" ]; then deaths=0; fi
057                 if [ $hospitalized == "null" ]; then
058                     hospitalized=0; fi
059                 echo $DATE $IDX
060                 echo "$FILEDATE $positive $hospitalized $deaths
061                     " >> $DATAFILE
062             #else
063             # echo not doing $FILEDATE
064             fi
065
066             IDX=$(( $IDX - 1 ))
067         done
068     }
069 }
070
071
072 gather_data()
073 {
074     GATHERCOUNTRY=$1
075     cnt=$2
076
077     echo gather $GATHERCOUNTRY
078     DATAFILE=covid19_${GATHERCOUNTRY}.data
079
080     # absolute values, followed by daily delta
081     if [ ! -f $DATAFILE ]
082     then
083         echo initializing
084         echo "date confirm deaths recover " > $DATAFILE
085     fi
```

## Listing 1: covid19.sh (continued)

```

086                                     120         echo $DATE $IDX
087 # from beginning until yesterday    121         echo "$DATE $confirm $deaths $recover " >>
088 IDX=$cnt                             $DATAFILE
089                                     122         #else
090 deltadeaths=0                         123         # echo not doing $DATE
091 deltaconfirm=0                       124         fi
092 deltarecover=0                       125
093                                     126         fi
094 while [ $IDX -gt 0 ]                 127
095 do                                     128         IDX=$(( $IDX - 1 ))
096 #DATE=`date --date="$IDX days ago"    +%Y-%m-%d` 129 done
097 DATE=`date --date="12:00 today -$IDX 130 }
098                                     131
099 CMD="curl https://covidapi.info/api/v1/country/${GAT 132 CNT=`get_count USA`
HERCOUNTRY}/${DATE}"
100
101 grep $DATE $DATAFILE >/dev/null
102 if [ $? -eq 1 ]
103 then
104
105 #
106 # we only do this if this date hasn't been
107 # retrieved
108 #
109 SINGLE=`$CMD 2>/dev/null `
110 ERR=`echo $SINGLE | grep "404 Not Found" | wc -l`
111 #
112 # only if date found
113 #
114 if [ $ERR -eq 0 ]
115 then
116     deaths=`echo $SINGLE | jq '.' | grep deaths |
117             sed 's/.*: //' | sed 's/,/' `
118     confirm=`echo $SINGLE | jq '.' | grep confirm |
119             sed 's/.*: //' | sed 's/,/' `
120     recover=`echo $SINGLE | jq '.' | grep recover |
121             sed 's/.*: //' | sed 's/,/' `
122
123     echo $deaths $confirm $recover
124
125     gather_data $STATE $CNT
126
127     CNT=`get_count $STATE`
128     gather_data $STATE $CNT
129
130     CNT=`get_count DEU`
131     gather_data DEU $CNT
132
133     CNT=`get_count ESP`
134     gather_data ESP $CNT
135
136     CNT=`get_count GBR`
137     gather_data GBR $CNT
138
139     gnuplot graphs.gp
140
141
142
143
144
145
146
147
148
149
150
151
152
153

```

## Listing 2: Hospitalizations and Deaths by State

```

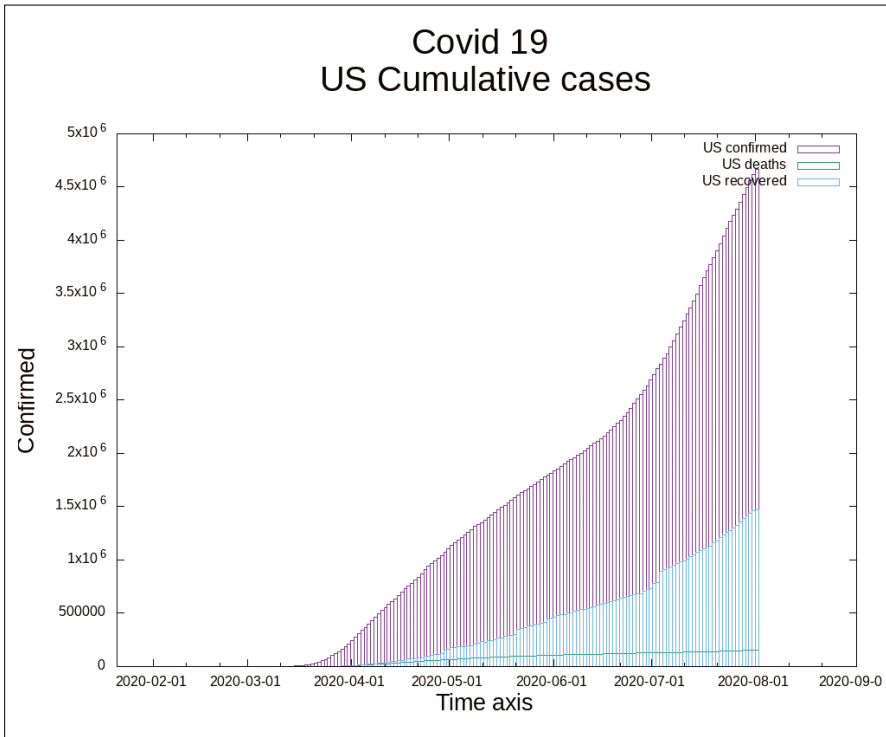
{
  "date": 20200415,
  "state": "MN",
  "positive": 2321,
  "negative": 41245,
  "pending": null,
  "hospitalizedCurrently": 197,
  "hospitalizedCumulative": 445,
  "inIcuCurrently": 93,
  "inIcuCumulative": 175,
  "onVentilatorCurrently": null,
  "onVentilatorCumulative": null,
  "recovered": 853,
  "dataQualityGrade": "A",
  "lastUpdateEt": "4/14/2020 17:00",
  "dateModified": "2020-04-14T17:00:00Z",
  "checkTimeEt": "04/14 13:00",
  "death": 87,
  "hospitalized": 445,
  "dateChecked": "2020-04-14T17:00:00Z",
  "totalTestsViral": 43566,
  "positiveTestsViral": null,
  "negativeTestsViral": null,
  "positiveCasesViral": null,
  "fips": "27",
  "positiveIncrease": 156,
  "negativeIncrease": 1540,
  "total": 43566,
  "totalTestResults": 43566,
  "totalTestResultsIncrease": 1696,
  "posNeg": 43566,
  "deathIncrease": 8,
  "hospitalizedIncrease": 40,
  "hash": "9521e0ce1f2b1ef5aa1a81bec48961d85170d78",
  "commercialScore": 0,
  "negativeRegularScore": 0,
  "negativeScore": 0,
  "positiveScore": 0,
  "score": 0,
  "grade": ""
}

```



**Table 1: Sample of Downloaded Data**

Country	Date	Confirmed	Deaths	Recovered
USA	2/21/20	15	0	5
Germany	2/21/20	16	0	14
England	2/21/20	9	0	8
Spain	2/21/20	2	0	2



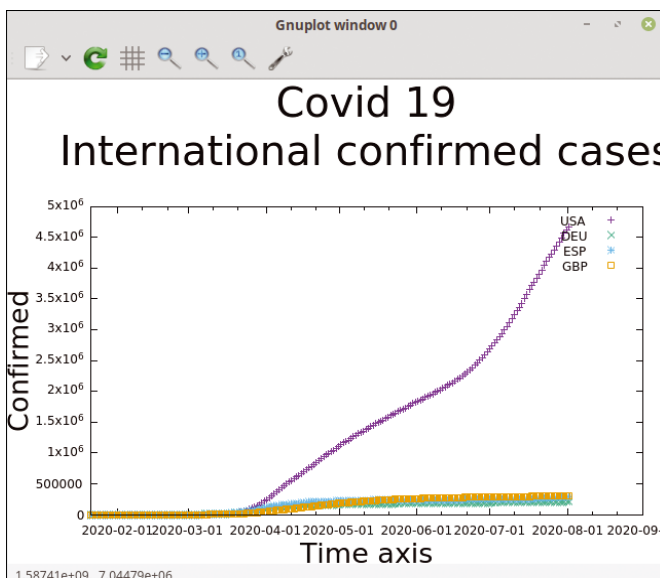
**Figure 1: US COVID-19 statistics.**

data by US state. (Similar projects track pandemic data for other countries – consult your local health resources.)

Just like at the national level, it is possible to retrieve all COVID-19 information by US state for a given date

with REST API calls. For instance, to obtain data on the state of Minnesota for August 21, 2020:

```
curl https://api.covidtracking.com/v1/states/mn/20200821.json | jq ". "
```



**Figure 2: International confirmed cases.**

hospitalizations up until that day. (Listing 2) Also included were the incremental changes in positive as well as negative tests results. Using this information, I could have graphed how quickly COVID-19 is spreading by graphing `positiveCasesViral` vs `totalTestsViral` or by graphing `hospitalizedCurrently` over time.

I settled on gathering positives, numbers of people hospitalized, and deaths at a state level. I didn't try to verify that all state totals added up at the national level, as I suspect there can be delays in the reporting chain from the local to the national level.

I can imagine that massive effort to come up with a common structure, as well as getting all the participants to gather all of these types of data. Despite all of their efforts, sometimes the data returned contained fields that were blank, had zeros, or simply had the value null.

### Comparing Countries

My COVID-19 gathering script will collect the information from four different countries (Great Britain, USA, Spain, and Germany), as well as statistics for a few US states. This data is temporarily stored in a text file but the information that I am gathering essentially looks similar to Table 1.

### Plotting the Data

Tabular data is actually very dense and conveys a lot of information, however, it does have the side effect of being rather dry, and when the volume of data is too great, it can be difficult to spot trends. I remembered the graphing tool gnuplot [6], which I have used in the past to give data a friendlier look (Figure 1).

Gnuplot is a cross-platform 2D and 3D graphing tool. You can use gnuplot to create line graphs, bar charts, histograms, or even candlestick charts.

Gnuplot is a command-line program that can accept its input via a pipe and send its output to standard output. The output from gnuplot can be redirected to a file, but it is also possible to define where the output should be written.

Gnuplot is available in the package repositories of many popular Linux distributions:

```
sudo apt-get install gnuplot
```

Despite the name, gnuplot is not affiliated with the GNU project, and, al-

The state data, unlike the national data, contains an amazing number of statistics submitted by the health authorities. The sheer number of values provided can perhaps only truly be appreciated by an epidemiologist or a statistician.

You can see how many people were hospitalized on a given day or the total number of

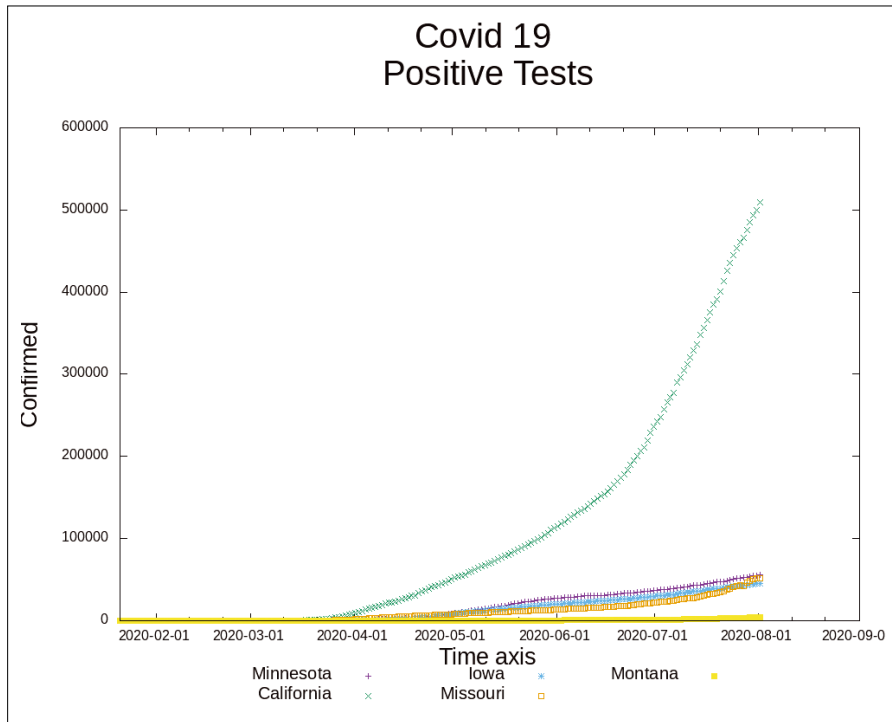


Figure 3: Comparing positive tests by state.

### Listing 3: graphs.gp

```

01 # Reset all plotting variables to their default values.
02 reset
03 clear
04
05 # set the terminal type (ie output format)
06 # also set the width and height
07 set term png size 1000, 800
08
09 # set x & y axis description
10 set xlabel font ",20" "Time axis"
11 set ylabel font ",20" "Confirmed"
12
13 # setup x and y axis values
14 set xdata time
15 set timefmt "%Y-%m-%d"
16 set xrange [ "2020-01-22":* ]
17 set format x "%Y-%m-%d"
18 set yrange [ 0:* ]
19
20 # graph of usa graph
21 set output 'country.png'
22 set title font ",30" "Covid 19 \nUS Cumulative cases"
23 plot "covid19_USA.data" using 1:2 title 'US confirmed'
  with boxes, \
24     "" using 1:3 title 'US deaths' with boxes, \
25     "" using 1:4 title 'US recovered' with boxes
26
27
28 # graph of minnesota statistics
29 set output 'statemn.png'
30 set title font ",30" "Covid 19 \nMN Cumulative cases"
31 plot "covid19_mn.Data" using 1:2 title 'positive' , \
32     "" using 1:3 title 'hospitalized' , \
33     "" using 1:4 title 'deaths'
34
35
36 # set legend below the graph
37 set key below font ",15"
38
39 # compare a few states against each other
40 set output 'statecompare.png'
41 set title font ",30" "Covid 19 \nPositive Tests"
42 plot "covid19_mn.Data" using 1:2 title 'Minnesota' , \
43     "covid19_ca.Data" using 1:2 title 'California' , \
44     "covid19_ia.Data" using 1:2 title 'Iowa' , \
45     "covid19_mo.Data" using 1:2 title 'Missouri' , \
46     "covid19_mt.Data" using 1:2 title 'Montana'
47
48
49 # compare USA against other countries
50 set output 'confirm.png'
51 set title font ",30" "Covid 19 \nInternational confirmed
  cases"
52 plot "covid19_USA.data" using 1:2 title 'USA' , \
53     "covid19_DEU.data" using 1:2 title 'DEU' , \
54     "covid19_ESP.data" using 1:2 title 'ESP' , \
55     "covid19_GBR.data" using 1:2 title 'GBP'

```

though it is free to use and redistribute, it has an unusual license. Because of this license, it is not possible to redistribute modified versions of the source code: “Modifications are to be distributed as patches to the released version.” It is still possible to release your own modified binaries of gnuplot, well, with a few conditions that are covered in the copyright [7] statement.

Gnuplot makes it possible to save your graphed output in quite a few different formats. You can save the output in all the common graphic file formats – PNG, GIF, JPEG, and SVG, but also other unusual types of output such as a Postscript, PDF, or LaTeX file.

When you start gnuplot as a command interpreter, it creates a graphical window where your graphed data will be displayed. Thus you can interactively test out some plotting options (Figure 2).

One of the additional advantages to running gnuplot as an interpreter is that,



once you are satisfied with the results, you can save the plot datafile. Conversely you can also load a datafile into the interpreter.

```
load "plotcommands.ext"
save "plotcommands.ext"
```

The actual script for generating graphs from the collected data is quite short (Listing 3). This script actually demonstrates how powerful gnuplot is. The main steps for drawing any graph are:

- defining the units on the X and Y axis
- labeling the axis
- plotting the data

These steps are all depicted in Listing 3.

The plot statement in Listing 3 is a bit confusing until you recognize that each set of data can come from a different file, and using 1:2 means that column 1 from the data file will be on the X axis and column 2 will be on the Y axis.

The comparative graph of the individual state infections, Figure 3, is much more helpful than viewing all the US figures in tabular form.

## Conclusion

The scripts described in this article are available at the *Linux Magazine* website [8]. I could have gone even further and collected information from the Twitter accounts of state governors and health departments [9], but I don't think important health information can be summarized into 288 characters. Besides, I am not the biggest follower on Twitter. ■■■

## Author

Christopher Dock is a senior consultant at T-Systems on site services GmbH. When he is not working on integration projects, he likes to experiment with Raspberry Pi solutions and other electronics projects. You can read more about his work at <http://blog.paranoidprofessor.com>. If you email him at [christopher.dock@t-systems.com](mailto:christopher.dock@t-systems.com), he will gladly answer any questions.

## Info

- [1] CovidAPI Project: <https://covidapi.info/>
- [2] Johns Hopkins Coronavirus Resource Center: <https://coronavirus.jhu.edu/>
- [3] CovidAPI Documentation: <https://documenter.getpostman.com/view/2568274/SzS8rjbe?version=latest#intro>
- [4] ISO country codes: [https://en.wikipedia.org/wiki/List\\_of\\_ISO\\_3166\\_country\\_codes](https://en.wikipedia.org/wiki/List_of_ISO_3166_country_codes)
- [5] Covid Tracking Project: <https://covidtracking.com/>
- [6] gnuplot: <http://www.gnuplot.info/>
- [7] gnuplot copyright: <https://sourceforge.net/p/gnuplot/gnuplot-main/ci/master/tree/Copyright>
- [8] Scripts used in this article: <ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/240/>
- [9] Twitter account information: <https://covid-19-apis.postman.com/>

# 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](http://admin-magazine.com/HPC)

## Manipulating stored geocoordinates in cellphone photos

# Obfuscation Filter

Mike Schilli loves his privacy. That's why he's created a Go program that adds a geo-obfuscation layer to cellphone photos before they are published on online platforms to prevent inquisitive minds from inferring the location.

By Mike Schilli

If you sell your stuff online, you might overlook the potential risk of sales-promoting cellphone photos revealing highly sensitive private information. When you take a picture of the goods at home with your cellphone, the image file may also contain the geodata with which the private address can be determined to within a few yards. Large sales platforms generally do not publish this meta-information, but who wants to give away more information than is absolutely necessary on Ebay or Facebook?

The cellphone also erases geodata directly if desired – but then it looks as if the user has something to hide. That's why the self-written Go program in this issue adds a geo-obfuscation layer to image files to make sure that the geocoordinates are randomly blurred. From this, it might be possible to determine

## 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](mailto:mschilli@perlmeister.com) he will gladly answer any questions.



the seller's location down to the neighborhood, but not the exact address.

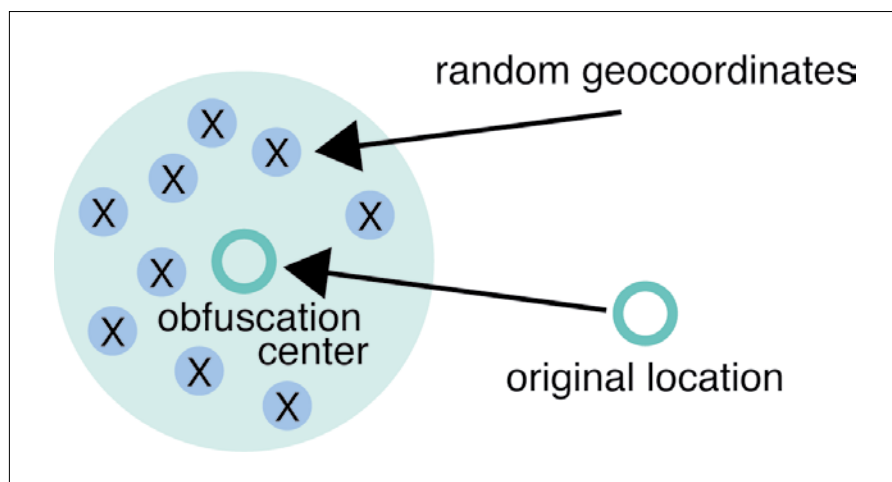
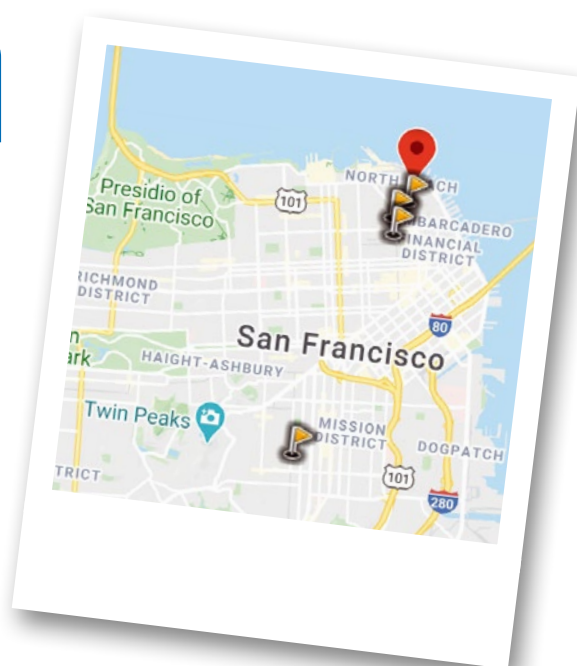
## Matches in the Radius

The procedure's goal is to move a photo's geotags randomly to an area within a defined action radius. If several snapshots are made, the target values are all within the action radius. In order to avoid anybody determining the center – and thus the location of the photographer – by analyzing hundreds of shots, the geofuzzer also shifts the center of the random circle to a neighboring area beforehand. To do this, it uses fixed, but secret, values for

latitude and longitude (Figure 1).

An image file's geolocation is contained in the JPEG format's Exif tags [1] and can be read out with tools, such as `exiftool`, or on the `GeoImgr` website [2]. The latter even conveniently displays the shot's location on a Google map.

Figures 2 and 3 each show the photo's geotags – a picture of a box with a Google Voice Kit I wanted to sell on Ebay. The original in Figure 2 shows my home address in San Francisco where I snapped the photo in my study. After calling the `geofuzz` program, the image file's geolocation shifts further north to



**Figure 1:** From the original location, the algorithm jumps to a new point and randomly selects geocoordinates within a predefined radius.

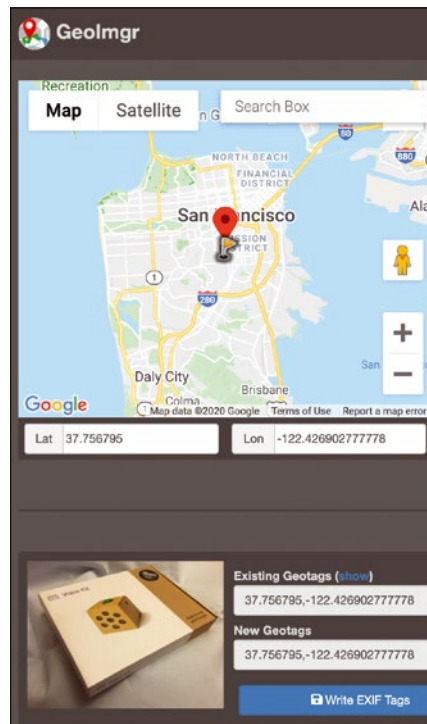
Lead Image Google Maps and Mike Schilli



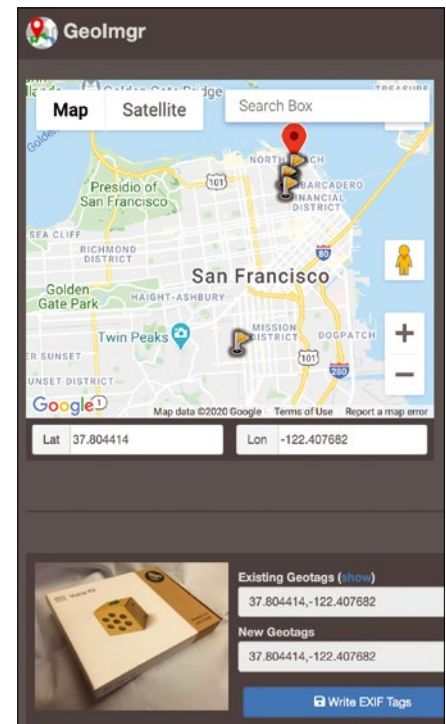
the Financial District. Figure 3 shows additional target values after several successive calls to the fuzzer, which scattered the results within the set action radius.

The geofuzz program generated from Listing 1 expects the name of the JPEG file to be manipulated on the command line, as shown in Listing 2. For the user to follow along, the running program prints both the original and the modified geolocations on Std-out. The fuzzer modifies the specified file directly, and the user can now post it without revealing too much about their location.

Look at the numbers in the output: My home in San Francisco is located at longitude 37° north and latitude 122° west, so the value for 37 is positive and 122 is negative. For comparison: Munich's Marienplatz is located at the geo-coordinates 48.137365 and 11.575127, which can easily be retrieved in Google Maps by right-clicking the mouse on the corresponding location and selecting



**Figure 2:** The actual location of this photo is near San Francisco's Mission District.



**Figure 3:** Successive calls of the geotag fuzzer move the geotag to locations within the action radius in the Financial District.

### Listing 1: geofuzz.go

```

001 package main
002
003 import (
004     "bytes"
005     "fmt"
006     exif "github.com/xor-gate/goexif2/exif"
007     "math"
008     "math/rand"
009     "os"
010     "os/exec"
011     "path/filepath"
012     "time"
013 )
014
015 func usage(msg string) {
016     fmt.Printf("%s\n", msg)
017     fmt.Printf("usage: %s image.jpg\n",
018         filepath.Base(os.Args[0]))
019     os.Exit(1)
020 }
021
022 func main() {
023     if len(os.Args) != 2 {
024         usage("Missing argument")
025     }
026
027     img := os.Args[1]
028
029     lat, lon, err := geopos(img)
030     if err != nil {
031         panic(err)
032     }
033
034     latFuzz, lonFuzz := fuzz(lat, lon)
035
036     fmt.Printf("Was: %f,%f\n", lat, lon)
037     fmt.Printf("Fuzz: %f,%f\n",
038         latFuzz, lonFuzz)
039     patch(img, latFuzz, lonFuzz)
040 }
041
042 func patch(path string,
043     lat, lon float64) {
044     var out bytes.Buffer
045     cmd := exec.Command(
046         "exiftool", path,
047         fmt.Sprintf("-gpslatitude=%f", lat),
048         fmt.Sprintf("-gpslongitude=%f", lon))
049     cmd.Stdout = &out
050     cmd.Stderr = &out
051
052     err := cmd.Run()
053     if err != nil {
054         panic(out.String())
055     }
056 }

```



## Listing 1: geofuzz.go (continued)

```

057
058 func geopos(path string) (
059     float64, float64, error) {
060     f, err := os.Open(path)
061     if err != nil {
062         return 0, 0, err
063     }
064
065     x, err := exif.Decode(f)
066     if err != nil {
067         return 0, 0, err
068     }
069
070     lat, lon, err := x.LatLong()
071     if err != nil {
072         return 0, 0, err
073     }
074
075     return lat, lon, nil
076 }
077
078 func fuzz(lat, lon float64) (
079     float64, float64) {
080     r := 1000.0 / 111300 // 1km radius
081
082     // secret center
083     lat += .045
084     lon += .021
085
086     s1 := rand.NewSource( // random seed
087         time.Now().UnixNano())
088     r1 := rand.New(s1)
089
090     u := r1.Float64()
091     v := r1.Float64()
092
093     w := r * math.Sqrt(u)
094     t := 2.0 * math.Pi * v
095     x := w * math.Cos(t)
096     y := w * math.Sin(t)
097
098     x = x / math.Cos(lat*math.Pi/180.0)
099     return lat + x, lon + y
100 }

```

## Listing 2: Invoking the Fuzzer

```

$ geofuzz ebay.jpg
Was: 37.756795,-122.426903
Fuzz: 37.804414,-122.407682

```

*What's here?* in the context menu (Figure 4). This confirms that Munich is further north than San Francisco and not west of the prime meridian (zero degree longitude), but east. Therefore, the value for Munich's 11° longitude is positive.

## Reading Is Easier than Writing

If the number of arguments passed on the command line is less than expected, line 24 in Listing 1 branches to the `usage()` function that starts in line 15, which displays the error, demonstrates the correct use, and terminates the program with an exit code of 1.

The geodata are available as latitude and longitude in degrees, minutes, and seconds in the Exif tags of the photo file's JPEG format. The `go-exif2` library on GitHub makes it surprisingly easy to read this relatively complex structure [1]. Luckily, I remembered that I had used the library once before in this magazine in an application for geosearching in a photo collection [3].

The `geopos()` function in line 58 of Listing 1 opens the image file passed to it by name, decodes the JPEG format with the call of the library function `Decode()`, and finds the latitude and longitude information of the location stored in the Exif tags with `LatLong()`. The function returns both values as floating-point numbers to the main program. This in turn calls the `fuzz()` function with them in line 34, which puts on the obfuscation filter (in line 78).

## Math in Space

If you travel any distance on the surface of the earth, you are not, strictly speaking, moving in a two-dimensional space, but on the surface of a more or less even sphere. The distance travelled

from one place to another, which are given as latitude and longitude, therefore cannot be computed by using simple two-dimensional Euclidean geometry, but it has to take into account the third dimension on the great circle of the sphere.

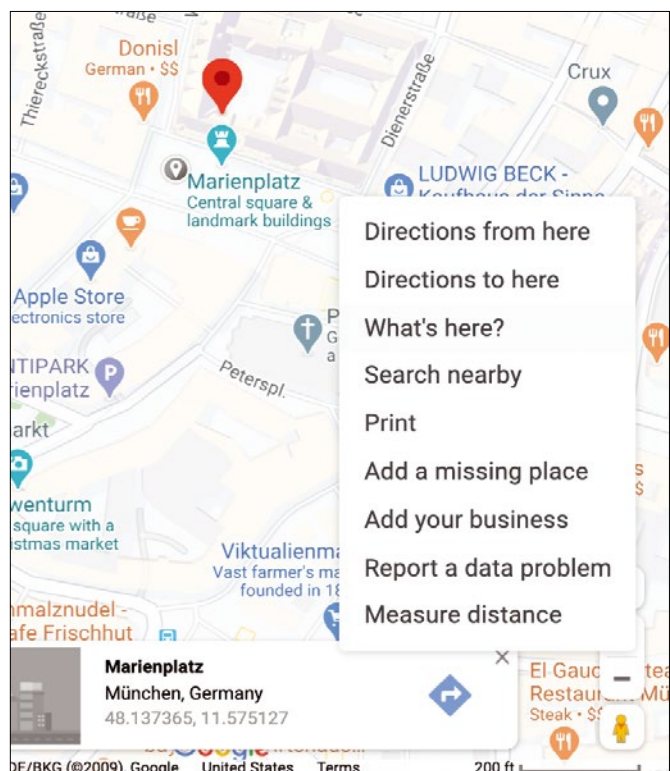


Figure 4: Geocoordinates of Munich's Marienplatz.

The fuzzer therefore has to calculate the distance  $(x, y)$  from the latitude and longitude of a starting point  $(x_0, y_0)$  from which someone moves away in a random direction on the great circle within the radius  $r$ . Fortunately, an expert on stack overflow has already found the solution to this geometric puzzle [4].

The radius  $r$  of the circle within which the algorithm scatters the coordinates is given in meters and not in degrees. To convert, line 80 divides the value of 1,000 meters (which corresponds to a scattering circle with a radius of one kilometer) by 111,300. Where does this constant come from? It corresponds to the distance in meters travelled by someone on the equator who moves exactly one degree. Since the earth has a circumference of about 40,075 kilometers at that point, one degree corresponds to the 360th part of it (i.e., about 111,300 meters).

As far as scattering random points is concerned, it helps to first simplify the assumption that the algorithm places the target points in a two-dimensional circle with the radius  $r$ . With two randomly generated values  $u$  and  $v$  in the range of  $[0, 1]$ , Listing 3 gives the polar coordinates of the move, which can be converted into Cartesian coordinates  $x$  and  $y$  with Listing 4.

Attentive readers may be wondering about the root  $\text{sqrt}(u)$  in the first line in Listing 3 – why doesn't the moving vector length  $w$  simply result from  $r * u$ , creating values evenly between zero and  $r$ ? This is because if the radii  $w$  were distributed linearly between zero and  $r$ , the random points would not be distributed evenly on the circular surface. If half of the points were below  $r/2$ , half of the results would be concentrated on the inner circle area, which contains only a quarter of the entire circular area. The root function corrects this and distributes the points evenly over the entire circular area.

However, the algorithm now has to take into account the fact that the circular surface is not on a two-dimensional plane, but on the globe. On the surface of the earth, radial distances given in degrees at the equator are at scale, but as the globe gets narrower toward the poles, the same segments of a circle get shorter in the west-east direction. After all, a degree in latitude at the equator is a longer distance than a degree further

### Listing 3: Polar Coordinates

```
01 w = r * sqrt(u)
02 t = 2 * Pi * v
```

### Listing 5: Editing Exif Tags

```
01 $ sudo apt-get install exiftool
02 $ exiftool -gpslatitude=37.<xx> -gpslongitude=-122.<yy> file.jpg
```

up or down and shrinks to zero near the poles. A correction formula extends the degree values for the calculated circle's  $x$ -direction (i.e., the determined difference in longitude) for regions further away from the equator:

```
x' = x / cos(y0)
```

Lastly, the latitude  $y_0$  is given in degrees and not as a radian, but the implementation of the cosine function in many programming languages expects radian values. Therefore, `fuzz()` converts the degrees into radians before applying the correctional east-west expander:

```
x = x / math.Cos(y*math.Pi/180.0)
```

Keep in mind that this method provides only an approximation, but it works well enough for relatively small circles and far away from the polar regions. Truth be told, since we're only dealing with random placements, a simpler approach to the fuzzing problem at hand would also have been possible, but hopefully this excursion shed some light on the fascinating field of globe geometry.

## exiftool to the Rescue

Although there is a `go-exif2` library for Go which reads Exif tags, there is no easy-to-use library that writes new tags or refreshes the existing tags in a JPEG file. This is why Listing 1 resorts to the Unix utility `exiftool`. On Ubuntu, this jack-of-all-trades can be set up via the package manager (Listing 5, first line). The tool expects coordinates in a floating-point format and either creates or refreshes the corresponding Exif tags in a JPEG image file (second line).

The `patch()` function from line 42 of Listing 1 calls the tool from Go via the `os/exec` interface. It takes the name of a program including parameters. To collect the output of the command-line utility invoked, programmers provide a byte

### Listing 4: Cartesian Coordinates

```
01 x = w * cos(t)
02 y = w * sin(t)
```

### Listing 6: Generating the Binary

```
01 $ go mod init geofuzz
02 $ go build geofuzz.go
```

buffer, assigned to the `Stdout` and `Stderr` outputs. `Run()` starting in line 52 invokes the external process, grabs its output, and checks the exit code.

Normally, `exiftool` returns with an exit code of `0`. In this case, the error err in line 53 is equal to `nil`, and `patch()` returns to the caller (the main program) after the work is done. `exiftool` modifies the JPEG file given to it directly and stores a backup of the original in `x_original`, but `geofuzz` is not interested in this, and therefore simply ignores it, leaving it in the same directory.

To generate an executable `geofuzz` binary from the source code in Listing 1, Listing 6 shows the Go commands for initializing a Go module and compiling the code while automatically downloading any dependencies from GitHub. Then you just need to install `geofuzz` in a path listed in your `PATH` variable, and a quick call to `geofuzz file.jpg` quickly drops a privacy-friendly obfuscation layer over the geotags in the file. Curious online snoopers will be furious, because they'll be searching in the wrong part of town! ■■■

## Info

- [1] Exif format: <https://en.wikipedia.org/wiki/EXIF>
- [2] Geolmgr: <https://tool.geoimgr.com>
- [3] "Programming Snapshot: Go program finds photos with nearby GPS coordinates," by Mike Schilli, *Linux Magazine*, issue 225, August 2019, p. 44, [https://www.linux-magazine.com/Issues/2019/225/Programming-Snapshot-Go/\(language\)/eng-US](https://www.linux-magazine.com/Issues/2019/225/Programming-Snapshot-Go/(language)/eng-US)
- [4] Generating random locations nearby?: <https://gis.stackexchange.com/questions/25877/generating-random-locations-nearby>





Usql offers a single user interface for managing multiple database systems

# Databases

Usql is a useful tool that lets you manage many different databases from one prompt. *By Marco Fioretti*

**T**hese days, databases are everywhere, from official Census records to personal music playlists. Linux offers many tools for creating, populating, and querying databases. Some users may even say too many, and this tutorial is an answer to that complaint. I will introduce you to usql, a little tool that is a lifesaver for many users who work with databases.

The most ubiquitous and flexible way to work with any database is in a text-based interface. Inside a client application, you type queries at a prompt. The syntax might vary depending on the implementation of Standard Query Language (SQL) [1] the database is using. Depending on the database type, the client either executes the query directly or, much more frequently, forwards it to a server that actually handles the data. The result of the query is then printed out, usually in a tabular format. Alternatively, you can store sequences of queries or commands in a text file and pass it to the client that will execute them automatically, possibly saving the result to a file.

If you always work with one type of database (for example, only SQLite), you can just choose a client for that specific database and get good at using

it. However, if you frequently switch back and forth between different database clients, each with its own personality and feature set, it can get very confusing. It is a little like having to edit text files all day long and being forced to continuously alternate between the vi and emacs.

Usql [2] is a single database client that works with several different database systems. Although the syntax of the actual queries might vary slightly depending on the database, other commands for operating the client are unified in a convenient way that will simplify your database experience and lower the learning curve for adding new database systems to your repertoire. Usql is a clone of psql, the standard command-line client for PostgreSQL databases. The goal of usql is to “support all standard psql commands and features” but to extend those commands to include other database systems.

At the time I wrote this article, the default usql package included support for most of the major databases you are likely to access from Linux, including PostgreSQL, Oracle, MySQL/MariaDB, SQLite3, and even Microsoft SQL Server. However, usql also has drivers for many

relational, non-relational, and even NoSQL databases. In addition to Linux, usql runs on Windows and macOS, with a modular architecture that facilitates code reuse.

But keep in mind that usql is “an adapter, not an abstractor” [3]. If you try to access ten different databases with usql, you will still need to know all the SQL dialects and how they differ from each other. With usql, however, you will be able to query all those databases in the same session of the same terminal, and you’ll have access to some extra features built into usql, such as syntax highlighting.

## Installing usql

Usql is written in the Go language. The easiest way to use it on Linux is to download the tar archive for amd64 systems from the release page [4] (version 0.7.8 at time of writing). Uncompress the tarball and place the resulting binary file, unsurprisingly called usql, in some directory of your path. For Ubuntu desktops:

```
#> tar xvf usql-0.7.8-2
linux-amd64.tar.bz2
usql
#> sudo mv usql /usr/local/bin
```

Lead Image © Kheng Ho Toh, 123RF.com

## Libraries

If you get a message that says the application can't open a shared object file, you have three choices. The easiest option is to install an older version of usql from the release page [4]; the most future-proof, if possible, is to upgrade your Linux system.

If neither of those alternatives is viable, you may still be able to make the last version of usql run. If you have the required library, but in a non-standard location, you can just add a symbolic link to that library in a place where usql can see it. For example, assuming usql complained because it could not find a library named `libcuc.so.60`:

```
#> sudo find / -name libcuc.so.* 2>/dev/null
/var/lib/flatpak/.../libcuc.so.60
...
#> sudo ln -s /var/lib/flatpak/.../libcuc.so.60
/var/lib/libcuc.so.60
```

The first command searches the whole file system to find files with a name that starts with the `libcuc.so` string and prints their locations. The part after the asterisk redirects all warning messages to `/dev/null`. In this example, there were several versions of that library scattered around the system, but only one with the version number required by usql. As expected, that file is in a non-standard place, brought into the system by some flatpak package. Therefore, the second command creates a symbolic link

named `libcuc.so.60` into the folder `/var/lib`, where usql can find it. If, after this operation, you get the same complaint about other libraries, you can repeat the procedure. Of course, it is up to you to decide if this (ugly) path makes more sense than using an older release of usql or upgrading your system.

Don't forget that, as far as Linux itself is concerned, you could create links called `libcuc.so.80` to any file, including older versions of that same library:

```
sudo ln -s /var/lib/flatpak/.../libcuc.so.59 /var/lib/
libcuc.so.60
```

or newer versions:

```
sudo ln -s /var/lib/flatpak/.../libcuc.so.61 /var/lib/
libcuc.so.60
```

and usql will happily load the linked libraries. What would happen afterwards, however, is anybody's guess. It is certainly possible that using a slightly higher or lower version of some library does not make any real difference for a given program, but do NOT count on it! I am mentioning this trick, which, by the way, you can apply to any program, more as a warning than as an actual suggestion. In any case, whatever you do on your own system, please report the problem to the usql developers!

At this point, you should be able to type usql at a command prompt and start using the program. If your Linux system has a version of some library that is older than what usql expects, it won't run. In that case, if you type usql at the prompt, you will get an error messages similar to the following:

```
#> usql
libcuc.so.60: cannot
open shared object file
```

Of course, many Linux applications fail to execute if the libraries are out of sync. The problem with the usql binary from the website is that the dependencies are not documented. (See the box entitled "Libraries" for more on what to do if your libraries aren't what usql is expecting.)

Another option for obtaining usql is to build it yourself in Go. One benefit of this approach is that, if you build usql yourself, you can customize the executable to contain only the drivers (see the box entitled "Custom Versions of usql").

## Getting Started

Usql provides two sets of commands: a big family of "normal" commands, plus

the internal "meta" commands of usql itself. The normal commands are nothing more than standard SQL queries that you would use in other clients to insert, edit, or fetch data. I won't describe the syntax of those queries in this article because this is not a general introduction to SQL, and the commands vary depending on the database. (You will find many good SQL tutorials online.)

The usql meta commands are all prefixed by a backslash, and the first two commands to learn are the ones that tell you what is available in your usql installation. The `\drivers` command first lists

all the database drivers that were compiled into your copy of usql. The backslashed question mark `\?` lists the available meta commands if typed without arguments. Add the name of a command to learn what it does, or use the `options` and `variable` keywords to view available options and variables.

## Configuration

It is possible to define the general configuration and start-up behavior of usql by writing the meta commands in the `$HOME/.usqlrc` file. Usql will also execute all the commands contained in a file

## Custom Versions of usql

Another way to set up usql on your system is to build it yourself with the Go interpreter. One benefit of building usql yourself is you could create a custom version that supports all and only the databases you really need. Of course, you'll need to set up Go on your system, which could be a complex task [5]. Once you have Go up and running, you can build usql with:

```
#> GO111MODULE=on
go get -u github.com/xo/usql
```

At this time of writing, the result of this operation would be a usql executable that only includes the drivers for Post-

greSQL, MySQL, SQLite3, and Microsoft SQL Server. To add support for other databases, you need to explicitly declare what you want at build time. To include all the drivers for all the databases that usql can talk to, for example, you should add the `-tags all` option:

```
#> GO111MODULE=on go get -u -tags all
github.com/xo/usql
```

You may use the same `-tags` switch to specify single drivers or a predefined groups of drivers – or even to exclude single drivers from the executable. For details, please see the usql website.



passed to it with the `-f` or `-file` switches:

```
#> usql -f some-database-script.txt
```

You might be wondering what happens if the file loaded using the `-f` switch contains commands and settings that conflict with the general `$HOME/.usqlrc` configuration file? This issue is important, especially if you want to prepare reusable usql scripts. Luckily, usql offers an easy solution: just add the `-x` or `--no-rc` option when launching usql at the prompt, and usql will completely ignore (for that session only!) the default configuration file. During an interactive session, you can load and execute a command file as follows:

```
\i FILE
\ir FILE
```

`\i` executes the contents of `FILE`, and `\ir` is similar except it looks for the file in the directory of the current script. `\ir` is helpful because, if you use usql on a regular basis, sooner or later you will end up creating your own library of usql scripts that could be organized in several folders and might even call each other.

## Connecting to Databases

Usql opens a database connection by parsing a string and passing its content to the appropriate database driver. Database connection strings (aka “data source names” or DSNs) can be passed to usql directly on the command line or at any moment during an interactive session. In an interactive session, you can pass all the necessary parameters (driver, database name, etc.) separately via the `\c` meta command. In general, DSNs that connect to multi-user database servers like PostgreSQL or MariaDB have the following structure:

```
driver+transport://user:Z
pass@host/dbname
```

The `driver` part is the name of the driver you wish to use, which corresponds to the type of database, or any of its aliases allowed by usql. When connecting to a PostgreSQL database, for example, the driver may be `postgres` or `pg`, whereas with MySQL, you should use `mysql` or just `my`. The `user`, `pass` (that is, `password`),

and `dbname` components are self-explanatory, and the `host` is the name or IP address of the computer hosting the database. If that computer is the same as the computer where you run usql, `host` should be the same as `localhost`.

The `transport` part is not mandatory, and unless you work with a database that requires it, or with non-default connection protocols, you may never need it.

To connect to a database from the Linux command line, use the following command:

```
#> usql my://marco:mypassword@localhost/Z
customers
```

or, after launching usql, from its own prompt:

```
\c my://marco:mypassword@localhost/Z
customers
```

The steps are slightly different if you want to connect to a serverless, file-based database like SQLite 3. In this case, the DSN has a much simpler structure:

```
driver:/path/to/file-on-disk
```

where the `driver` could be `sqlite3`, `sq`, or `file`, or the command might not specify a driver. Either of the following commands would open an already existing SQLite3 database contained in the single file `$HOME/my-sqlite-db.sqlite3`:

```
#> usql sqlite3:Z
//$HOME/my-sqlite-db.sqlite3
#> usql $HOME/my-sqlite-db.sqlite3
```

The second command will work without errors only if the `$HOME/my-sqlite-db.sqlite3` file already exists and is, indeed, an SQLite3 database. If your intention is to make usql create a new, empty SQLite3 database and save it into a file with the specified name, you must use the first command, which includes a recognizable driver name. In this case, be sure to use the right number of slashes: the `sq` and `sqlite3` driver names want two slashes after the colon, whereas the `file`: name only requires one:

```
#> usql sqlite3://Z
$HOME/my-sqlite-db.sqlite3
#> usql file:/$HOME/my-sqlite-db.sqlite3
```

To close an open connection from inside usql, just type `\Z`. In any moment, you can also verify the parameters of the connection you are using by typing `\conninfo`.

## Editing and Reusing Queries

Once you are connected to a database from inside a usql session, you can communicate with it just as if you were using a native client. Usql also keeps the current query in a dedicated buffer that you can edit and reuse. Use the `\e` meta command to edit the buffer and the `\w` command to write a query into the buffer. Optional arguments for a file name and line number allow you to edit queries saved to disk in previous sessions. The `\p` command shows the buffer contents and `\r` resets the buffer.

Use the `\g` command to re-execute the query in the buffer. If you also want to execute every value of the result, type `\gexec`. The `\gset` command stores the result of the query in usql variables for further reuse.

## Output Formatting

A nice feature of usql is its many options for how to print the result of a query. You can print in CSV, JSON, or table format, with either plain text or HTML. Print to CSV or JSON with the `-c` and `-j` options. The options for printing to tables are too many to describe, but they are mostly self-explanatory. `\H` toggles the HTML output mode and `\T`, followed by a string of valid HTML table attributes, will apply the attributes to the current table.

## Variables and System Interaction

One thing I particularly like in usql is its capability to interact with the system it runs on. In Linux, you can type `\!` to open an interactive shell. Typing `\!` followed by a shell command will just execute the command and then return you to the usql prompt. Usql also lets you set and use environmental and also internal variables. For example:

```
#> \set MY_NAME 'Marco'
#> \set MY_NAME = 'Marco'
#> \unset MY_NAME
#> \setenv NUMBER_OF_CUSTOMERS Z
FOUND_CUSTOMERS
```

**Listing 1: Shell Script Pseudo Code**

```
01 usql -f myusqlscript_1 -C -o USQL-OUTPUT.csv
02 usql -f myusqlscript_2
03 echo "The number of customers is $NUMBER_OF_CUSTOMERS"
```

In the first line, the `\set` meta command defines a variable and assigns a value to it. Without parameters, it shows names and values of all the previously defined variables. The `\unset` command, of course, deletes the specified variable. The command on the last line is, in my opinion, the most powerful, because it assigns a value (in this case, the current value of an internal variable called `FOUND_CUSTOMERS`) to an external variable. This feature is an efficient way for a usql script to pass what it finds in a database to whatever other script had called it. The snippet of pseudo code in Listing 1, which may be a part of any Linux shell script using usql to interact with databases, sums up the two methods:

In Line 1, usql executes the queries found in the file `myusqlscript_1` and writes the result, in CSV format, to a file called `USQL-OUTPUT.csv`. In Line 2, usql executes the contents of the file `myusqlscript_2` without creating any file. However, if the file `myusqlscript_2` contains this combination of usql commands already:

```
\gset FOUND_CUSTOMERS
\setenv NUMBER_OF_CUSTOMERS 2
FOUND_CUSTOMERS
```

then the shell script of Listing 1 will find the result of the work done by usql inside `$NUMBER_OF_CUSTOMERS` and will reuse as needed (see line 3). Of course, no data exchange method between usql and shell script is better than the other.

Most of the time, an output file is a more efficient way to store and pass around lots of output, and environment

variables may be more convenient to exchange one or a few values.

Like normal shell scripts, usql

meta commands can use back ticks, that is, inverted single quotes, to assign the output of a command to a variable. At the usql prompt, or in a usql file, you may write statements like the following:

```
\echo Welcome `echo $USER`
Welcome marco
=>
```

but you may also combine this feature with the `\set` meta command to save values obtained from the shell in some variable:

```
=> \set MY_NAME `echo $USER`
=> \echo :MY_NAME
marco
```

The last `\echo` statement introduces one final feature of usql: variable interpolation. Once you have created a variable and assigned a value to it with `\set`, you can substitute its value to its name in composite statements, just like you can in shell scripts. All you have to do is prefix the variable name with a colon:

```
=> \set CURRENT_CUSTOMER marco
=> \set CURRENT_TABLE sales
=> select * from :CURRENT_TABLE 2
where "name" = ':CURRENT_CUSTOMER';
=> \p
=> select * from sales 2
where "name" = 'marco';
=> \g
```

Usql can handle variable names and interpolated strings – either alone or enclosed in single or double quotes, depending on the SQL syntax of the current

database. As long as you place a colon before the name and any quotes, usql will understand that what follows is a variable name that should be replaced with the value of the variable before executing the query.

**Conclusion**

Usql is a useful tool that lets you interact with several different database systems from a single interface. You can also run scripts and access Linux command line programs. You can even run scripts that let you generate usql command files on the fly. If you work with database systems, I hope this tutorial will encourage you to try out usql, because it could save you a lot of time. ■■■

**Author**

**Marco Fioretti** is a freelance author, trainer, and researcher based in Rome, Italy, who has been working with Free and Open Source Software since 1995 – and on open digital standards since 2005. Marco is a Board Member of the Free Knowledge Institute (<http://freenknowledge.eu>), and he blogs about digital rights at <http://stop.zona-m.net>.

**Info**

- [1] Introduction to SQL: <http://www.w3schools.com/sql/>
- [2] Usql: <https://github.com/xo/usql>
- [3] “Usql – A Universal Command-Line Interface for SQL Databases”: <https://news.ycombinator.com/item?id=13780587>
- [4] Usql release page: <https://github.com/xo/usql/releases>
- [5] Installing Go: <https://www.howtoforge.com/how-to-install-go-programming-language-on-linux-ubuntu-debian-centos/>





Securing Internet services on your home network

# Single Source Protection

Mistborn bundles important Internet services on your home network and secures them with a WireGuard VPN tunnel, Pi-hole, iptables rules, and separate containers. *By Ferdinand Thommes*

**C** OVID-19 has forced many people to work from home, relying on Internet services for file sharing, videoconferencing, and more. In addition to outside threats that

exploit the current situation, security risks within the services themselves also pose a problem. For instance, the videoconferencing platform Zoom [1], which has had an influx of users since the be-

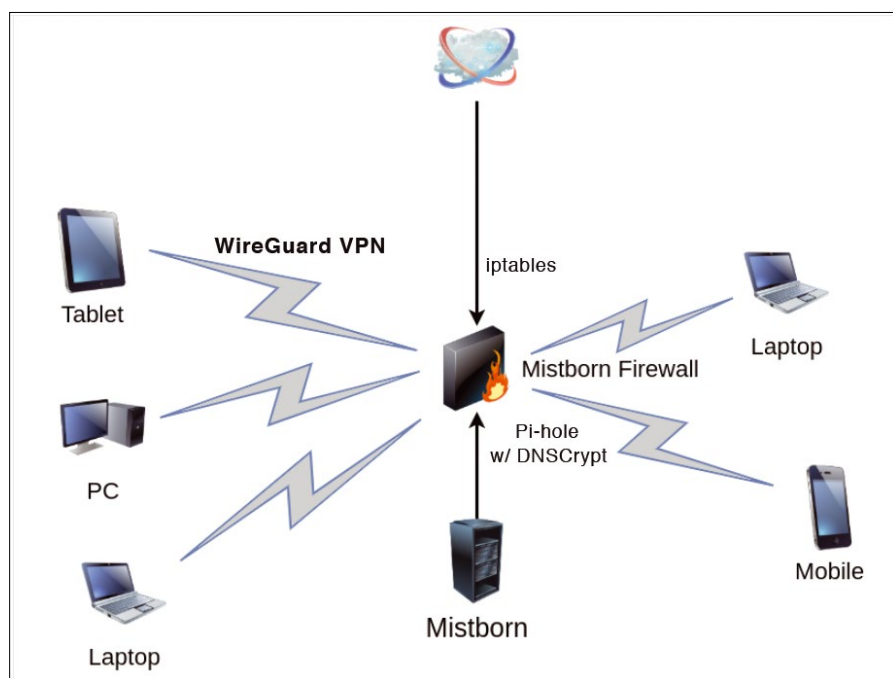
ginning of the pandemic, can hardly keep up with the task of closing its security gaps.

To avoid these pitfalls, Steven Foerster, the developer of security software at Cyber5k, needed an easy-to-implement security solution for all his family's Internet activities. The result is the Mistborn [2] project on GitLab, which is exclusively based on free software. (The name comes from the epic fantasy book series of the same name by Brandon Sanderson.)

Mistborn offers the script-controlled setup of a VPN tunnel with WireGuard, as well as ad-blocking with Pi-hole using DNSCrypt [3] (Figure 1). In addition, Mistborn lets you activate and manage other services, such as Nextcloud, Cockpit, Syncthing, Rocket.Chat, Home Assistant, Jellyfin, Bitwarden, ONLYOFFICE, Tor, and Jitsi.

## Setup

In our lab, we used Mistborn on a Lenovo ThinkPad X220 as the server and a ThinkPad X230 as the client, both with Ubuntu 20.04 LTS (Focal Fossa). You control the software via a web interface from any device.



**Figure 1:** With Mistborn, all traffic runs through a WireGuard tunnel. Access from outside is controlled by the integrated firewall. © 2019, Steven Foerster

Lead image © Nah Ting Feng, 123RF.com

The developer recommends 1GB RAM and 15GB memory for WireGuard with Pi-hole. If you want to use the Cockpit management tool, you need at least 2GB RAM. If services like Jitsi Meet, Nextcloud, Jellyfin, Rocket.Chat, Home Assistant, or ONLYOFFICE enter into play, you need at least 4GB RAM, which also increases the storage space requirement to around 25GB. For videoconferences with Jitsi Meet, 10GB bandwidth is optimal.

Alternatively, you can run Mistborn in the cloud, as long as the cloud environment supports PostgreSQL databases. Options are available for about one dollar per month.

## Static IP and DDNS

If you install Mistborn on a device on your home network, it requires a static IP address [4]. If the services will be available only on your LAN, it is sufficient to give the computer a private static address. This can usually be easily done via the router configuration or the operating system.

The easiest way to obtain a fixed public IP address is to use one of the many dynamic DNS (DDNS) service providers. Some routers, such as the AVM FRITZ!Box, offer DDNS as part of their software.

You can install Mistborn directly or via SSH. If the installation is taking place via SSH, the setup creates an iptables rule that allows future connections via SSH from the same IP address, but blocks all other external connections. The PC continues to accept internal connections via the WireGuard tunnel. The same applies to installation on remote devices via SSH.

All services in Mistborn run in Docker containers. Mistborn automatically sets up and manages the containers for

```

File Edit Index Help
pi@mistborn:~ $ git clone https://gitlab.com/cyber5k/mistborn.git
Clones after 'mistborn' ...
remote: Enumerating objects: 174, done.
remote: Counting objects: 100% (174/174), done.
remote: Compressing objects: 100% (104/104), done.
remote: Total 480 (delta 118), reused 92 (delta 63), pack-reused 306
Receive objects: 100% (480/480), 80.35 KiB | 2.43 MiB/s, Finished.
Resolve differences: 100% (281/281), Finished.
pi@mistborn:~ $
pi@mistborn:~ $ sudo bash ./mistborn/scripts/install.sh
Creating user: mistborn
Adding mistborn to sudoers
/home/pi /home/pi
/home/pi

Wi-Fi is currently blocked by rfkill.
Use raspi-config to set the country before use.

Running as mistborn

          _ _ _ _ _
         / / / / /
        / / / / /
       / / / / /
      / / / / /
     / / / / /
    / / / / /
   / / / / /
  / / / / /
 / / / / /
/ / / / /

(Mistborn) Set default admin password: █

```

**Figure 2:** You can install Mistborn quickly on the server with two commands. After that, you only have to enter a password; the rest is automatic.

you. It also dynamically creates iptables rules as required to prevent external data traffic via the containers.

the script does most of the work leaving you little to do besides reading the out-

## Installation

For the basic Mistborn installation,

### Listing 1: Installing Mistborn

```

$ git clone https://gitlab.com/cyber5k/mistborn.git
$ sudo bash ./mistborn/scripts/install.sh
$ sudo mistborn-cli getconf

```

```

File Edit Index Help
Removing intermediate container 6d599408cfab
--> 4fd4786e4cd
Step 3/5 : RUN touch /etc/traefik/acme/acme.json
--> Running in d4766d42b1b5
Removing intermediate container d4766d42b1b5
--> ce6c17ebcd6
Step 4/5 : RUN chmod 000 /etc/traefik/acme/acme.json
--> Running in 725ab4c37668
Removing intermediate container 725ab4c37668
--> 1674303da5e5
Step 5/5 : COPY ./compose/production/traefik/traefik.toml /etc/traefik
--> 6d5ddeda1fb4
Successfully built 6d5ddeda1fb4
Successfully tagged mistborn_production_traefik:latest
mistborn already in /etc/hosts
address=/mistborn/19.2.3.1
backup up original volumes folder
Created symlink /etc/systemd/system/multi-user.target.wants/Mistborn-base.service -> /etc/systemd/system/Mistborn-base.service.

          _ _ _ _ _
         / / / / /
        / / / / /
       / / / / /
      / / / / /
     / / / / /
    / / / / /
   / / / / /
  / / / / /
 / / / / /
/ / / / /

Watch Mistborn start: sudo journalctl -xfu Mistborn-base
Retrieve wireguard default config for admin: sudo docker-compose -f /opt/mistborn/base.yml run --rm django python manage.py getconf admin default

```

**Figure 3:** The installation took about 15 minutes on the X220 laptop used as the server. However, if the network connection is slow or you are installing on a Rasp Pi, the process may take longer.



put in the terminal as a matter of interest. The first two lines in Listing 1 set up the software. The first line downloads the script from GitLab; the second line starts the actual installation (Figure 2). This takes place below `/opt/mistborn/`.

On the laptop, the process took about 15 minutes on a fast network (Figure 3). For a list of installation steps, visit Mistborn's GitLab page [5]. When prompted to install the resource-hungry Cockpit management tool on a Raspberry Pi, answer *no* unless you absolutely need it.

After the installation success message, wait another minute and then create the configuration with the command from the last line of Listing 1 (Figure 4).

## On the Client

If you are familiar with WireGuard already, you will probably notice the similarity between WireGuard's configuration file `wg0.conf` and the configuration file used on Mistborn. Hence the first step on the client is to install WireGuard. For Ubuntu up to and including version 19.10, the integration of a Personal Package Archive (PPA) is required [6]; you can retrieve the software directly from the Focal Fossa repository using APT. This method also works for many other distributions.

The next step is to copy the configuration file from the server terminal and store it as `wg_admin.conf` on the client in the previously created `/etc/wireguard/` directory. Listing 2 shows an example; after this, start the virtual network interface via `systemd` (Listing 3, first two lines).

If you get an error message with the first command, follow up with the command from the last line of Listing 3. If the output complains that `resolvconf` was not found, just install the `openresolv` package retroactively.

If everything worked, now call up the interface in a web browser on `http://home.mistborn`. Depending on the hard-

```

pi@mistborn:~$ sudo docker-compose -f /opt/mistborn/base.yml run --rm django python manage.py getconf admin default
Starting mistborn_production_postgres ... done
Starting mistborn_production_redis ... done
PostgreSQL is available
# "10.44.147.2" - WireGuard Client Profile
[Interface]
Address = 10.44.147.2/32
# The use of DNS below effectively expands to:
# PostUp = echo nameserver 10.44.147.1 | resolvconf -a tun.%i -m 0 -x
# PostDown = resolvconf -d tun.%i
# If the use of resolvconf is not desirable, simply remove the DNS line
# and use a variant of the PostUp/PostDown lines above.
# The IP address of the DNS server that is available via the encrypted
# WireGuard interface is 10.44.147.1.
DNS = 10.44.147.1
PrivateKey = 0EGCcUphfau942yvQIXcp4d4w6Mo1vmWTZdKoj1DhXY=

[Peer]
PublicKey = kCEyI5M7cEsrs2+CQ2rjzn6wFpMHR/W0aT0ldvc02w=
PresharedKey = +hDQ+TBiYS/TdqaS5mhqQE8iFxrYmc17K5Cv7+Cxwfc=
AllowedIPs = 0.0.0.0/0,::/0
Endpoint = 192.168.178.40:58137
pi@mistborn:~$

```

**Figure 4:** During the test phase, the developer replaced the initially complex command for creating a configuration for WireGuard on the client with a simpler command.

## Listing 2: Example `wg_admin.conf`

```

# "10.15.91.2" - WireGuard Client Profile

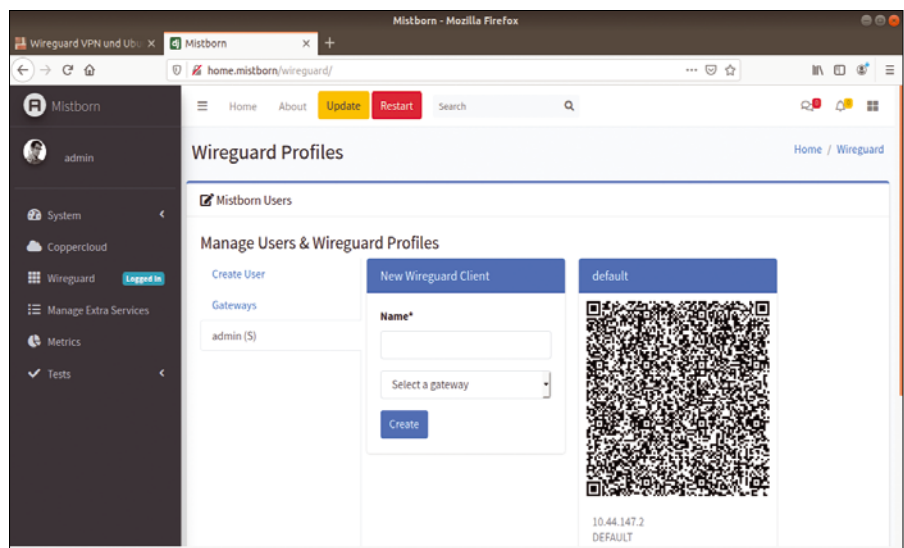
[Interface]
Address = 10.15.91.2/32

# The use of DNS below effectively expands to:
# PostUp = echo nameserver 10.15.91.1 | resolvconf -a tun.%i -m 0 -x
# PostDown = resolvconf -d tun.%i
# If the use of resolvconf is not desirable, simply remove the DNS line
# and use a variant of the PostUp/PostDown lines above.
# The IP address of the DNS server that is available via the encrypted
# WireGuard interface is 10.15.91.1
DNS = 10.15.91.1

PrivateKey = cPPf1VGsxVFW2/1MmhiFTXmMh345bGqoqArD/NgjiXU=

[Peer]
PublicKey = DfIV1urYZXqXKiU4rOSf00Iu589pEO+59dHV5wSN0mU=
PresharedKey = Z1S05NuAnZ7JhzVCuUnYQQLW0QYmMoqG0pG1SNXU1h0=
AllowedIPs = 0.0.0.0/0,::/0
Endpoint = <Mistborn public IP address>:39207

```



**Figure 5:** Immediately after starting the web interface, you can create new users and WireGuard profiles for additional clients or gateways. For mobile clients, simply scan the configuration as a QR code.



**Listing 3: Starting the Virtual Network Interface**

```
$ sudo systemctl start wg-quick@wg_admin
$ sudo systemctl enable wg-quick@wg_admin
$ sudo systemctl status wg-quick@wg_admin
```

ware, it may take a few minutes to connect to the server, as it first has to create the containers.

**Getting Around**

The default view after starting Mistborn is the Profile view where you can create new users, set up a gateway (more about this later), or set up new clients and profiles (Figure 5). Click on *System* in the left-hand sidebar. This takes you to the Pi-hole view (Figure 6) – Pi-hole is enabled by default – or the Cockpit administration interface. All services open in a separate tab.

Next up in the sidebar is *Coppercloud*, which lets you block or grant access to a given set of IP addresses via iptables. Lists entered here are converted to iptables rules at system startup and then executed.

Under *Manage Extra Services* (Figure 7), you will find all the third-party services that Mistborn securely supports. Additional services like the Matrix messenger, GitLab, or various game servers are in development.

All of these services can be set up with the push of a button. As soon as you start a service, a green line appears to inform you that the start-up may take a few minutes. Using a Rasp Pi as the server, it took up to three minutes until a service was ready, depending on the complexity of the application.

Currently you have to update the web page manually to see if the service is ready. After updating, you can start and use the respective application. You only need to start services once. After a restart, you can open them directly.

Finally, you'll find *Metrics* and *Tests* in the sidebar. *Metrics* provides an overview of the firewall's performance, while *Tests* provides port scanning, runs a DNS leak test, and displays the public IP address.

**Gateway**

For services like Netflix that do not work well with WireGuard, you can add a gateway. A gateway is another client that sits upstream of the VPN and

makes proprietary services like Netflix think they are seeing the public IP address of the device running Netflix.

Mistborn does most of the setup for a gateway. As with other clients, you only need to store the configuration created by Mistborn in `/etc/wireguard/gateway.conf` on the client (Figure 8).

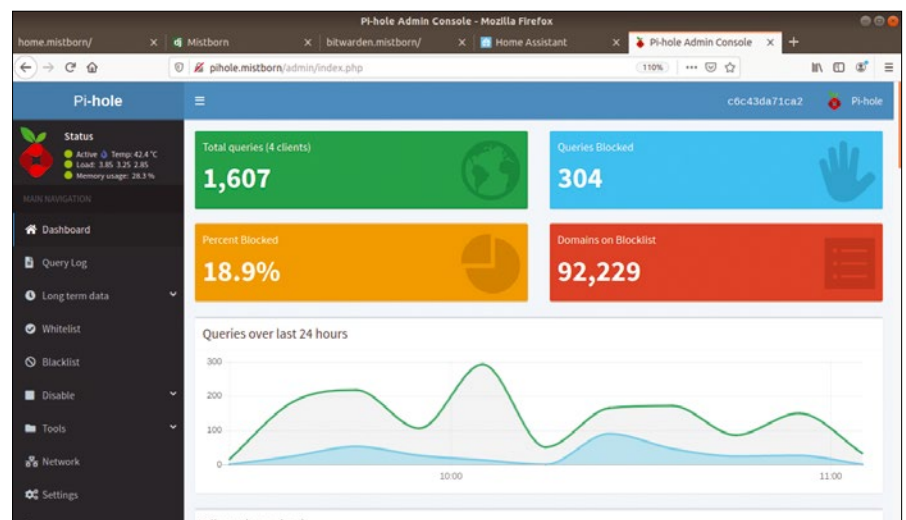
The configuration is created on the profile page below *Gateways*, where you first assign a name. Then press the *Create* button to create a profile, select the profile, and then copy the configuration file.

For mobile devices, you do this by scanning the displayed QR code. The setup for the gateway client is described in the documentation [7]. To get Mistborn running on Android devices, see the “Mistborn on Android” box. There is currently no viable solution for iOS.

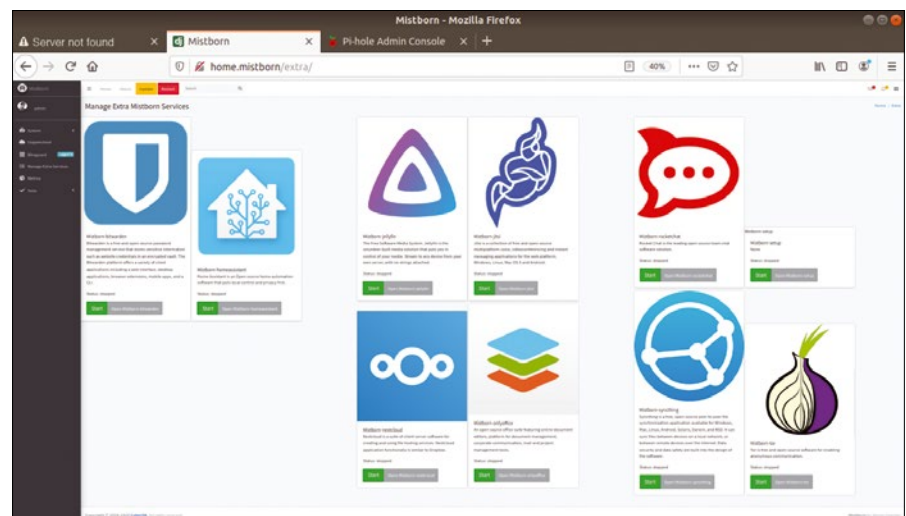
**Security**

Mistborn provides security in several ways. All services run in separate Docker containers. To see which Docker containers are active, use:

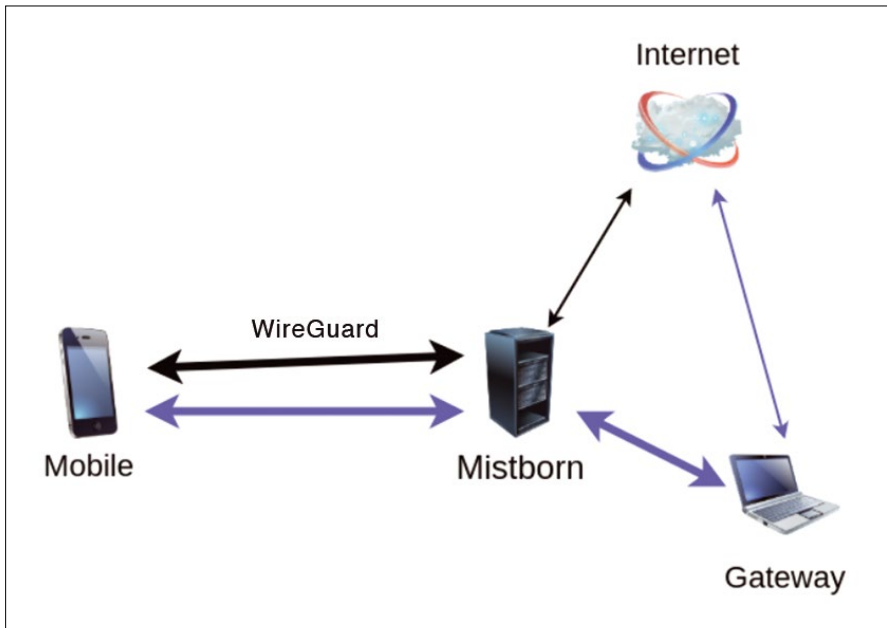
```
sudo docker container ls -a
```



**Figure 6: The server-side tracking and advertising blocker Pi-hole, which is enabled by default, is ideally suited for use on the Rasp Pi. It runs as a DNS server and thus blocks undesirable requests on all devices on the network using lists that have already been included or created by the user.**



**Figure 7: Manage Extra Services lists all the previously installed services. Clicking on Start lets you set up and then start the service in a few minutes. All services work in their own containers and use the WireGuard tunnel.**



**Figure 8:** A gateway is another client that also uses the WireGuard tunnel, but makes proprietary services like Netflix think that there is a direct connection. © 2019, Steven Foerster

### Mistborn on Android

We also tested Mistborn on Android. The procedure is similar to that for other clients. First you create a new client with Mistborn. After you have installed WireGuard on your Android device, open the application and click on the plus sign in the lower right corner. In the menu that now appears, select *Scan from QR Code* and load the configuration directly. After you start WireGuard, you can start Mistborn in your browser.

There is one more hurdle with Android. Some of the services in *Extras* require Transport Layer Security (TLS). To satisfy this request, Mistborn creates a certificate with a 10-year validity period during the installation on the server. You can import this to your Android device by tapping on *Security | Additional settings | Encryption and credentials to Install from store* and import the certificate found at `/opt/mistborn_volumes/base/tls/cert.crt`. The developer has promised a download button for the certificate soon.

Access to Mistborn is only available via WireGuard: The virtual private server service does not respond to unauthenticated traffic.

The `iptables-persistent` package takes care of automatically loading the iptables rules, which are created during the installation of Mistborn and each time Docker is started. A chapter in the documentation [8] provides detailed explanations.

### Conclusion and Outlook

Mistborn impressed us in the test. The idea of securing a home network and managing as many services as possible in a single interface has been professionally implemented. This fairly young application still has some rough edges, but the core features run smoothly. We were able to smooth out some of these rough edges during the test phase by cooperating with Foerster, who was very accessible. He already has plans for further services and other extensions [9].

If you run into problems with Mistborn, check out the Troubleshooting section on the website [10]. If you

can't find an answer there, Foerster will be happy to help you via email at [steven@cyber5k.com](mailto:steven@cyber5k.com).

For testing and for the basic applications (WireGuard and Pi-hole), a Raspberry Pi 4 may be sufficient. If you want to use further services on a permanent basis, you will want to change to more powerful hardware. Using the cloud is a good option, as you do not need the DDNS service there. However, a laptop built in the last 10 years like the Thinkpad X220 from 2011 will do the trick. But no matter what hardware you opt for, Mistborn is definitely worth a closer look. ■■■

### Info

- [1] Zoom: <https://www.cnet.com/news/zoom-security-issues-zoom-buys-security-company-aims-for-end-to-end-encryption/>
- [2] Mistborn: <https://gitlab.com/cyber5k/mistborn>
- [3] Pi-hole: <https://en.wikipedia.org/wiki/Pi-hole>
- [4] Static address: <https://www.ionos.com/digitalguide/server/configuration/provide-raspberry-pi-with-a-static-ip-address/>
- [5] Installation: <https://gitlab.com/cyber5k/mistborn#installation>
- [6] Setting up WireGuard on Ubuntu 18.04: <https://linuxize.com/post/how-to-set-up-wireguard-vpn-on-ubuntu-18-04/>
- [7] Gateway: <https://gitlab.com/cyber5k/mistborn#gateway-setup>
- [8] Security: <https://gitlab.com/cyber5k/mistborn#technical-and-security-insights>
- [9] Mistborn Roadmap: <https://gitlab.com/cyber5k/mistborn#roadmap>
- [10] Troubleshooting: <https://gitlab.com/cyber5k/mistborn#troubleshooting>

### Author

**Ferdinand Thommes** lives and works as a Linux developer, freelance writer, and tour guide in Berlin.

## The sys admin's daily grind: pwquality

# Strong Passwords

Regular password changes are a thing of the past: Strong passwords for each individual service provide more protection. Charly pimped his Ubuntu accordingly with a suitable PAM module.

By Charly Kühnast

Changing the password regularly, about every 60 or 90 days, is now considered obsolete. It is better to use a separate strong password for each service and each login. The requirement for how strong (i.e., how complicated) a password must be is something that – at least on your own systems – you can define yourself.

On my test machine with Ubuntu, I can use almost any simple password I want – that has to change. To make

sure it does, I first have to install the pwquality PAM library:

```
$ sudo apt install libpam-pwquality
```

Then I have to add a line to the `/etc/pam.d/common-password` configuration file. On Ubuntu 18.04 “Bionic Beaver,” the default looks like this (this may be slightly different on other systems):

```
password [success=1 default=ignore] \
    pam_unix.so obscure sha512
```

This line can remain as a fallback, but in front of it – and this is important – I need to insert the line from Listing 1. This is a single line, which I just wrapped for Listing 1 to improve readability. With the individual parameters (Table 1 breaks

them down), the password requirements can be easily controlled.

After restarting the system, the new password rule takes effect. To test it, I changed the password of the user `bob` (Figure 1). In doing so, I intentionally entered a password that was too short in the first round and one that can be found in common dictionaries in the second. The system categorically rejected both – and that's the way it should be.

As my third attempt, I entered a new password that complied with the modified rules: `Cm1.Sya-n`. This seems complicated, but it is mnemonic. It's the first letters and punctuation of the first words of Melville's *Moby Dick* [1], with a `1` instead of an `l`, because I need a digit according to the new password rule. The system accepted the password without complaint. ■■■

**Listing 1: Password Requirements**

```
password requisite pam_pwquality.so \
    retry=4 minlen=9 difok=4 lcredit=-2 \
    ucredit=-2 dcredit=-1 ocredit=-1 \
    reject_username enforce_for_root
```

**Table 1: pwquality Parameters**

Parameter	Meaning
<code>retry</code>	Number of incorrect attempts
<code>minlen</code>	Minimum password length
<code>difok</code>	Number of characters that can match the old password
<code>lcredit</code>	Minimum number of lowercase letters
<code>ucredit</code>	Minimum number of uppercase letters
<code>dcredit</code>	Minimum number of numbers
<code>ocredit</code>	Minimum number of non-standard characters
<code>reject_username</code>	Password and username cannot be identical
<code>enforce_for_root</code>	Rules also apply for root

```
bob@influx:~ $ passwd
Changing password for bob.
Current password:
New password:
BAD PASSWORD: The password is shorter than 8 characters
New password:
BAD PASSWORD: The password fails the dictionary check -
is based on a dictionary word
New password:
Retype new password:
passwd: password updated successfully
```

Figure 1: After the change, the system rejects overly simple passwords.

**Info**

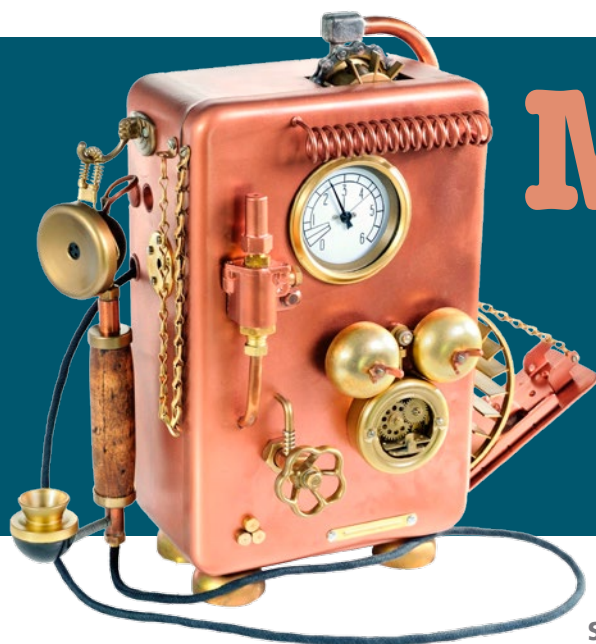
[1] “Call me Ishmael. Some years ago – never mind how long precisely ...”: <http://www.online-literature.com/melville/mobydick/2/>

**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.







# MakerSpace

Overview of the Serial Communication Protocol

## In Series

We explore serial communications, from the electrical specs to protocols and libraries, with an example of serial communication with an Arduino. *By Scott Sumner*

### Telegraphy

The telegraph was the world's first serial protocol and one of the earliest electrical long-distance communication methods. A series of dots and dashes, as in Morse code, represented letters and numbers. Each telegraph station had a telegraph sounder, which was an electromagnet that pulled down an iron bar to make the tell-tale click. The sending device (key) was a large momentary push-button switch with a handle and a shorting bar that would hold the circuit closed when not in use.

The telegraph circuit was a party line, with all of the sounders, keys, and batteries connected in series. When the line was idle, everyone's shorting bars were closed, and all of the sounders were held in the down position (magnets active). To send a message, the operator would open the shorting bar, and all of the sounders would pop up, serving as an alert to each telegraph operator to get ready to copy a message. Once sending was complete, the operator would close the shorting bar, and all of the sounders would click back down, ready for the next station to send a message.

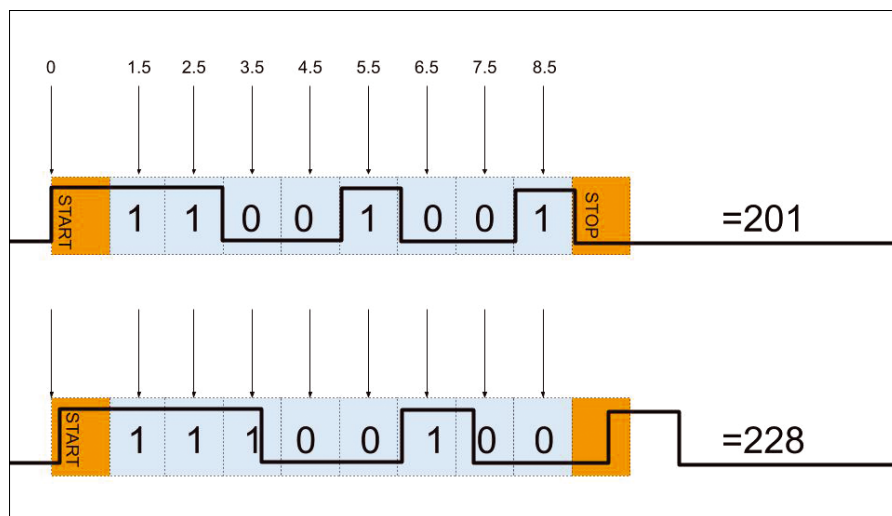
The word “serial” in terms of computers might bring several things to mind. In modern times, it might be the universal serial bus (USB). Not too long ago, it might have brought to mind a 9-pin connector on the back of your desktop – or the even bigger 25-pin connector a bit before that. Different, still, the term might bring up memories of modems, printers, or peripherals connected to specialty computers. If you look at early mainframes, the serial port was the main interface to a

text terminal and thus the human interface to the computer.

Some of these usages have been superseded by newer or different technologies, but serial is still alive and well. Although not as common on today's computer equipment, serial communication is far from gone, and its availability can provide some interesting possibilities for talking to unique hardware.

### Basic Principles

In the simplest sense, an electric circuit used for communication just uses a



**Figure 1:** The signal flow of one 8N1 character on a serial line. The top graph shows proper timing, and the lower graph shows timing that is a little bit off. You can see already, though, how this can change the value received on the other end.

completed circuit to represent a 1 and a broken circuit to represent a 0. See the “Telegraphy” sidebar for how this worked in real life for years. The limitation of a simple telegraphy circuit is that for each signal you want to send, you need a dedicated wire.

The serial protocol expands slightly on the simple telegraphy circuit. It now relies on timing to determine when to see whether the current signal is LOW or HIGH. When you’re setting up a serial connection, you’ll either receive or agree on a couple of parameters that define this timing; for example, *9600,8,N,1* specifies 9600 baud (the speed), 8 data bits, no parity bit, and 1 stop bit, which makes one character of data look like Figure 1.

At a slower speed, say 1 baud, you receive 1 bit every second. Here’s how the sequence works if I’m receiving data: I start by watching for the line to go active or HIGH. The start bit is always 1 and is not a part of the data. When I receive a start bit, I start my stopwatch and wait 1.5 seconds, at which point I’m about halfway through the next data bit. That gives the line time to stabilize electrically before I read it. As each second passes, I look and see what the state of the line is. If it is HIGH, I write down a 1. If it is not active (open circuit, or LOW), I write down a 0. This process continues until all 8 bits are received.

After all the bits for one character are received, at least one bit’s worth of the LOW state will occur before the next character starts. It’s important to note that timing is specific to each character. Nothing in the standard says that characters must arrive in a specific timing. After a complete character is received, the line may be idle for an indefinite period of time.

The parity parameter in the first example was (N)one. The other choices are (E)ven and (O)dd. Parity, which is error checking built in to the protocol, adds an extra bit after the start bit that makes the total number of 1s even or odd, depending on the way the parameter is selected.

Regardless of how you send the bits, the timing is what ensures your data gets from one end to the other.

In the simplest sense, the electrical representation of LOW and HIGH can be line switching between Ground and V+. When V+ happens to be 5V, it is

often referred to as TTL serial. Depending on the equipment (e.g., a Raspberry Pi), this might also be 3.3V.

You’ll also hear terms like RS-232, RS-422, and RS-485 [1] [2]. RS stands for “recommended standard” and defines connectors, pinouts, and electrical characteristics about how these signals are handled. The common RS-232 protocol used in almost all x86 computers for serial communications calls for a negative voltage of at least -3V but no more than -15V to represent a 1 and a positive voltage of at least +3V but no more than +15V to represent a 0, with respect to signal ground.

### Serial on the Desktop

As I mentioned previously, until recently, desktop PCs usually included a 9-pin serial port. Even if it is not brought out to the back panel, you can probably still find a header for one on the motherboard. In the early days of PCs, serial ports were used for input devices like mice or digitizer tablets; to connect to other specialty hardware like scales, cash drawers, or industrial equipment; to synchronize your personal digital assistant; and for a modem to talk to the outside world.

Today, you might have to order a USB dongle to get a serial port on your computer.

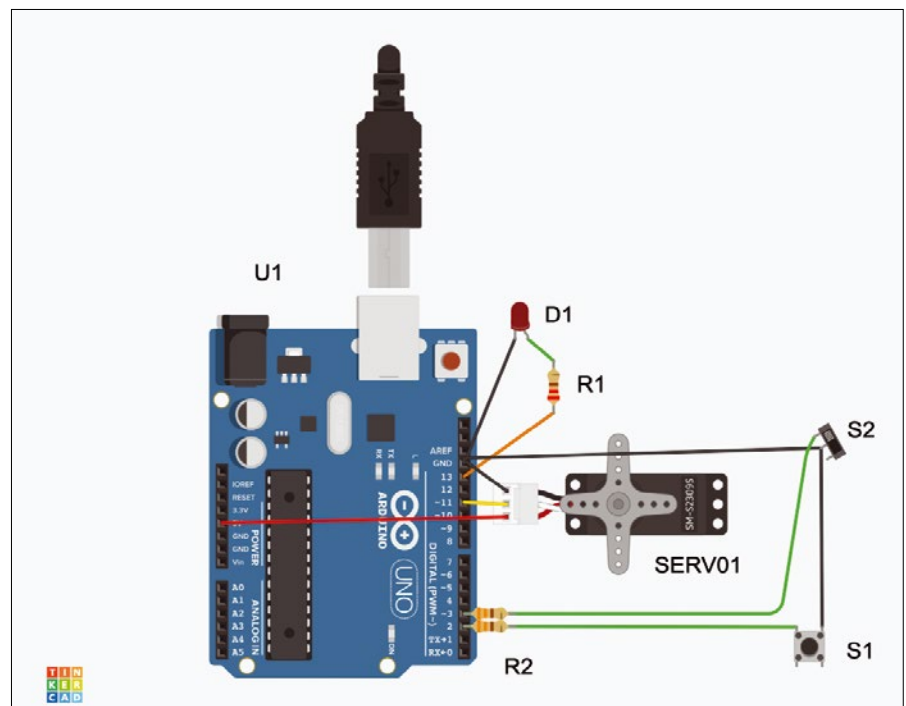
They’re readily available and come in many flavors. The most common is still RS-232, but you can also get them for RS-485 or even more specialty types, like SPI or I2C. The most common USB-to-serial adapter may already be hiding in plain sight on your workbench.

### Serial on an Arduino

Every Arduino [3] with a USB port has a USB-to-serial adapter built in. After you program your Arduino, the serial port can be used to talk with a PC and external devices. I often use this capability to get the best of both worlds. I can connect sensors, buttons, and input devices along with relays, servos, and other custom outputs to the Arduino and use the computer to display a large GUI, talk on the network, or do more complex processing than the Arduino can do on its own. The trade-off is that I have to maintain two sets of

**Table 1: Arduino Parts List**

Name	Quantity	Component
D1	1	Red LED
R1	1	220-ohm resistor
R2	2	330-ohm resistor
U1	1	Arduino Uno R3
SERVO1	1	Microservo
S1	1	Push button
S2	1	Slide switch



**Figure 2: The Arduino wired up to an LED, a servo, a push button, and a slide switch.**

code: one on the computer and another on the Arduino.

With a few parts (Table 1), you can build a very simple serial-controlled device with two input devices, a button and a slide switch, and two outputs, an LED, and a servo (Figure 2). The LED can be on or off, and the servo can move to any position between 0 and 180 degrees.

## Setting Up the Arduino

Notice that the Arduino code (Listing 1) does not have much logic. The only thing it does is respond to incoming messages. In this case, whatever program is controlling the device makes all the decisions, so the logic lives elsewhere. The Arduino just follows the instructions it receives over the serial port and reports changes to the attached inputs.

Lines 1-5 bring in the appropriate library and initialize global variables: `servo` from the `Servo.h` library represents a servo motor, and the integers `oldSwitch` and `oldButton` store previous states of the button and switch.

The Arduino calls the `setup` function (lines 7-16) once when the program is first run. As the name implies, it sets up any Arduino features that will be used and initializes any hardware that needs to be set up before the main loop runs. In this case, it sets up input and output pins and the serial port.

Lines 9-11 tell the Arduino whether the specified pin is used as an input or an output. The first argument is the pin number, and the second argument is the mode. The `INPUT_PULLUP` mode not only sets the pin to an input but also enables the Arduino's pull-up resistor.

Buttons and switches are passive components (not electrically active on their own) so the pull-up resistor provides the electrical equivalent of a default value. The pull-up resistor is relatively weak, so when the button or switch is connected to ground, the pin will see that instead, which guarantees a clean 0 or 1 each time the button or switch changes.

The Arduino Uno has one serial port located on pins 0 and 1. You'll notice, though, that nothing is connected to

those pins. It also has a USB-to-serial converter as part of its onboard hardware, but everything happens behind the scenes, so you don't have to worry about it. I'll be using serial from the attached computer over the USB port.

The `Serial.begin` function tells the Arduino to turn on the serial port (as opposed to being a general-purpose I/O), and `9600` is the baud rate. `Serial.timeout` tells the Arduino that any code waiting for input from the serial port should stop waiting after 5,000msec, which prevents the program from stalling if a proper signal isn't received. Eventually, it will time out and move on with the program.

## Servo Setup

The `servo` variable created in line 3 hasn't been connected to anything yet, so the `servo.attach` line tells the Arduino which pin the servo is on (11 in this case).

The special Arduino `loop()` function starts in line 18 and does exactly what its name implies: It runs continuously until power is removed from the Arduino.

### Listing 1: Arduino Code

```

01 #include <Servo.h>
02
03 Servo servo;
04 int oldSwitch = 0;
05 int oldButton = 0;
06
07 void setup()
08 {
09   pinMode(2, INPUT_PULLUP);
10   pinMode(3, INPUT_PULLUP);
11   pinMode(13, OUTPUT);
12   Serial.begin ( 9600 );
13   Serial.setTimeout ( 5000 );
14
15   servo.attach ( 11 );
16 }
17
18 void loop()
19 {
20   char incoming=0;
21   int readState=0;
22
23   if ( Serial.available() )
24   {
25     incoming = Serial.read();
26     if ( incoming == 'L' ) digitalWrite ( 13 , HIGH );
27     else if ( incoming == '1' ) digitalWrite ( 13 , LOW );
28     else
29     {
30       int i=0;
31       String fullNumber = String ( incoming );
32       String number;
33
34       number = Serial.readStringUntil ( '.' );
35       fullNumber.concat ( number );
36       i = fullNumber.toInt();
37
38       if ( i >= 0 && i <= 180 ) servo.write ( i );
39     }
40   }
41
42   readState = digitalRead ( 2 );
43   if ( readState != oldButton )
44   {
45     oldButton = readState;
46     Serial.print ( "Button:" );
47     Serial.println ( readState );
48   }
49
50   readState = digitalRead ( 3 );
51   if ( readState != oldSwitch )
52   {
53     oldSwitch = readState;
54     Serial.print ( "Switch:" );
55     Serial.println ( readState );
56   }
57 }

```

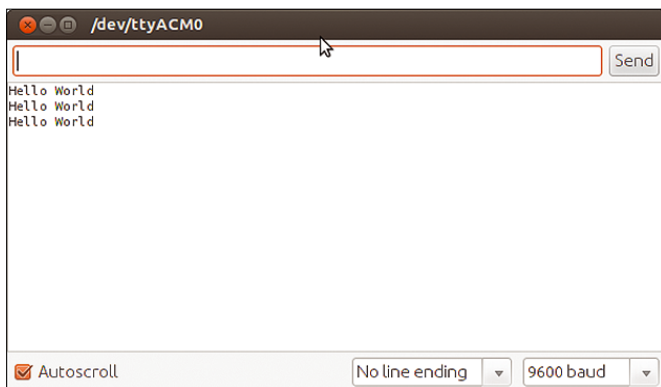


**Listing 2: Python Serial Transmitter**

```

01 import serial
02 import time
03 ser = serial.Serial('/dev/ttyAMA0') # open the Arduino serial port
04
05 while 1:
06     ser.write('L') # turn the LED on
07     time.sleep ( 1 ) # wait a second
08     ser.write ( 'l' ) # turn the LED off
09     time.sleep (1)
10 ser.close() # close port

```



**Figure 3:** The Arduino serial monitor lets you communicate with your Arduino over its serial port; select the speed from the lower right drop-down box.

After initializing two local variables, `incoming` and `readState`, a checking routine looks for any characters that are available from the serial port. `Serial.available` returns the number of characters waiting in the buffer. If it is not zero, the processing inside the `if` statement proceeds.

The `Serial.read` gets the first character from the serial port and saves it in the `incoming` variable. If the incoming character is an uppercase `L`, a call to `digitalWrite` turns the LED on; otherwise, the program checks for a lowercase `l`, which turns off the LED.

By line 28, if nothing has happened, the Arduino has probably received a number, which is processed in lines 29-39. After initializing the local variable `i` to 0, lines 31 and 32 set up strings for the incoming data from the serial port. The `fullNumber` variable is given an initial value of the string version of `incoming` – the first character received (i.e., the first digit).

The variable `number` is initialized to a blank string. Line 34 requests characters from the serial port until it receives a period. The received characters go into `number`, which is concatenated to `fullNumber` in the next line. Now `fullNumber` really

does have the full number received from the port. Line 36 uses `toInt` to turn the string into an integer and stores it in `i`.

Finally, if `i` is greater than 0 and less than 180 (the valid range for a servo), `servo.write` asks the servo to move to the angle specified in `i`.

## Buttons and Switches

Variable `readState` stores the state of Arduino pin 2, which determines whether the button is pressed or not. Note that a value of 0 is pressed and 1 is not pressed, which might seem backward.

If `readState` is different from `oldButton`, the button has changed state. In that case, line 45 updates `oldButton`, line 46 sends the string `Button:` out the serial port, and line 47 sends the new state of the button; `Serial.println` then tells the Arduino to add a newline after printing the `readState` variable.

Checking the switch works exactly the same as checking the button, except on pin 3 (line 50) and by checking and updating `oldSwitch` (lines 51 and 53) instead of `oldButton` and printing `Switch:` instead of `Button:` (line 54).

## Testing the Code

With the Arduino's serial monitor, you can enter data directly to your Arduino. After your sketch is running, open the Serial Monitor by clicking its icon. Be sure to select 9600 baud in the lower right corner (Figure 3); then, you can

type `L` (uppercase) to turn the LED on and `l` (lowercase) to turn it off.

To move the servo, enter a number between 0 and 180 followed by a period. Because the Arduino doesn't know how many digits are in your number, the period lets it know that you're done sending. Sending a number and waiting five seconds also works, because `Serial.setTimeout` (Listing 1, line 13) makes sure the program doesn't stall.

Independent of the output, if the button or switch changes state, you'll get a message in the serial window. If a program were to receive this serial output, it could interpret the messages and take action on the basis of the values, as shown in the Python snippet in Listing 2.

## Controlling the LED

As with any Python program, `import` brings in an external library. In this case, line 1 is the interface to serial ports [4], and line 2 is the time module, which provides the `sleep` function.

The `serial.Serial` function opens a serial port. The only required argument is the port name. In this case, `/dev/ttyAMA0` is the Arduino. Without any other parameters, communication will default to 9600 baud, 8N1 (data, parity, stop bits).

The infinite loop (lines 5-9) sends an uppercase `L` out the serial port (line 6), waits one second (line 7), sends a lowercase `l` out the serial port (line 8), and waits one second (line 9). Although line 10 will never be reached in this example, `ser.close` shuts down a port. Python also takes care of the port if the program terminates.

Once you've programmed the Arduino, run the Python snippet, and your LED should start blinking. Of course, you can now change the timing or send other commands by modifying the desktop program instead of reprogramming the Arduino. This arrangement can be very handy if the Arduino is embedded in another piece of equipment (or suspended 30 feet in the air).

## Listening to Inputs

The Python code in Listing 3 reports the changing states of the button and switch. The infinite loop that starts in line 4 gets a line from the serial port with `ser.readline` (line 5) and splits it at the colon (line 6). To the left of the colon, the string will either be `Switch` or `Button`. If it is `Switch` (line 7) and the

status (right of the colon, `statusParts**[1]`) is 0, then the switch is ON (line 8). Otherwise the switch is off (line 9). Lines 10-12 work the same

way, except it checks and reports the state of the button.

When you run the snippet in Listing 3 and fiddle with the button and switch,

you should see messages announcing the changes in state in the terminal. Although this example just shows a message, it is not hard to see how this could trigger other code.

### Listing 3: Python Serial Receiver

```
01 import serial
02 ser = serial.Serial('/dev/ttyAMA0') # open the Arduino serial port
03
04 While 1:
05     status = ser.readline()
06     statusParts = status.split ( ":" )
07     If statusParts [ 0 ] == "Switch":
08         If statusParts [ 1 ] == "0": print ( "The switch is ON" )
09         Elif statusParts [ 1 ] == "1": print ( "The switch is OFF" )
10     Elif statusParts [ 0 ] == "Button":
11         If statusParts [ 1 ] == "0": print ( "The button is pressed" )
12         Elif statusParts [ 1 ] == "1": print ( "The button is released" )
```

### Serial on a Server

Any time you are connected to a server over SSH, you are using a serial protocol. Today, it is carried by TCP/IP, along with all of your other network traffic, rather than over a dedicated line, but the principle is the same. When you open a terminal window in your favorite Linux distro, you're actually emulating what used to be a hard-wired serial connection.

You can still set up a hardware serial terminal as a console to your Linux machine. The first step is to get it wired to the computer. You probably have a 9-pin serial port, so you'll need a 9-pin serial cable. Most hardware terminals use a 25-pin connector, so you also need a 9-pin to 25-pin adapter, which you can easily find on Amazon or at your local electronics shop.

Once everything is connected, look at the manual for your terminal to find out how to set up the baud rate and other connection parameters, which should be 115,200 baud, 8 data bits, no parity bit, and 1 stop bit.

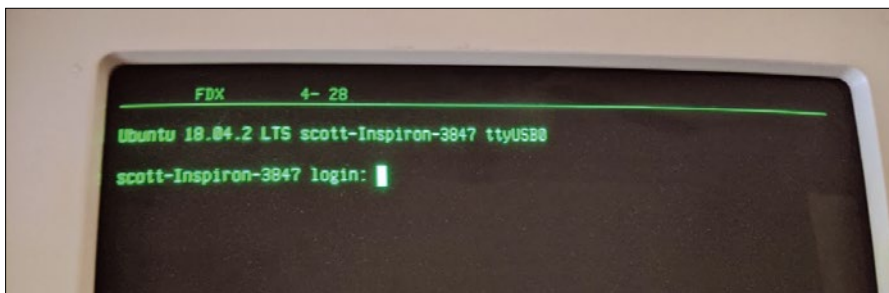
Once you are wired up, you need software to present a shell to the terminal over the serial line. A program called `agetty` should already be available in your distribution or available from the package manager. Open a terminal window on your desktop and type `agetty`. If it says *not enough arguments* you're good to go. The man page shows the command format:

```
agetty [options] port [baud_rate...]
```

See the "Finding Your Serial Port" box to determine your port name; then, enter something like:

```
agetty ttyUSB0 115200
```

If all goes well, a login prompt should appear on your hard-wired terminal (Figure 4). Once you log in, the system's default shell will appear. You can also configure `agetty` to run a custom handler when something connects (e.g., any kind of service you've written), pass the connection off to another program to start a dial-up Internet connection, or anything along those lines.



**Figure 4:** The `agetty` command running on a Wyse serial terminal. Once logged in, this physical terminal will work just like a terminal window.

### Finding Your Serial Port

Linux reports all of its hardware, including serial ports, as files in `/dev`. If your computer has a port on the back of the CPU, then it is probably found at `/dev/ttyS0`. If you're trying to talk to an Arduino, it is probably `/dev/ttyAMA0`, whereas if you're using a USB dongle, it is probably something like `/dev/ttyUSB0`. You can find your port quickly and easily as follows:

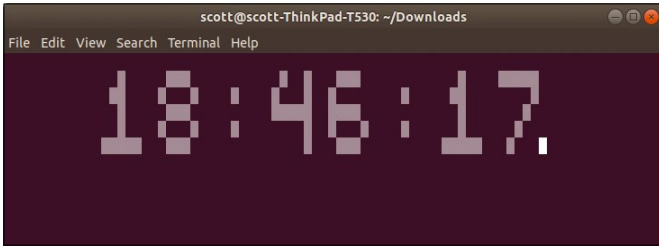
1. Make sure your USB serial port or Arduino is unplugged
2. In a terminal, enter `ls -l /dev/tty*>noUSB.txt`
3. Connect your USB serial port and give it a second or so to register
4. In the terminal, enter `ls -l /dev/tty*>USB.txt`
5. Finally, enter `diff noUSB.txt USB.txt`

You'll see something like:

```
98a99
> /dev/ttyUSB0
```

In steps 2 and 4 above, you're using `ls` to get a list of `tty` devices found in `/dev`. The `-l` requests output of one entry per line instead of output in columns. The `>noUSB.txt` redirects output to a text file. After connecting a USB device, step 4 makes a new list that will include the device that's just been connected, and step 5, the `diff` command, compares the two output files.

The first line of output reports that after line 98 in the original file (`noUSB.txt`) the second file (`USB.txt`) has an additional (a) line 99. The next line of output shows that line after the chevron. In this case, that's the name of the USB serial port that appeared when the device was connected.



**Figure 5:** The Big Print clock running in a terminal window. I didn't have to make any changes to the code between running it on a desktop and in the physical terminal; *curses* took care of all of the code translations, modes, and so on.

## Curses Library

Whether you're using a hard-wired terminal or just a virtual terminal in a window on your computer, this example will work just fine. Because so many different terminals are available with different sizes, capabilities, and options, the *curses* library [5] was developed to handle all of the details in the background, so the developer can just write code and not worry about the capabilities of the underlying hardware. If a particular feature isn't available on a terminal, *curses* will downgrade the features until it can be displayed. If necessary, the output will just be unstyled black and white text.

## The Big Print Clock

A popular thing to do in a serial terminal is display big block text several lines tall. The `bigPrint.py` Python script shows a clock in block text (Figure 5) and is a great example of an application of the *curses* library.

In the Python program in Listing 4, `import` brings in the *curses*, *time*, *os*, *string*, and *datetime* libraries. Note that line 5 is a little different: Rather than importing the entire library, it only imports the `datetime` module. By default, Python imports entire libraries, so an `import` in this format allows you to bring in only the parts you need.

In programming, a `class` is a set of variables and functions that represent a bigger unit. You can create *instances* of classes

that all have the same capabilities but operate independently. If a program were an orchestra, a class could represent each musician.

Here, the `console` class represents a set of functions to display block text on the screen.

The `__init__` function is a special function in a class. It is called automatically whenever the class is instantiated (created for the first time). Similar to `setup` in the Arduino code, it lets you get the class ready for whatever it needs to do. Here I use it to set up *curses*.

Line 9 initializes the *curses* library, and lines 10 and 11 set up how I want *curses* to function overall. The `curses.noecho` function says not to echo (print) characters as they are typed; therefore, any

## Listing 4: clock.py

```

001 import curses
002 import time
003 import os
004 import string
005 from datetime import datetime
006
007 class console:
008     def __init__ ( self ):
009         self.screen = curses.initscr()
010         curses.noecho()
011         curses.cbreak()
012         self.screen.nodelay ( True )
013         self.screen.keypad ( True )
014
015         self.digits = dict()
016         self.digits [ '0' ] = ""
017     ###
018     # #
019     # #
020     # #
021     ###
022     ""
023         self.digits [ '1' ] = ""
024     #
025     ##
026     #
027     #
028     #####
029     ""
030         self.digits [ '2' ] = ""
031     ###
032     # #
033     #
034     #
035     #####
036     ""
037         self.digits [ '3' ] = ""
038     ###
039     # #
040     #
041     # #
042     ###
043     ""
044         self.digits [ '4' ] = ""
045     # #
046     # #
047     #####
048     #
049     #
050     ""
051         self.digits [ '5' ] = ""
052     #####
053     #
054     ###
055     #
056     ####
057     ""
058         self.digits [ '6' ] = ""
059     ###
060     #
061     ####
062     # #
063     ###
064     ""
065         self.digits [ '7' ] = ""
066     #####
067     #
068     #
069     #
070     #
071     ""
072         self.digits [ '8' ] = ""
073     ###
074     # #
075     ###
076     # #
077     ###
078     ""
079         self.digits [ '9' ] = ""
080     ###
081     # #
082     #####
083     #
084     ###
085     ""
086         self.digits [ ':' ] = ""

```



## Listing 4: clock.py (Continued)

```

087
088 #
089
090 #
091
092 """
093
094 def bigPrint ( self , text , position ):
095     x = position [ 1 ]
096     y = position [ 0 ]
097     originalY = y
098
099     for char in text:
100         if char in self.digits:
101             firstLine = True
102             for line in self.digits [ char ].split (
103                 "\n" ):
104                 if firstLine == True:
105                     firstLine = False
106                     continue
107                 outLine = ""
108                 for char in line:
109                     if char == "#": outLine += chr ( 97 )
110                     else: outLine += char
111                 self.screen.addstr ( y , x , "{0:5s}".
112                     format ( outLine ) , curses.A_ALTCHARSET
113                     )
114                 y += 1
115
116         if y > ( originalY + 4 ): break
117         y = originalY
118         x += 7
119
120     def loop ( self ):
121         looping = True
122         startTime = time.time()
123         nextClock = startTime + 1
124
125         while looping:
126             key = None
127             try:
128                 key = self.screen.getch()
129             except:
130                 pass
131
132             if key == ord ( " " ):
133                 looping = False
134
135             if time.time() > nextClock:
136                 now = datetime.now()
137                 self.bigPrint ( now.strftime ( "%H:%M:%S" )
138                     , ( 1 , (80-56) / 2 ) )
139                 nextClock += 1
140
141         screen = console()
142         screen.loop()

```

characters drawn to the screen have to be put there explicitly. The `curses.cbreak` line tells *curses* to return any key presses immediately, instead of waiting for the Enter key to be pressed.

Sometimes you also might want to check to see whether a key is pressed and continue whether it is or not. *Curses* calls this `odelay`. Here, I have to set this mode explicitly on `self.screen`, the terminal display. In this program, I'm using the entire terminal as a single window, but *curses* supports splitting it into smaller windows and controlling them individually, so each can react to keys differently; therefore, I have to tell it explicitly which window I want to place in `odelay` mode.

The `self.screen.keypad` line tells *curses* to interpret escape sequences coming from the terminal automatically. Special keys like the arrow keys, function keys, and Page Up/Down keys usually transmit an escape and then a key code. In keypad mode, you get a constant instead, which is easier to compare and makes a program more readable.

The rest of the `__init__` function defines the block characters that will be displayed in the Big Print clock. Line 15

creates a dictionary of definitions. The `self.digits` key (which is 0 in line 16) specifies the character being stored. Triple quotes start a block string. Until Python encounters another set of triple quotes, everything will be considered a part of the string. Lines 17-21 are a block of lines and characters that define what should be filled in to draw the character identified in the key.

Each set of seven lines defines the numbers 0 through 9 and a colon. I chose the # character for the blocks, but that choice is arbitrary. The `bigPrint` method (lines 94-114) shows why.

### Big Print

To create the block text on the display, the user calls `bigPrint`, which accepts two arguments: `text`, the string to display, and `position`, a tuple of the upper left corner that marks the starting point at which the block numbers are placed on the screen. (Python classes expect the instance of a class itself, called `self`, as the first argument of each class method. Therefore, it looks like each method takes one more argument than it actually does – even methods that take no arguments.)

Lines 95 and 96 break apart the position tuple and define `x` and `y`. The keen observer will note that the `y` coordinate is decoded from the first position in the tuple, which is the opposite of typical Cartesian coordinates. This is just the way *curses* handles coordinates. If your text seems to be going awry, it never hurts to check that you haven't swapped `x` and `y` at some point. Line 97 stores the starting `y` position in `originalY` which is reset after drawing each character.

Line 99 starts a set of loops to retrieve one character at a time from `text`, the string to display. The next line checks to see if the program knows how to draw this character (i.e., it is defined in the `self.digits` dictionary). If so, the drawing proceeds.

Starting on line 101, if you go back to the digit definitions in `__init__`, the first line is always blank (created by the newline after the triple quotes). That way, the columns of the digits align so they are easier to read (and define). This flag is needed later to discard the blank line.

Each line in `self.digits` is a full-size representation of the digit to be drawn.

Therefore, line 102 (working right to left) uses `split` to divide `self.digit` at each newline. The index `char` comes from the loop on line 99 – the character the program is working with right now. The `for` then processes each character with the next block of code.

If `firstLine` is true, the flag is cleared (line 104) and the program continues. This discards the blank line at the top of the digit.

The `outLine` variable is the string that is drawn to the screen. Lines 107-109 create content by walking down each character in `line` (the big digit definition). If it is a pound sign (hash mark), the code adds `chr(97)` to `outLine`. ASCII character 97 is normally a lowercase *a*, but in “graphics” mode it is a solid block. If the character isn’t a pound sign, the program passes through whatever is in the original string (probably a space).

Line 110 tells `curses` to put it all on the screen; that is, on the screen saved in `self.screen`, it adds a string (`addstr`). The first two arguments are the screen coordinates where the string is to appear (remember, the *y* coordinate is first). The third argument is the string to draw. I use a `format` statement defined with `{0:5s}` to make sure that five characters are always output. The final argument draws with graphics characters (`curses.A_ALTCHARSET`) instead of text. The end of the block has some housekeeping to get ready for the next line and ultimately the next digit by incrementing `y`, then checking to see whether five lines have been drawn. If so, it moves to the next digit and back to the top of the requested coordinates to draw the next digit before increment-

ing `x` by 7. If the program were drawing one character per “cell,” it would only have to move over by one, but because each “cell” in this case is essentially a pixel it has to move over the full width of the character, plus a little to space things out.

## Main Loop

The main loop (lines 116-134) calls all of the functions discussed so far. It starts by initializing `looping` – the flag that signals whether to exit – to `True`. The `time.time` function gets the current time, and `nextClock` is set to `startTime + 1` before the loop starts on line 121.

In the loop, `key` initializes before trying to get a key in a `try/except` block. If a key is available, the `try` succeeds, and `key` contains the value; otherwise, the `except` block runs and does nothing (pass).

Lines 128 and 129 check to see whether `key` is a space. If so, `looping` is set to `False`. The next trip through the loop will terminate because `looping` is the condition checked (line 121) to see whether it should continue or not.

The value of `nextClock` (line 119) is when the clock needs to be refreshed. The loop is running much faster than once per second. So once the current time (`time.time`) is greater than `nextClock`, it is time to update the display. Line 132 gets the new time, and line 133 calls `self.bigPrint` and formats the current time as `hour:minute:second`. Finally, line 135 increments `nextClock` by one second so the loop knows when to refresh the display again.

By keeping track of when updates need to happen in the future, the main loop can keep doing other tasks (e.g.,

monitoring other communications channels, checking the status of a server, etc.) until it is time to redraw the display.

The final two lines run the program. Everything I’ve talked about so far won’t do anything unless it is initialized and called. Line 136 creates an instance of the class in `screen`, and `screen.loop` (line 137) starts the main loop to set everything in motion.

## Conclusion

In this article, I explored serial communications, starting with electrical specs and working all the way up to protocols and libraries. I also looked at different flavors of serial and how they can interact with one another. The next time you need to talk to a unique device or move data to or from a small-board computer like the Arduino, consider starting with these ideas and expanding them into your next project. ■■■

## Info

- [1] RS-232: <https://en.wikipedia.org/wiki/RS-232>
- [2] RS-485 on Wikipedia: <https://en.wikipedia.org/wiki/RS-485>
- [3] Arduino: <http://www.arduino.cc>
- [4] Python serial library: <https://pythonhosted.org/pyserial>
- [5] Python curses library: <https://docs.python.org/3/howto/curses.html>

## Author

**Scott Sumner** is the Assistant Manager of the Charlie Noble Planetarium at the Fort Worth Museum of Science and History. He enjoys using Python to solve as many problems as possible.



# MakerSpace

Control WS2812 LEDs  
with a Raspberry Pi

## Colorful Lights

Control a matrix of WS2812 LEDs with the Raspberry Pi.

By Martin Mohr

**Y**ou are probably familiar with the large colored LED strips in many shop windows. Often these light strips use WS2812 RGB LEDs. Ready-made controllers are available, but they usually have a very limited feature set. In this article, I show you how to control a matrix of WS2812 LEDs with the Raspberry Pi.

### Power Games

The WS2812 is a programmable RGB LED that is driven by a small controller that has a data byte for each of the three basic colors (red, green, and blue). This arrangement makes it possible to display more than 16 million colors with one LED. Each LED has four connections: two for the 5V power supply and two (*DI*, data in, and *DO*, data out) for asynchronous serial data transmission.

The output of one LED can be connected to the input of the next, which theoretically allows any number of WS2812 units to be connected in series. However, at some point, it takes so much time to write the values into the LEDs that they start to flicker. With 1,024 LEDs in series, it is just about possible to supply all the LEDs with data 30 times per second so that no visible flickering occurs.

The signal for driving the LEDs is constructed so that every single data bit is encoded in a complete period of the sig-

nal. The pulse/pause time of the signal indicates whether you are looking at a logical 1 or a logical 0. This method is well suited because a continuously changing signal is less susceptible to interference. Synchronization occurs when a single data bit is transmitted in a period of 1.25µs. Resetting an LED requires a low signal on the data line for at least 50ms.

As you can see, generating a signal for the WS2812 is not easy. The program must be written in a machine-oriented programming language like C. Moreover, very precise timing behavior must be maintained. Fortunately, you can use a library that takes care of generating the correct signal.

Because I plan to use a Raspberry Pi to control the matrix, another minor obstacle is the operating voltage and the signal level of the WS2812 diodes, which need 5V. The Raspberry Pi provides that operating voltage, but the signal levels it generates reach a maximum of 3.3V. If you look up the corresponding values in the WS2812 data sheet [1], you will find that the LEDs only detect signals from 3.6V upward without error, which makes an additional semiconductor necessary to raise the signal level. On the Internet you can find many examples by hackers that have done without this additional semiconductor. It might work, but it might not. The 74HC125 quad buffer/



line driver used in this project costs just 25 cents – well-invested money for assured signal levels.

The library has a setting for the brightness of the WS2812 LEDs. This value should be set as low as possible, because a single LED draws up to 12mA of current at full brightness. For a typical strip with 64 diodes multiplied by 64, this results in a total current of almost 800mA for the Raspberry Pi to provide. This amount of current would overload many power supplies. Also, you should not send that much current through a ribbon cable. If you need the full brightness of the LEDs or want to work with more than 64 LEDs, you have to supply them with sufficient power from a separate 5V power supply.

### Hardware Structure

When setting up the hardware, you have only one point to consider: If you generate the signal with a PWM generator, you need root privileges to execute the program. If you use the SPI interface, the program will run with user rights. You should avoid being the root user as often as possible.

Using the SPI interface (BCM pin 10, header pin 19) to generate the LED signals is a sensible option: Connect Raspberry Pi header pin 19 to the input of the 74HC125 and its output to the input of the WS2812 LED matrix [2].

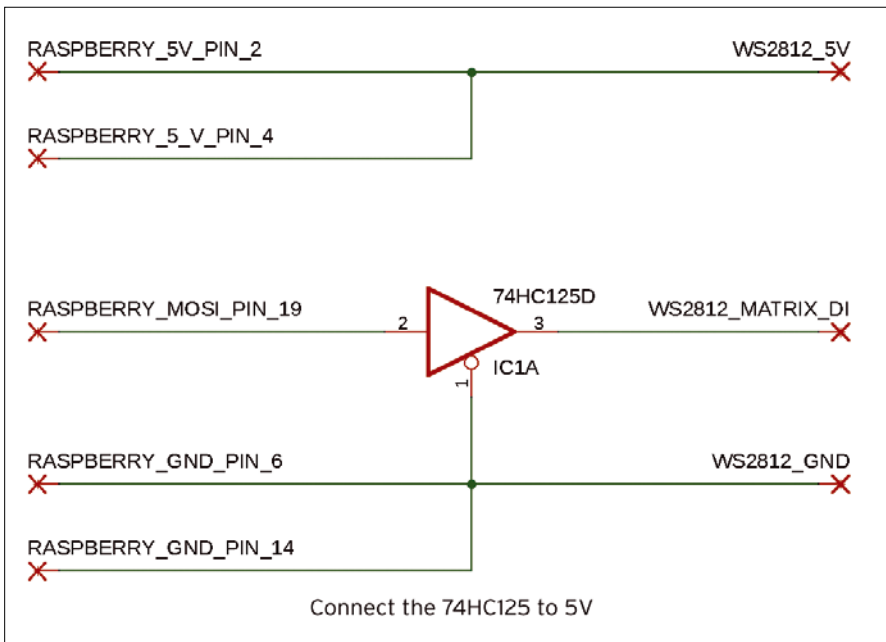
The SMD version of the 74HC125 [3], which you solder on a prefabricated board [4], ensures you do not use too much space. To distribute the current

that flows through the ribbon cable, use both Pi 5V connectors and two ground lines (Figure 1).

To protect the LED matrix and the 74HC125, I 3D-printed a case made of transparent filament for the assembly (Figure 2). The case contains simple stress relief to prevent the thin wires coming off the PCB and the LED matrix. The 3D printer STL and SCAD files can be found on the *Linux Magazine* FTP site [5].

### Preparing the Raspberry Pi

On the Raspberry Pi, you need to use a current image of the Raspberry Pi OS and desktop [6]. Moreover, the installation of the WS2812 driver requires some additional tools and libraries [5]:



```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt install openjdk-10-jdk
scons build-essential libpcre3
libpcre3-dev swig
```

To control the LEDs over the SPI interface, you have to activate them with `raspi-config` in menu item *5 Interfacing Options* | *P4 SPI*.

To save energy, the Raspberry Pi models from the third generation onward lower the CPU frequency, which also changes the frequency of the SPI clock generator. Data transfer to the WS2812 LEDs will not function properly until you set the SPI generator to a fixed frequency by adding the line

```
core_freq=250
```

to the end of the `config.txt` file.

If you want to drive more than 64 LEDs, the SPI buffer might be too small. The buffer size is defined by adding

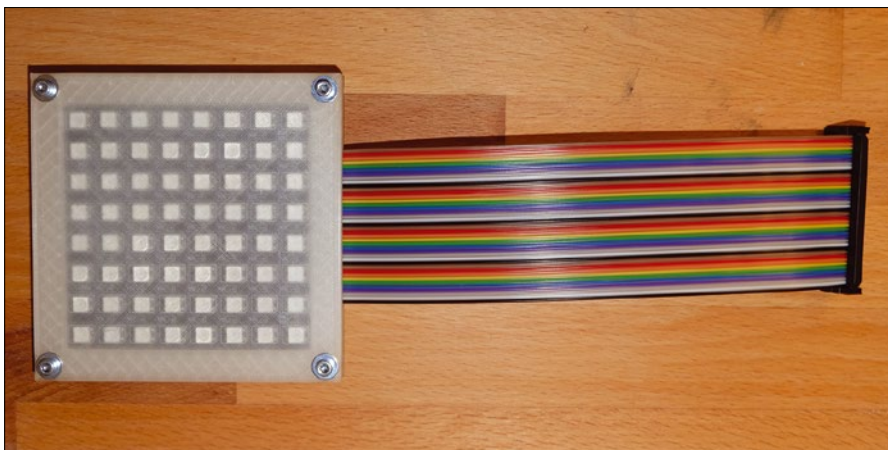
```
spidev.bufsiz=65536
```

to the single-line `/boot/cmdline.txt` file. To apply the changes, restart the Raspberry Pi.

### Installing the C Library

The C [7] library uses BCM notation for the GPIOs, which means that the SPI pin is number 10 in the software (header pin 19). The library is freely available for download online. To install the library on the Raspberry Pi, enter:

**Figure 1:** The circuit diagram for controlling the signal level to the WS2812 LED matrix. The pin numbers correspond to the physical header pins.



**Figure 2:** A 3D-printed housing protects the electronics.

**Listing 1: WS28Test.java**

```
// WS28Test.java
import com.github.mbellings.ws281x.*;
public class WS28Test{
    public static void main(String[] args){
        Ws281xLedStrip matrix = new Ws281xLedStrip(64,10,800000,10,4,0,false,LedStripT
ype.WS2811_STRIP_GRB,false);

        matrix.setPixel(0,255,0,0);
        matrix.setPixel(1,0,255,0);
        matrix.setPixel(2,0,0,255);
        matrix.render();
    } // main
} // class
```

```
$ cd ~
$ git clone https://github.com/
jgarff/rpi_ws281x.git
$ cd rpi_ws281x
$ scons
$ ./test -g 10
```

The last command starts a test program that makes the LED matrix light up in different colors.

**Java Wrapper**

Various programming languages have wrappers for the C library, including Java [8]. A wrapper is a piece of software that provides functions of one programming language by means of an interface to another programming language.

To install the Java wrapper, use the commands:

```
$ cd ~
$ git clone https://github.com/
```

```
rpi_ws281x/rpi_ws281x-java.git
$ cd rpi_ws281x-java/src/scripts
$ bash createNativeLib.sh
$ cd ~/rpi_ws281x-java/
$ ./gradlew assemble -x signArchives
```

To run the wrapper, you also need the Log4J library, which you can download and install with:

```
$ cd ~
$ wget http://apache.lauf-forum.at/
logging/log4j/2.13.1/
apache-log4j-2.13.1-bin.tar.gz
$ tar -xzf
apache-log4j-2.13.1-bin.tar.gz
```

Note that the library must be in the class path during execution.

After completing all the installation steps, test the WS2812 strip with an initial small program (Listing 1) that imports the library for controlling the

**Table 1: Ws281xLedStrip() Parameters**

Parameter No.	Type	Meaning
1	int	Number of LEDs connected
2	int	GPIO connection used (10 for SPI, 18 for PWM)
3	int	Frequency for controlling the LEDs
4	int	Direct memory access to use (10 is typically a good choice)
5	int	Brightness, 0 to 255 (4 to 32 is recommended)
6	int	PWM channel
7	boolean	Signal inversion (if using an inverting level converter)
8	LedStripType	See Table 2
9	boolean	true turns off the LEDs when the program ends

**Table 2: LedStripType**

LedStripType	Possible Types
LedStripType.SK6812_STRIP_<Type>	RGBW, RBGW, GRBW, GBRW, BRGW, BGRW
LedStripType.WS2811_STRIP_<Type>	RGB, RBG, GRB, GBR, BRG, BGR

LEDs; then, to communicate with the LEDs, it creates an object named `matrix`. Table 1 shows the meanings of the individual parameters. The object sets the first three pixels to red, green, and blue, and the `matrix.render()` command writes the data to the LEDs.

To build the Java program, use the command:

```
$ javac -cp /home/pi/rpi_ws281x-java/
build/libs/rpi_ws281x-java-2.0.0-
SNAPSHOT.jar:. WS28Test.java
```

If you do not want to include the class path in the command, you can optionally work with the `CLASSPATH` variable. I chose the notation with the parameter `-cp` for better visibility of the individual libraries.

The command

```
$ java -cp /home/pi/rpi_ws281x-java/
build/libs/rpi_ws281x-java-2.0.0-
SNAPSHOT.jar:/home/pi/
apache-log4j-2.13.1-bin/
log4j-api-2.13.1.jar:/home/pi/
apache-log4j-2.13.1-bin/
log4j-core-2.13.1.jar:. WS28Test
```

starts the program. The `-cp` parameter also passes in all the required libraries.

**Meander**

The `Matrix.java` program (Listing 2) causes all the LEDs in the matrix to flash blue. You can change the output arbitrarily (e.g., to output a Pacman or a ticker).

The LEDs are arranged in a wave on the matrix (Figure 3), which makes it a bit difficult to program them at first. To make life a bit easier, I used the multidimensional array `buffer` to create the animations (line 6). Within the buffer, the individual LEDs can be controlled by reference to their `x` and `y` coordinates.

After making all changes in the buffer (Listing 2, lines 39 and 42), the program transfers the data to the LED matrix with the command `writeBufferToMatrix()` (lines 7 to 25), so you do not have to care about the physical arrangement of the LEDs.

**Conclusions**

The WS2812 LED is a small technical masterpiece. If you combine many elements to create a matrix, you can realize

**Listing 2: Matrix.java**

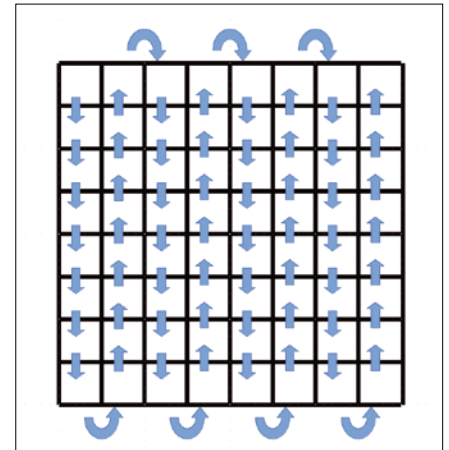
```

01 // Matrix.java
02 import com.github.mbellling.ws281x.*;
03 import java.lang.*;
04
05 public class Matrix {
06     static int buffer [][][] = new int[8][8][3];
07     private static void writeBufferToMatrix() {
08         Ws281xLedStrip matrix=new Ws281xLedStrip(64,10,800000,10,8,0,false,LedStrip
                                Type.WS2811_STRIP_GRB ,false);
09
10         int i=0;
11         for (int x=0 ; x<8 ; x++) {
12             if (((i)/8)%2)==0) {
13                 for (int y=0 ; y<8 ;y++) {
14                     matrix.setPixel(i,buffer[x][y][0],buffer[x][y][1],buffer[x][y][2]);
15                     i++;
16                 }
17             }
18             else {
19                 for (int y=7 ; y>-1;y--) {
20                     matrix.setPixel(i,buffer[x][y][0],buffer[x][y][1],buffer[x][y][2]);
21                     i++;
22                 }
23             }
24             matrix.render();
25         } // writeBufferToMatrix
26
27         static void fill(int r,int g,int b, int a) {
28             for (int x = 0 ; x<a ; x++) {
29                 for(int y = 0 ; y<a ; y++) {
30                     buffer[x][y][0]=r;
31                     buffer[x][y][1]=g;
32                     buffer[x][y][2]=b;
33                 }
34             }
35         } //fill
36
37         public static void main(String[] args) {
38             for (int i=0; i<6000;i++){
39                 fill(0,0,255,8);
40                 writeBufferToMatrix();
41                 try {Thread.sleep(1000);} catch (Exception e){}
42                 fill(0,0,0,8);
43                 writeBufferToMatrix();
44                 try {Thread.sleep(1000);} catch (Exception e){}
45             } //for
46         } //main
47     } //class

```

really great projects. For example, you could display multicolored tickers or even small video animations with these modules.

In the project in this article, I used an 8x8 matrix, but the matrix modules are also available in larger sizes, and several can be connected together. ■■■



**Figure 3:** The LEDs follow a winding course in the matrix.

**Info**

- [1] WS2812 data sheet: <https://cdn-shop.adafruit.com/datasheets/WS2812.pdf>
- [2] 8x8 LED matrix: <https://www.aliexpress.com/item/4001073507941.html>
- [3] SN74HC125: <https://www.mouser.com/productdetail/texas-instruments/sn74hc125d?qs=gqbMQSs93zP7lljTmJMVw%3D%3D>
- [4] PCB for SMD module: <https://www.aliexpress.com/item/32952881938.html>
- [5] 3D printer files and code: <ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/240/>
- [6] Raspberry Pi OS download: [https://downloads.raspberrypi.org/raspbian\\_latest](https://downloads.raspberrypi.org/raspbian_latest)
- [7] LED library: [https://github.com/jgarff/rpi\\_ws281x](https://github.com/jgarff/rpi_ws281x)
- [8] Java wrapper: <https://github.com/rpi-ws281x/rpi-ws281x-java>

**Author**

Martin Mohr has had a fondness for everything that flashes since his early youth, reinforced by an apprenticeship as an electronic technician. After studying computer science, he has mainly developed Java applications, but the Raspberry Pi rekindled his old love of electronics.





**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](http://shop.linuxnewmedia.com/subs)

**GET IT NOW!**

FAST DELIVERY WITH OUR PDF EDITION



The **open source community** is all about trying new things and making small changes – reinventing or adapting to get exactly the software you want. This month, we look at a couple of examples of innovative tools that tweak the mold. First up, you'll learn about a tool that was once called Snowflake and now goes by Muon. Muon/Snowflake brings the benefits of a GUI to remote SSH sessions and also comes with support for secure FTP.

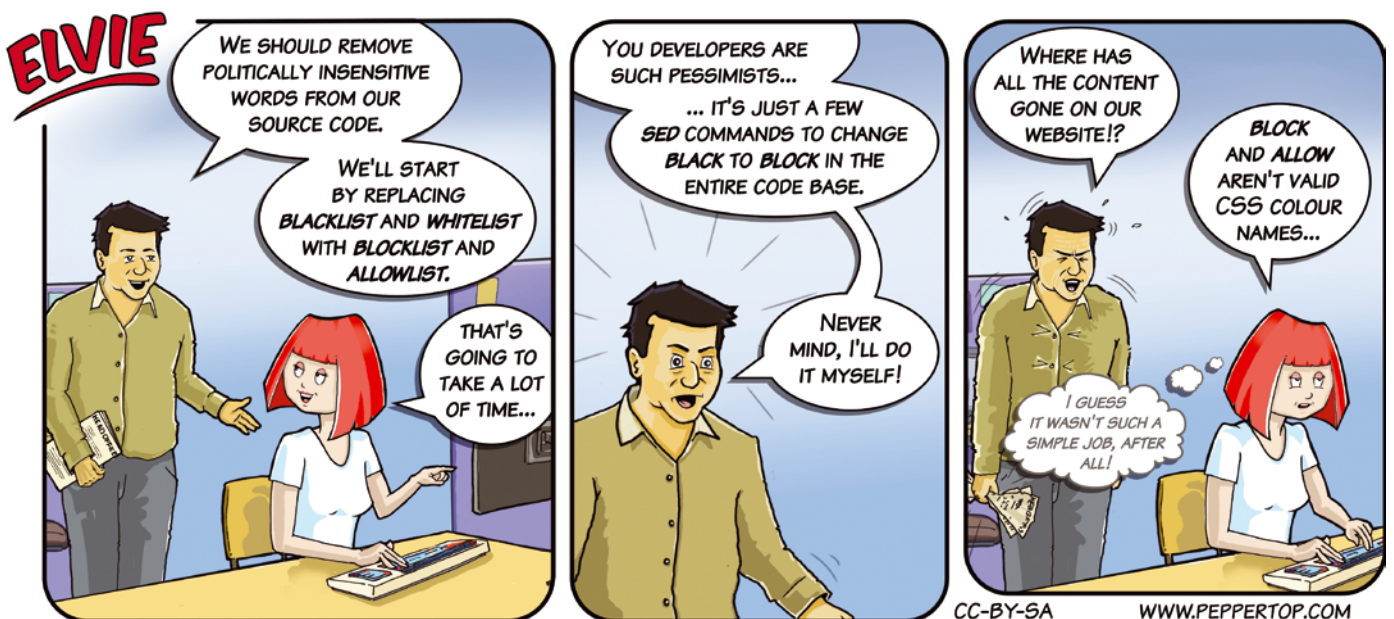
Linux users increasingly rely on messaging and chat systems to collaborate and communicate with colleagues and friends. But why send all that data into the cloud when the person you are trying to reach is in the next room? Also in this issue, we bring you BeeBEEP, an open source office messenger service that keeps it all local. Elsewhere inside, our series on Inkscape continues with a look at how to modify or animate an object with a home-built extension, and editor-in-chief Joe Casad takes a walk through his memories to celebrate 20 years of this magazine you are holding.



Image © Olexandr Moroz, 123RF.com

# LINUXVOICE ▶

<b>Doghouse – Future of FOSS</b>	<b>68</b>
<i>Jon "maddog" Hall</i>	
To build the future of FOSS, we need to focus on communicating its value – especially to young people.	
<b>Muon/Snowflake</b>	<b>70</b>
<i>Ferdinand Thommes</i>	
Muon/Snowflake lets you manage SSH access via an easy-to-use GUI with a wealth of useful functions.	
<b>BeeBEEP</b>	<b>74</b>
<i>Christoph Langner</i>	
This complete chat solution allows you to send messages and share files without relying on the cloud or complicated office infrastructure.	
<b>FOSSPicks</b>	<b>78</b>
<i>Graham Morrison</i>	
This month Graham explores Carla, digiKam 7, NoiseTorch, diskonaut, Surge 1.7, Trigger Rally, and more.	
<b>Tutorial – Tremble</b>	<b>84</b>
<i>Paul Brown</i>	
Writing your own extension for the Inkscape vector graphics app opens up a whole world of possibilities.	
<b>20 Years of Linux Magazine</b>	<b>90</b>
<i>Joe Casad</i>	
Editor-in-chief Joe Casad tells our enchanting 20-year story.	





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

# MADDOG'S DOGHOUSE

To build the future of FOSS, we need to focus on communicating its value – especially to young people. BY JON “MADDOG” HALL

## Old Friends and Future Programmers

Recently three things happened: the celebration of my 70th birthday, a Facebook discussion about songs of the mid-1960s, and the response to a birthday wish.

Being 70 years old has various consequences. I do not run up the stairs as quickly as I did when I was 18. I typically go to sleep earlier and get up later. I take a lot of pills to keep alive – not the fun stuff we may have done when we were younger. I start to look around my home and realize that I have to get rid of some of the things that remind me of my younger days, which brings me to the second topic.

Many of the songs of the mid-to-late 1960s were songs of protest, but also songs of hope and love. We protested what we considered needless wars and sang about making the world better – songs by Dylan; Guthrie; Paul Simon; Peter, Paul and Mary (amazing that after all this time searching for “PP&M” comes up with their names); and other artists. Their songs are still there, and more have followed, but I do not hear people singing them as much.

All of this brings me to the third topic and the reason why I am writing.

I sent a birthday wish to a Facebook person, and he wrote back that he had seen me at many conferences over the past 25 years, but we had never really sat down to talk. He was not blaming me for this; it is just the way things go sometimes. However he lamented that he was “mostly the sole maintainer” of what he considered “one of the building bricks” of GNU/Linux. He was worried that a lot of the other initial “masons of those bricks” were retiring.

I remember in the early days of SourceForge looking at the number of developers of Free and Open Source Software (FOSS) and seeing around one million people registered. Of course those were not *all* of the developers of free software, since some of them maintained their code on other sites and some of the people registered were not developers, but it gave me a gauge of the community. In those days, many of the users of FOSS were the people who understood the community and were either developers themselves or otherwise heavily devoted to the movement.

Just as Steve Jobs realized that Apple should not be a computer company, but a consumer product company, FOSS is moving from a “hacker community” to a “user” community, just as the initial computer industry did 50 years ago.

The number of projects that initially existed on SourceForge were in the tens of thousands, but today on GitHub there are over 100 million repositories and 50 million developers. GitLab is another set of repositories boasting 30 million registered users, 3,000 “active” contributors, and 100,000 organizations. And of course SourceForge is still there with 32 million users.

Of course not all of these repositories are “programs,” and many of those counted as programmers are not programming full time. Some are university students looking for a convenient place to put the sources for their projects.

However a significant number of commercial companies and governments are using FOSS in their businesses, products, and solutions, with downloads (of course) vastly overwhelming the “commits” of code.

None of this is bad, but it could be better.

For a long time good FOSS projects have realized that FOSS developers come and go. They insist that code contributions “fit” with their style of coding so they are easier to maintain in the future.

It takes more than just coding to make a FOSS project; it takes marketing to the end users, and it takes marketing to future programmers for your project. Sorry, but it does, and part of that is developing a warm and welcoming demeanor to your team and documenting why your project is important.

We need more consumers, managers, and governments to wake up to the fact that more and more FOSS code drives the world.

We need more universities to teach their courses with FOSS code, not just in the computer science and engineering courses, but in law (copyrights, patents, and licensing), business administration (business models and plans), and finance.

We need to give encouragement to the professors who want to teach using FOSS and ask our employers to hire college graduates from universities that teach with it.

We need more high schools to also use FOSS and teach using open source software.

We need more and more mentors to take young developers and teach them the basic marketing models of how to earn a good living writing code for solutions that people need, rather than for products that are only delivered closed source.

We need to build the future now. ■■■



# 9 Years of ADMIN on One DVD

# ADMIN

Network & Security

COMPLETE ARCHIVE DVD  
ISSUES 00-49

ISSUE 50/2019

While this ADMIN magazine is not intended to be the basis of our liability, the contents of this magazine and its associated software and services cannot be held responsible for any damage, loss, or destruction of data and computer systems related to the use of this disc.



OVER  
**3,700**  
PAGES OF UP-CLOSE  
SYSTEM  
ADMINISTRATION

Smart and Practical Information  
for Network Professionals



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

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

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

## ORDER NOW!

[shop.linuxnewmedia.com](http://shop.linuxnewmedia.com)

# A convenient SSH GUI Safe Passage

Muon/Snowflake lets you manage SSH access via an easy-to-use GUI with a wealth of useful functions. **BY FERDINAND THOMMES**

**M**any power users believe they can manage connections more effectively from the command prompt, but a graphical interface offers several benefits. Snowflake is a graphically-based SSH terminal that comes with built-in SFTP support. This article describes how to set up secure connections and transfer files with Snowflake.

But first a note: Now that Snowflake has started to gain some traction around the world, the developers have decided to change the name (a time-honored open source prerogative). The tool is now called Muon, but many references to it online, including the official project documentation and even the GitHub page, still refer to it as Snowflake, so I'll call it Snowflake for this article. The documentation will probably update over time, so if you have trouble tracking down Snowflake by the time you read this article, look for Muon and know it is the same tool.

## Basics

Snowflake runs on Linux, the BSD derivatives, HP-UX, and Windows; a version for macOS is under development. You only need to install the software on the client. Snowflake requires at least Java 11, regardless of whether you prefer Oracle's proprietary version or the free variant.

Versions 1.0.3 for Java 11 and 1.0.4 for Java 13 are available on the GitHub page [1]. If you already have Java 13 installed, use the newer 1.0.4 version. The existing Java version is revealed by typing the command `java -version` in a terminal window.

In addition to an MSI installer for Windows and a generic installer for 64-bit flavors of Linux, the pure Java version is also available, as is a deb package for Debian and its offshoots. The packages weigh in at around 25MB each.

### Listing 1: Installation

```
$ sudo chmod +x snowflake-1.0.3-setup-amd64-bin
$ java -jar snowflake.jar
```

Under Debian and derivatives, you can install the package on your system with `apt`, specifying the full path. The package is usually located in `~/Downloads/`, so that the command

```
sudo apt install \
~/Downloads/snowflake_1.0-3.deb
```

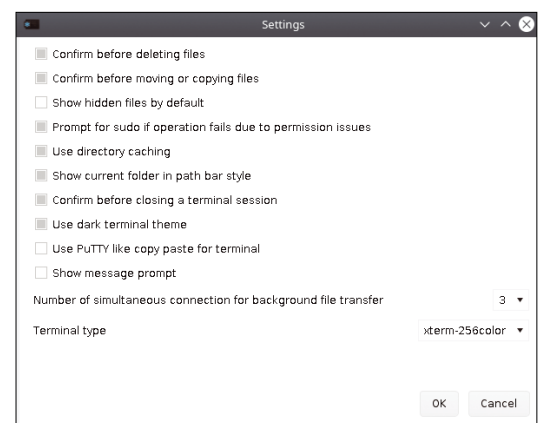
will typically do the trick. The command

```
sudo ./snowflake-1.0.3-setup-amd64.bin
```

runs the generic installer. The application ends up in `/opt/snowflake/`.

In both cases, you then proceed to launch the software in a terminal or from the menu. Depending on the distribution, you may need to give the file the appropriate permissions before starting it. This is what the command in the first line of Listing 1 does; you can start the Java package immediately after the download with the command from the second line. In the test, the program behaved identically for all installation methods.

After the first start, you will see a window with a button for new connections at top left. On the right, there are three buttons that, from left to right, end a



**Figure 1:** The presets in Snowflake offer features that include changing the number of simultaneous connections or requiring a confirmation before deleting files.



session, access the configuration, and provide information about Snowflake. Thanks to useful presets, Snowflake's configuration is quite a short process (Figure 1). To the left of these switches, a menu lets you switch between connections.

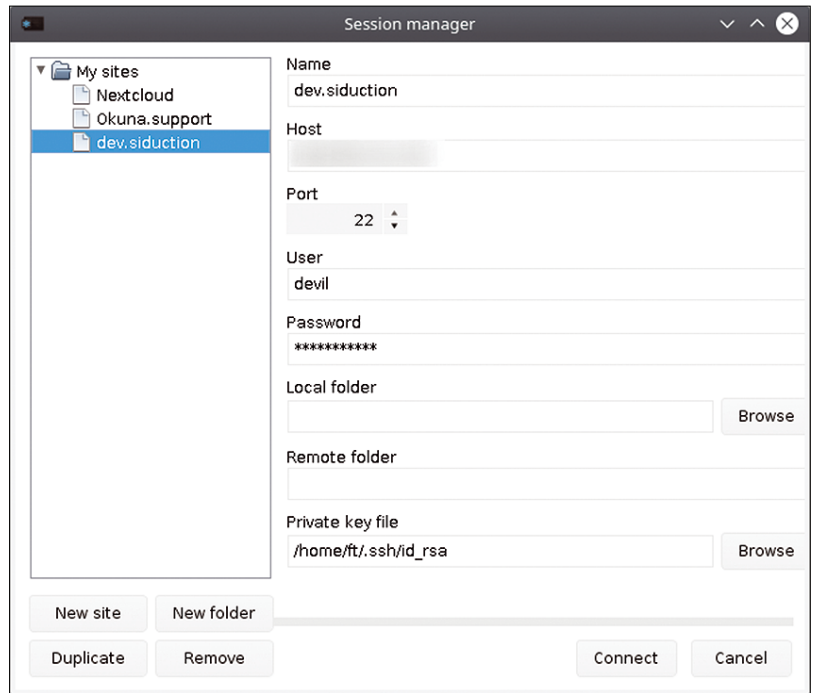
### New Connections

Except for a lot of white space there is nothing else to see at first. A click on *New Connection* opens the *Session Manager*, which follows the same principle as FileZilla. Enter the required access credentials for the remote stations. If required, you can also create folders via *New Folder* to which you can distribute the computers for a better overview.

To enter a remote host, first enter a meaningful freely selectable name and then the IP address in *Host*. Complete the *User* and *Password* fields with the appropriate information. *Local Folder* and *Remote Folder* let you specify the directories that act as the starting point. However, you can leave these fields empty and later manually navigate to the desired starting point (Figure 2).

To authenticate with a key at the remote terminal, enter the private part of the key including the path in the bottom field. In this case, the password field can remain empty, because the software asks for the password for the key before connecting. A click on *Connect* then connects you to the desired computer.

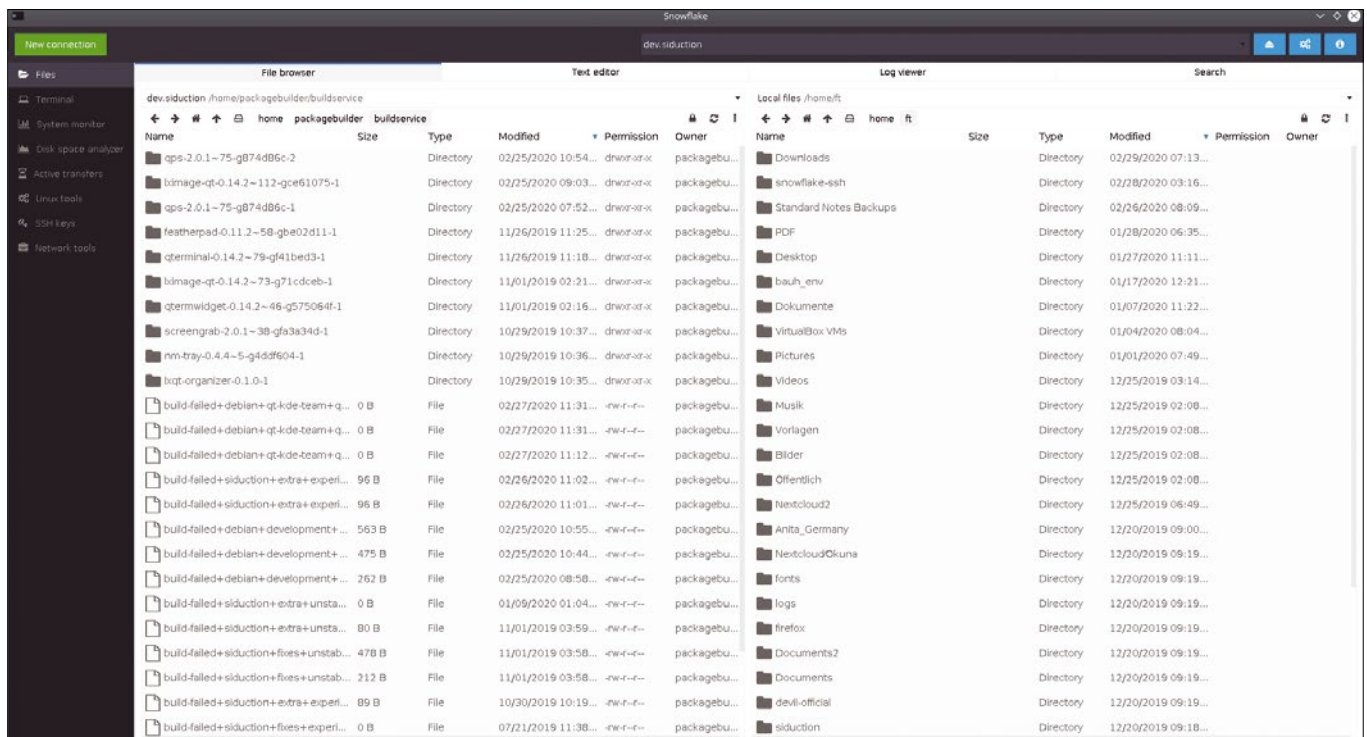
In the test, where all servers were secured with keys, the connection worked in this way on some machines, but not on others. The solution to the mystery lay in the different protocols used to create the keys: Those created using RSA worked,



**Figure 2:** In Snowflake's Session Manager, you enter the credentials of the remote parties you wish to connect to and save them for future sessions.

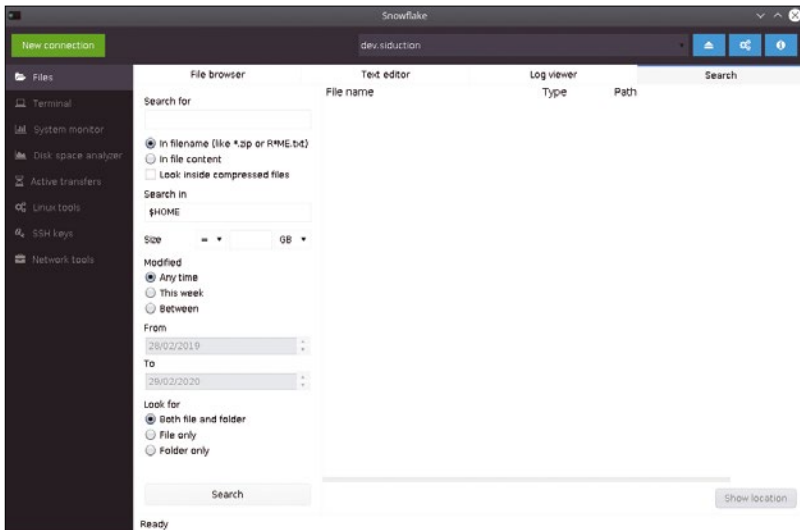
while those with the ED25519 protocol refused to cooperate.

The difference is in the private key's header. For keys generated with RSA, the top line contains *BEGIN RSA PRIVATE KEY*, whereas for ED25519 it contains *BEGIN OPENSSH PRIVATE KEY*. This is currently confusing Snowflake, as can be seen in a bug report on GitHub [2]. Until the Snowflake developers have fixed this bug, you will want to create any new keys with RSA. Converting the key into the format supported by the program did not work in our lab.



**Figure 3:** The file manager located in the *Files* tab shows the selected server on the left and the client on the right.





**Figure 4:** In addition to a file manager, a text editor, and a log viewer, the *Files* integrates a search function based on the powerful Linux `find` command, which offers various options for narrowing down the desired object.

### Clients and Servers

After opening the connection, you will find yourself in the *Files* tab of the SFTP-based file manager. On the left, you will see the directory tree on the remote computer, on the right the tree on the client. In this view, you can navigate through directories via the menu; trigger actions such as copying, deleting, renaming, and moving files; or packing and unpacking archives via the context menu. Changing permissions and downloading files is also possible (Figure 3).

Moving files or directories from the client to the server and vice versa is a simple drag and drop action. For actions with a larger scope, you can run them in the background. In the same tab, besides the file manager, there is a text editor, a log

viewer, and a detailed search function based on the Linux `find` command (Figure 4).

The log viewer is particularly practical, as it opens even logs with a size of several terabytes quickly, making downloads unnecessary. Scripts can be executed via the file manager by opening the context menu of the script file and selecting *Run file in Terminal*.

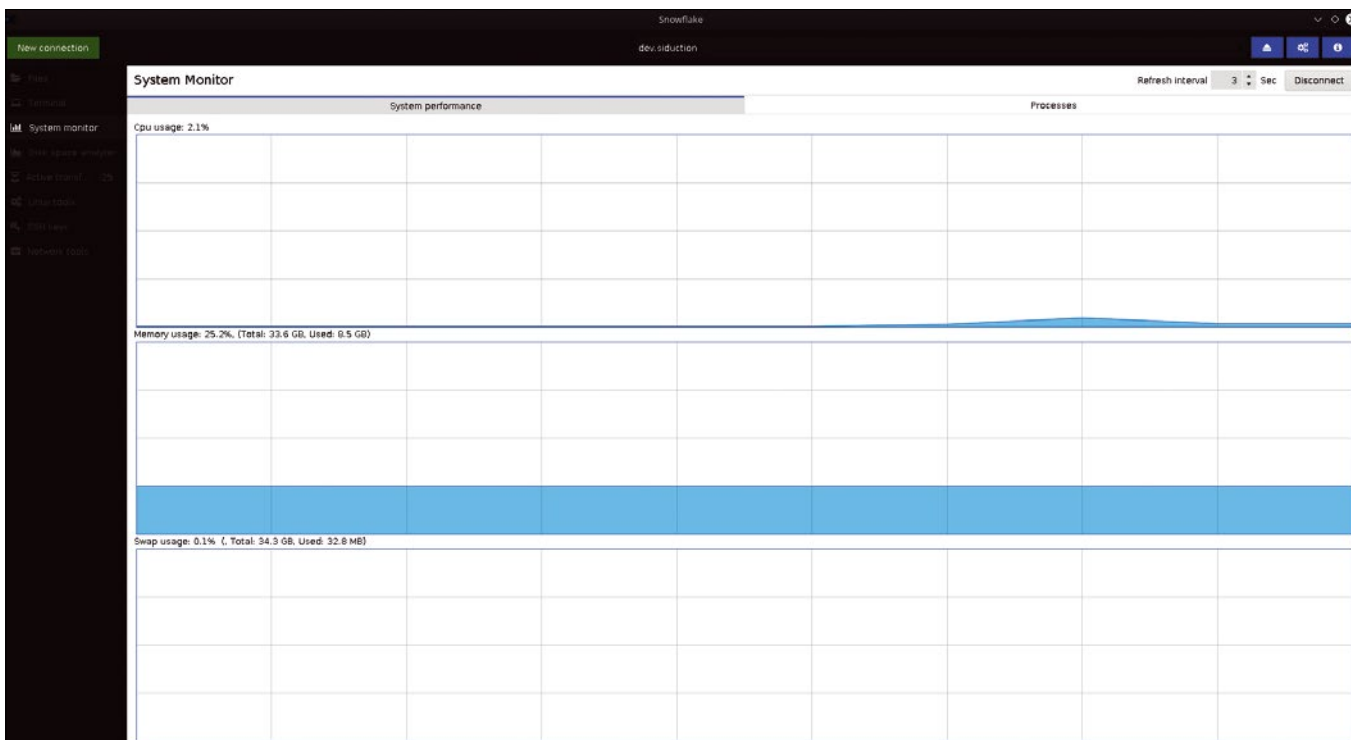
If you want to manage another server instead of the client in the file manager, click on the drop-down menu in the right-hand view that contains the path of the client and select *New Tab* and then *SFTP*. Now connect to another server stored in the Session Manager or to be newly entered. You can then transfer files between the two servers.

You need root rights on the server for some actions. This also works in the file manager if you configured `sudo` on the server in advance. If an action requires root privileges, Snowflake asks if you want to use `sudo`. Even the text editor can call `sudo`.

### Additional Functions

As you may have noticed, the bar on the left edge fills up with several tabs after connecting. In addition to the standard *Files* tab, which contains the file manager among other things, the *Terminal*, *System Monitor*, *Disk Space Analyzer*, *Active Transfers*, *Linux Tools*, *SSH Keys*, and *Network Tools* tabs appear here. Some of them have additional subsections.

You can use a terminal opened on the server in the same way as you would use it on your client. If necessary, you can connect to other servers from there.



**Figure 5:** The *System Monitor* tab shows the CPU and RAM utilization at a glance. Another tab shows the load caused by individual processes.

The *System Monitor* tab (Figure 5) offers a graphical display of the CPU and RAM utilization and also lists all running processes on another tab. A button at the bottom lets you terminate processes directly in this view.

The *Disk Space Analyzer* shows the occupancy of the file system; alternatively you can analyze individual directories. In *Active Transfers* you can see the current data transfers from or to the server. *Linux Tools* takes you to three tabs for system information, system services, and processes listening on ports (Figure 6).

The *SSH Keys* tab shows you the keys on the server as well as on the client and lets you create new keys and move a public key to the server. *Network Tools* takes you to tools such as Ping, Port Check, Traceroute, and tools for DNS lookups.

## Conclusions

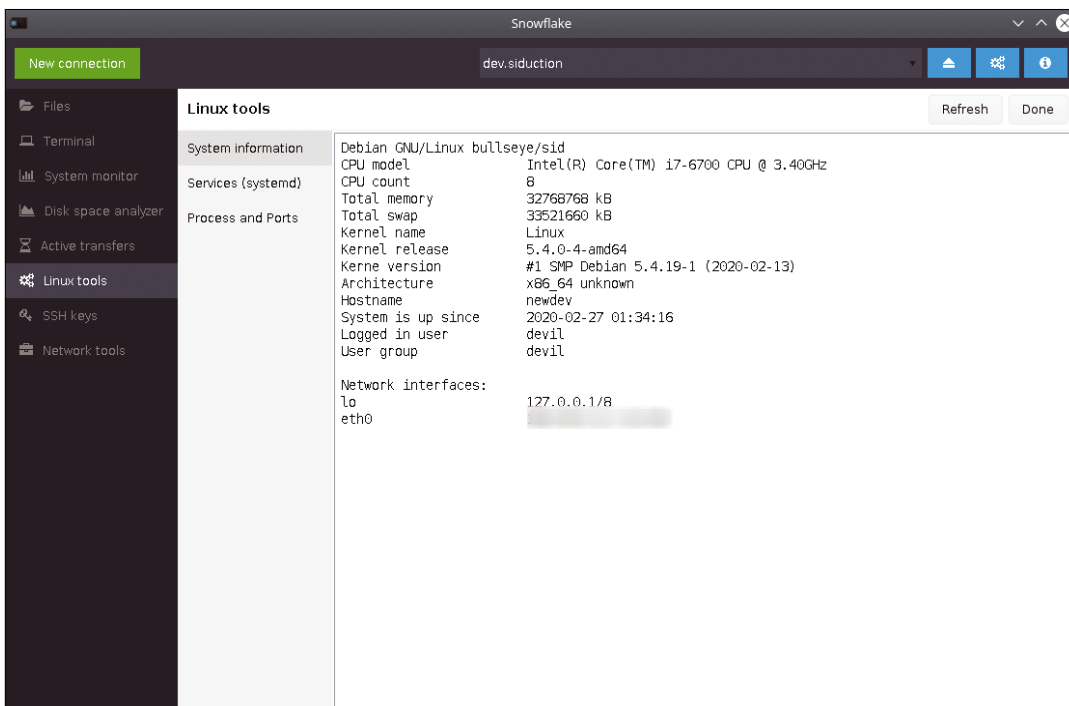
Once you have freed yourself from the dogma that server management has to take place in the terminal, Snowflake SSH proves to be an extremely useful tool. The software combines so many tools under a single interface that the usual

argument that a GUI is slower than working in a terminal does not apply here. The developer's to-do list [3] includes, among other things, integrating a plugin system and support for Mosh [4].

The program doesn't really need any documentation, since almost all functions are intuitively accessible. The developer explains many features on the DEV developer community page [5]. Although Snowflake is only four months old, its agile development still promises some wizardry for the future. ■■■

### Info

- [1] Muon/Snowflake on GitHub: <https://github.com/subhra74/snowflake/>
- [2] Bug report: <https://github.com/subhra74/snowflake/issues/50>
- [3] To-do list: <https://github.com/subhra74/snowflake/projects/1>
- [4] Mosh: <https://mosh.org/>
- [5] Snowflake on DEV: <https://dev.to/subhra74/how-to-make-you-life-easier-on-remote-linux-servers-ssh-g7m>



**Figure 6:** The *Linux Tools* tab displays processes and their ports, a list of system services, and other information about the system, hardware, kernel, and operating system.



# BeeBEEP open source office messenger

## Messenger Bee

BeeBEEP offers a complete chat solution for small businesses or projects, allowing you to send messages and share files within your network without relying on the cloud or complicated office infrastructure. **BY CHRISTOPH LANGNER**

**K**eeping office communication both simple and secure can be a challenge. No matter whether it's email, modern short messages and voice messages, or sharing your desktop, too many communication paths travel through the cloud and across the Internet on paths that are not transparent. The classic email in particular is a dinosaur. Encrypted messages involve undesirable overhead for most users, and even strategically important information often crosses the web without protection.

Solutions hosted in-house are one alternative. A separate Jabber server removes the need for WhatsApp. Your own mail server enables secure communication – at least within your own organization. And there are a number of open source programs that let colleagues share your desktop.

But independence from major vendors and doing without commercial services often requires a great deal of time and work. Among other things, it usually includes configuring, administering, and maintaining servers and software.

That's where small organizations and workgroups may appreciate the simplicity of the open source BeeBEEP program [1]. It makes it easier to

share messages, files, and even your desktop while keeping this information within your network.

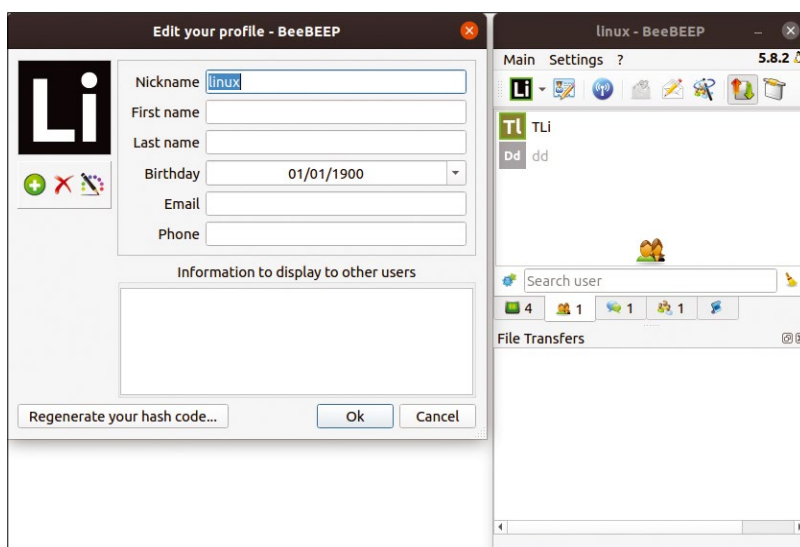
### Ready to Launch

The current version, BeeBEEP 5.8.2, is available for download from the project homepage [2]. The software is available for Linux, Mac OS X, and Windows. The developer also maintains variants for the Raspberry Pi and OS/2.

For Linux, the software is available as a deb package (for Ubuntu 18.04 or newer) and via Snapcraft [3]. You can install the deb package on Ubuntu with a double-click (see also the "Ubuntu 20.04" box). Information on how to import the Snap variant can be found on the Snapcraft project page. Users of Arch Linux can also find the program in the Arch User Repository.

When first launched, the program asks you to choose a nickname. If necessary, you can change this information later using *Start | Edit Profile...*; you can also enter or change other personal information there. After entering your nickname, BeeBEEP is immediately ready to launch. You do not need to configure servers or create accounts. The Messenger automatically lists all other users active on the local network (Figure 1).

**Figure 1:** The contact list is automatically filled with the program users who are active on the local network.



### Ubuntu 20.04

Our test on Ubuntu 20.04 showed that the deb package does not fulfill the dependency on the `libxcb-screensaver0` library. Accordingly, an error message appears when calling `beebeep` in a terminal window. The same thing happens with the statically built variant, which you can access in the form of a tarball. Make sure you manually add the library during the install as follows:

```
$ sudo apt install ./beebeep*.deb
$ sudo apt install libxcb-screensaver0
```



BeeBEEP does not require an Internet connection. All messages and files to be transferred remain completely on your network [4]. The system secures the communication with a 256-bit AES key, so the data on the local network remains protected.

If there is a connection to the Internet, the software checks at startup to see whether a newer version is available. In addition, the developer uses Google Analytics to anonymize the user data. If necessary, you can disable the update check in the settings. However, there is no such switch for analysis of the data by Google.

## Start Chatting

You can start a chat by double-clicking on the icon of the desired partner. This opens a dialog in which you type your message (Figure 2). The tabs at the bottom of the window let you select emoticons, format the text with colors and fonts, and transfer individual files or entire folders.

You will also find an option for sending a screenshot of your desktop as well as a screencast of the current events on the screen – technically speaking BeeBEEP transmits a series of new screenshots (see also the “Wayland” box). You can record a voice message via the microphone icon next to the input field.

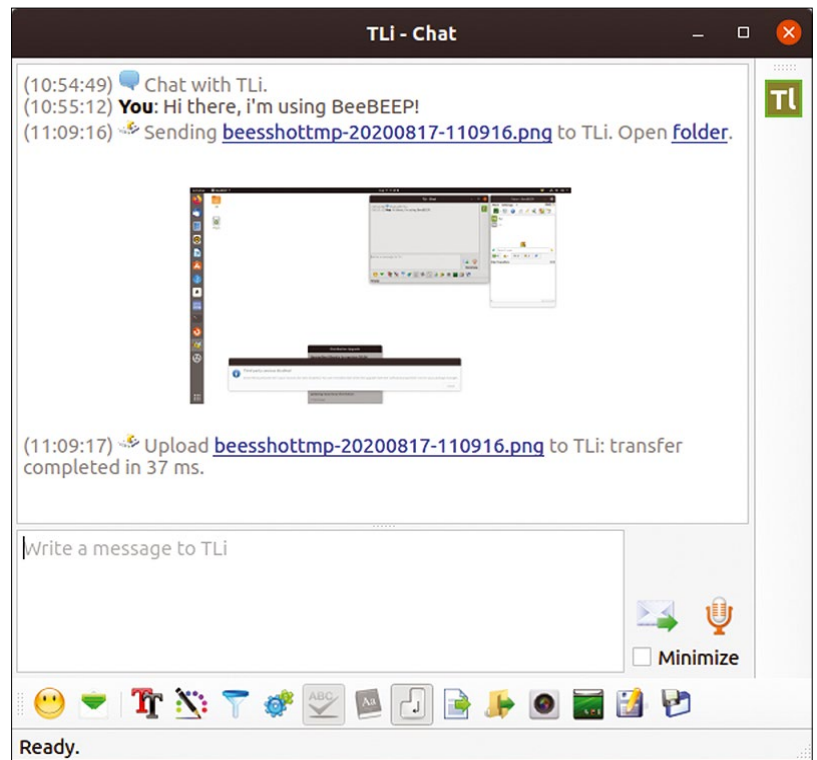
Your contact does not need to be currently online. In the list, the status is shown by the color of the icon in front of the name and the color of the name itself. A gray icon means that the contact’s client is not currently active. However, you can still write a message to this user. As soon as the user starts their client, they will automatically receive the messages sent while they were offline.

In the main window, you will find a row of tabs at the bottom. On the far left, the *Activities* tab shows an overview of what is happening on the BeeBEEP network. In the *Chats* tab, you will find the history of your previous conversations. To start a group chat, press the *Search Users* button to the left of the *Show Options* field, and select *Create New Group Chat* (Figure 3). You can write to each user of the network in a single action using the *ALL USERS* entry.

## Wayland

The successor to the classic X11 display server, Wayland, makes life difficult for users who want to grab screenshots or create screencasts of their desktop – this also applies to BeeBEEP.

If you launch BeeBEEP in a Wayland session in Gnome, you will only see a black image when you grab a screenshot. In this case, you have no choice but to log out of the desktop and select a classic X11 session in the login manager.



**Figure 2:** If the target user is currently offline, BeeBEEP delivers the messages as soon as they become available.

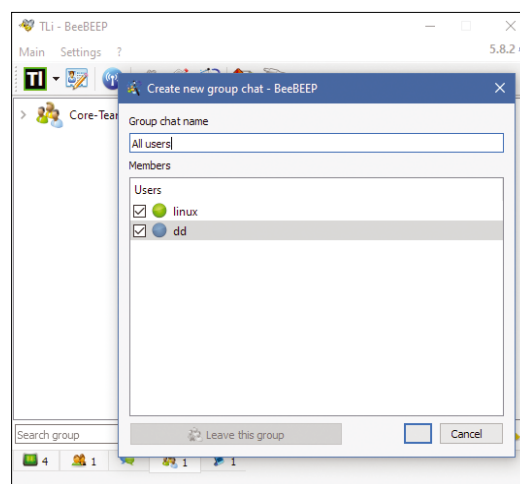
## Sharing Files and Folders

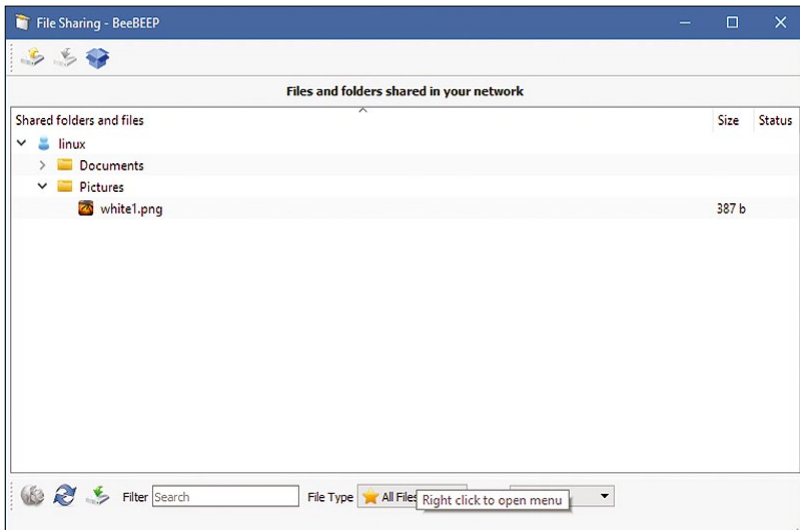
When transferring data, the program goes beyond sending files through the chat. In *Settings | File Transfer*, you can use the *Enable File Sharing* option to transfer files in the style of a network share.

When you enable the function, it activates an icon titled *Show file sharing window* on the far right in the top bar. This is a dialog that lets you share a file or an entire folder with other users. However, the folders should not be too large; the system supports a maximum of just 8,192 files in folders and subfolders.

The other BeeBEEP users on the LAN get access to shared data in the same dialog on your client. To

**Figure 3:** If you have a large number of users, you can simplify messaging by combining contacts into groups.

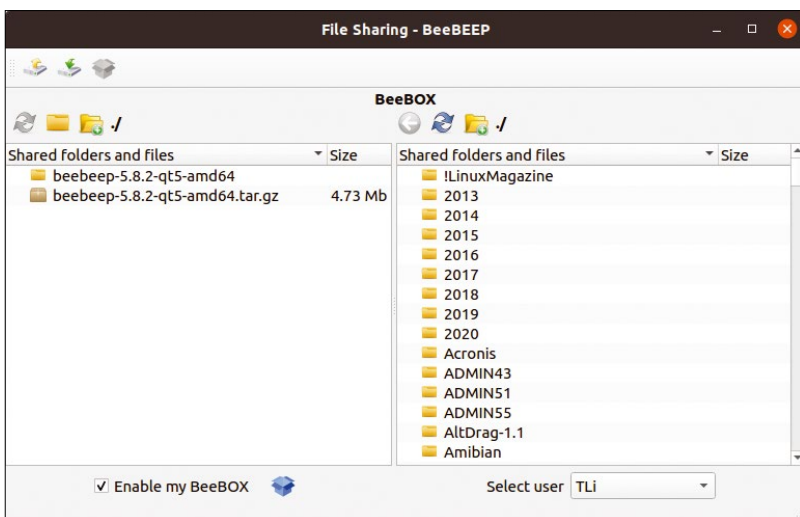




**Figure 4:** No matter which operating system you are using, you can easily access data shared on the local network. The example shows a system with Windows 10.

do this, simply go to the *Files and folders shared on your network* function via the icon in the header of the dialog. You can then look through the shared folders in a file browser and download data by clicking on the corresponding icon in the footer of the data dialog (Figure 4). However, this mode only supports access in one direction: Downloads work;

**Figure 5:** BeeBOX, which can be optionally enabled, lets you write files to a network share on a remote computer.



uploads do not. The system also does not let you edit the existing data.

BeeBEEP offers an extended function for sharing files in the form of BeeBOX. To enable this, click on the blue *Show BeeBOX* icon and then check *Activate my BeeBOX* in the dialog. The software then prompts you to select a folder for the transfer. In this mode, you see a two-column file manager that shows the data in the BeeBOX folder you selected on the left and the data that a contact has shared on the right. You can select this from the drop-down list below (Figure 5).

Once you have selected the remote device, transfers in both directions are possible. In same manner as a normal file manager, simply drag the data to be transferred to the other half of the window.

The program only lets you copy files and folders from A to B. If you try to transfer a file that already exists on the target system, it appends a time stamp to the copied file. It is not possible to delete files on the target computer. This helps to avoid nasty surprises.

### Conclusions

BeeBEEP impresses with its features, simplicity, and robustness. If you are looking for software that makes it easier to collaborate on a project with friends, fellow students, or other employees, you are likely to find the program appealing.

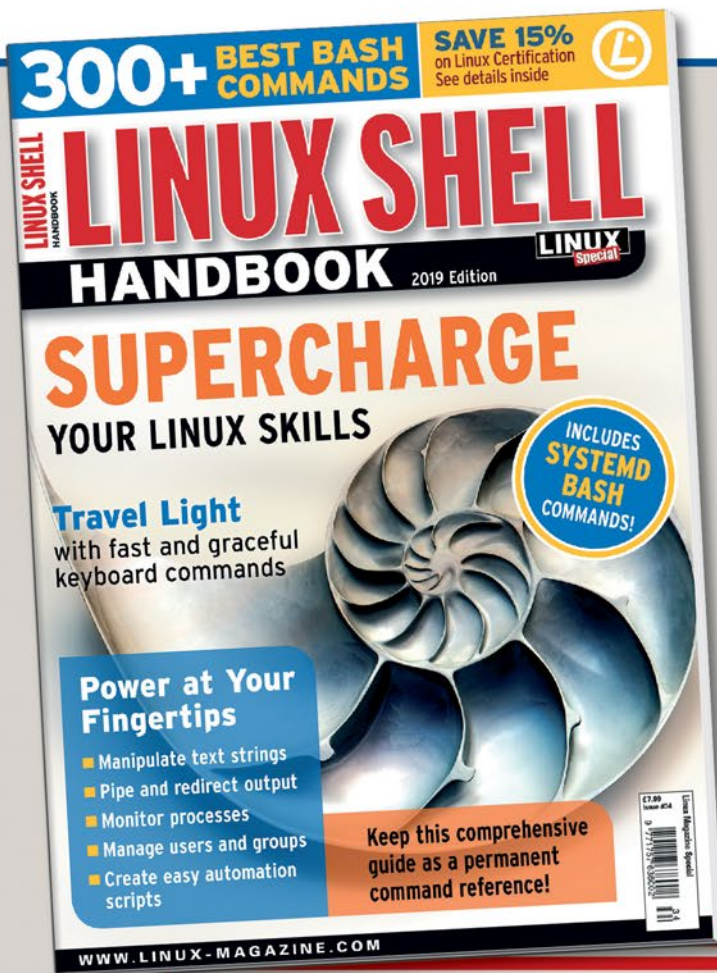
In addition, the program is available for all common PC operating systems, and it does not necessarily need to be installed with administrative privileges – in that sense it is truly portable software. If you want, just copy the archive to your computer, unpack it, and start the application. ■■■

### Info

- [1] BeeBEEP: <https://www.beebeep.net>
- [2] Downloads: <https://www.beebeep.net/download>
- [3] Snapcraft: <https://snapcraft.io/beebeep>
- [4] Frequently asked questions: <https://www.beebeep.net/faq#faq16>

# EXPERT TOUCH

The *Linux Shell Handbook* is available now!  
This edition is packed with utilities  
for configuring and troubleshooting systems.



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.

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](http://shop.linuxnewmedia.com/specials)



# FOSSPicks

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



The release of digiKam 7 has Graham thinking about resurrecting his old photo manager for KDE, Kalbum, last updated in 2004! **BY GRAHAM MORRISON**

## Photo management

# digiKam 7

One of the biggest compromises you make when you forgo the privacy infringement of online photo storage is convenience. Not only is online storage usually limitless and cheap, it's typically combined with the same clever AI-generated search metadata that drives whatever online platform you choose. Google, Apple and Amazon, for example, all silently trawl through your uploaded photos to recognize common items, colors, locations, situations, and of course, faces. This can really help you find a single image in a collection that

might have started over a decade ago, and it's a process that can't be easily replicated on a local machine with local storage and no access to the cloud. Until now.

DigiKam 7 is the latest release of KDE Plasma's flagship photo manager, a photo manager that also happens to work brilliantly on other desktops and (keep it quiet!) Windows and macOS. Its best new feature is one that finally delivers on a function the app has long been experimenting with: deep-learning-powered face recognition. This is thanks to contributions from Google Summer

of Code developers, who first demonstrated that neural networks could improve recognition, before integrating the new deep neural network features now found in the wonderful OpenCV library. It uses a previously trained dataset to discover faces in your photos from just a single example, without any online communication. Faces can then be tagged with whatever name you assign to them. Just like any cloud service, the tags will even show the face as a thumbnail, making it trivially easy to find photos with the same person.

If you've not installed digiKam for a while, a first-run wizard steps you through the database back end (SQLite) before asking where your photos are located and whether you want metadata in the database or photos themselves. Although digiKam has always supported a huge number of image formats, this release adds support for Canon's new CR3 RAW files. This is an important addition when you consider how prevalent Canon cameras are in the open source community, thanks to the fabulous Magic Lantern firmware. One of the best things about digiKam is that despite the huge size of many RAW files the import,

navigation, and loading of photos is typically very fast. Facial recognition can slow this down, but fortunately you can turn on an option to scan new images automatically.

The application itself is still a powerful tool that's better suited to organization than editing. It does have an editor and some excellent color analysis and filters, but it excels at rating, geolocation, tags, notes, album generation, and export. The last of these supports many of the aforementioned online hosting services, as well as some excellent HTML rendering options for hosting your own photo collection. This release adds a beautiful HTML5 Responsive theme to the latter, for instance, which is perfect for dropping onto a server and browsing from a smartphone. For editing, you're still better off using darktable or RawTherapee, but digiKam now seems to concede this point with good grace, accepting its place as a brilliant catalog and library tool and happy to help its users access external applications and services when needed. Thankfully, facial recognition is no longer one of those external services you need to use.

**Project Website**  
<https://www.digikam.org>



**1. Albums:** digiKam is fast, which makes it perfect for organizing your photo collection. **2. RAW support:** Recent versions support many RAW formats, including Canon's CR3. **3. Editable metadata:** There are few photo applications that you can trust to really delve into photo data. **4. Geolocation:** Either update the embedded details, or increase your privacy by storing geolocation details outside of the image metadata. **5. Editing:** While not as powerful as darktable, digiKam's internal editor is perfect for tweaks for social media posts. **6. Ratings:** Tag or rate your photos in the way that works best for you. **7. Light table:** View similar shots side-by-side to choose the best. **8. Tags:** Even face recognition can now be used to help you sort and arrange your photos.

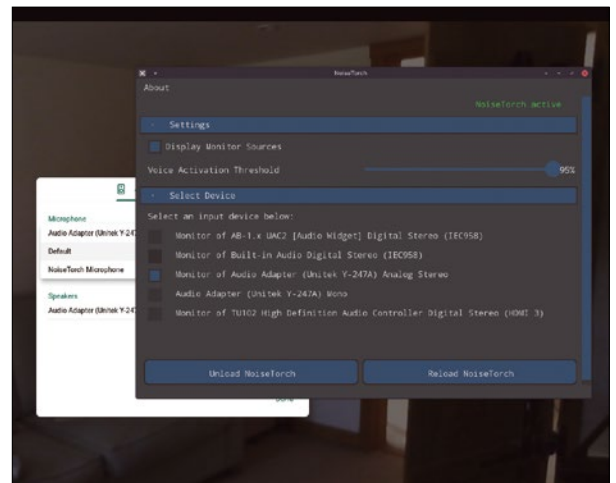
## Noise remover

# NoiseTorch

One of the most common problems we all encounter with remote video calls and conversations is background noise. Even the quietest home office is easily plagued by the sound of fans, air conditioners, children, brewing coffee, and distant traffic. Some chat platforms, such as Google, do offer some kind of noise reduction, but they never give you any control over its parameters. NoiseTorch is a very clever solution to this problem that both reduces noise and gives you complete control over the process. Remarkably, it does this within the confines of PulseAudio. It's an audio processor that sits between your microphone, or audio input, and the application expecting the audio input. This is why, when NoiseTorch is installed, you're first

asked to select an input device to monitor. The UI even helpfully filters out the various display monitor sources that litter so many of our inputs, after which you can launch NoiseTorch proper.

The great thing about using PulseAudio is that NoiseTorch works with any application, because as soon as it's running, PulseAudio thinks it has another input source. This appears as *NoiseTorch microphone* in your applications, but it's really just a virtual input using your previously configured input filtered through NoiseTorch's noise reduction algorithm. Your original input or microphone is also selectable, which is great if you need to quickly revert to an input without the noise reduction. The algorithm NoiseTorch uses is called RNNNoise, a product of deep learning that's



NoiseTorch is a brilliant way of reducing the level of background noise from your microphone (or any other input), without further dependencies.

hosted by Mozilla Research. It's great to see a static open source library being used for something practical, especially when a closed source input filter could so easily be subverted. Other than input selection, there's only a single value to adjust while a call is in progress. This acts as a noise gate, cutting the sound completely when it falls beneath a certain level. You won't directly appreciate the quality of the end result, but your colleagues and family on the other end will.

**Project Website:** <https://github.com/lawl/NoiseTorch>

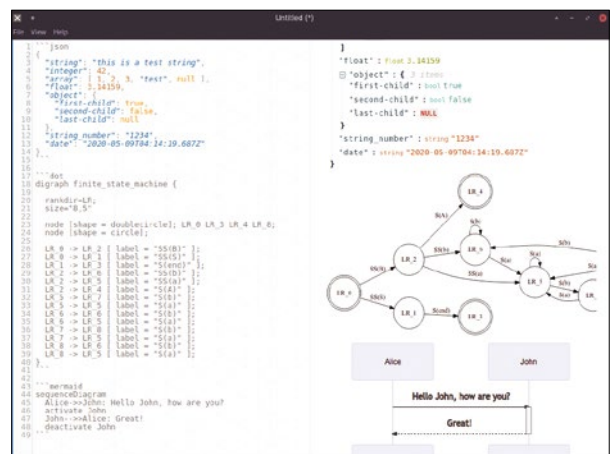
## Markdown editor

# Aurora Editor

We've looked at plenty of minimal Markdown editors in these pages, but the shiny new Aurora Editor is showing plenty of promise. It even lowers expectations by describing itself as "yet another lightweight Markdown editor," which still never fails to win us over. First impressions indicate that there's not that much difference between Aurora and the average editor, however, with the UI being even more minimal than most. The main view is split horizontally between a left pane and a right pane, and the left pane is where you do your work and actually edit the text. This pane has a simple geometric texture on its background that can't be changed, but it does put you in a scientific frame of mind

while writing. You can select between a dark and a light theme from the View menu and toggle full-screen mode. These are the only configurable options, but it doesn't mean Aurora isn't comprehensive in its editing ability.

The right pane shows a preview of the HTML rendered output. This is easy for text, but what makes Aurora unique is its ability to render more than the simple HTML prerequisites of lists, emoji, and tables. You can include mathematical formulas written in MathJax, which are rendered perfectly in the preview. Its source code highlighting is some of the best we've seen, including the difficult-to-parse territory of JSON. The latter even includes folding, so you can hide a specific hierarchy of elements in the preview.



Regardless of which elements you embed within your Markdown, Aurora Editor seems to create a perfectly rendered preview.

But you can even include Graphviz and Mermaid source directly in your Markdown, and they'll be rendered perfectly in the preview. Most importantly, if you choose to export your Markdown as a single HTML file, all this beautiful output is maintained, which means you can easily use Aurora to produce user-friendly output and documentation for websites. As this is such a functional early release, we can't wait to see what developments follow.

**Project Website**  
<https://aguang-xyz.github.io/aurora-editor>

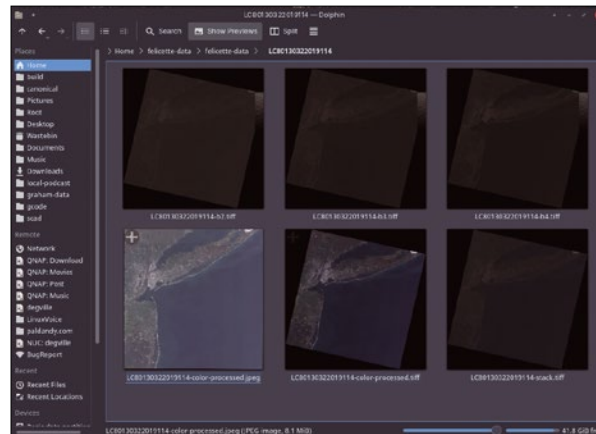
Satellite imagery

# felicette

We've all become desensitized to the miracle of pervasive access to satellite imagery from various online mapping services. It's a brilliant way to be a virtual tourist, to check out a location before you travel, or even to see how much your hometown has changed since you left for college. But when it comes to experimentation, their cloud data is distant and unobtainable from our Linux boxes. There's obviously OpenStreetMap, which can help, and KDE's Marble has various layers with satellite imagery, but nothing if you want to access some real detail or image data that's more timely. Which is when felicette can help. This is a command-line tool for downloading very high resolution satellite imagery from the NASA/USGS

prestigious Landsat satellite imaging array, which is constantly taking 115-mile-wide photos of the Earth at almost every location, day and night.

The command couldn't be simpler to use. Type `felicette -l "London"`, for example, to grab the latest images of the UK's capital. Being simple to use doesn't mean it's simply downloading the images though. In the background, felicette is downloading blue, green, and red band wavelength imagery and stacking these with different tools to create the final composite JPEG and TIFF, the latter of which is typically 350MB in size. If you want London, Ontario, then you can use latitude and longitude coordinates instead, such as `felicette -c 43 -81.25`. There are



Download recent high resolution satellite imagery directly from the command line.

two further options for enhancing the image: the first by adding the panchromatic band, and the second by adding an infrared layer to highlight vegetation. Thanks to the high resolution of the images, and the fact that they're normally recent, felicette is the perfect way to automate image collection for a specific area you're interested in, so you can then compare changes over time. But it's just as useful if you want to see how your hometown has changed.

**Project Website**  
<https://github.com/plant99/felicette>

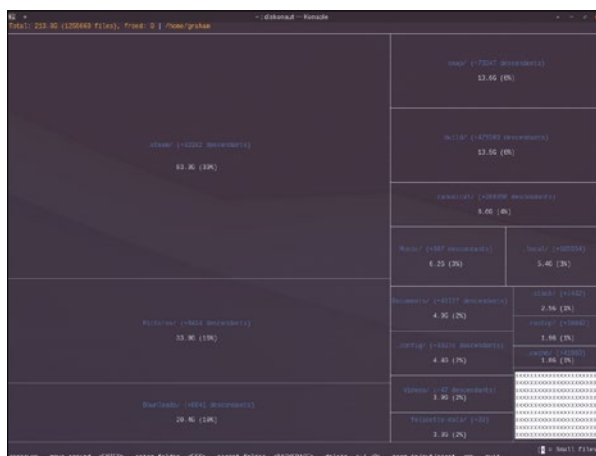
File explorer

# diskonaut

It's incredible how cheap local storage has become. Terabytes of data are within easy reach. And yet, we're all still making the same old compromises when it comes to data storage on the fastest and best media we own. While many of us have an adequate stash of SSDs for day-to-day use, it's equally likely our best M.2 PCIe NVMe SSDs are reserved for the data we access every millisecond. That means we still need to monitor how much we store and where we store it. Just like 120m compact cassettes in the '80s, high-density floppies in the '90s, recordable optical media in the '00s, and flash storage in the recent era, what's stored where is still very important. Which is why tools like diskonaut will always

be useful, because they help you discover exactly what's taking up your valuable storage. It also helps that it's a fantastic piece of eye candy.

Even from the command line, diskonaut exudes plenty of style. It's invoked by simply adding the path you wish to explore or by running the command from any directory. You're immediately presented with an updating view of a local folder scan, with new folders and files automatically populating a grid, called a visual treemap, as they're scanned, indexed, and collated. You can start exploring even before this grid has been fully populated, using either Vim navigation or arrow keys and Enter to access a folder. The size of each cell is proportional to the amount of



Freeing up storage space has never been so satisfying, thanks to diskonaut.

storage it and its descendants (subdirectories) requires. When you find something surplus to your requirements, you can delete it with a simple `Ctrl+D`. Diskonaut will even keep track of how much space you've been able to free, which is handy if you need to hit a specific target to install a new game.

**Project Website**  
<https://github.com/imsnif/diskonaut>



## Software synth

# Surge 1.7

**W**e first looked at Surge a little while ago, but the project has seen so much development and increasing community momentum over the last 12 months that it's worth looking at again. Surge is a software synthesizer that can genuinely compete with both hardware equivalents and with commercial software synthesizers on other platforms. This makes it almost unique on Linux, because the only other soft synth that gets close is the equally brilliant but less versatile Helm. One of the things that makes Surge unique is its ability to interpret MIDI polyphonic expression (MPE) data. While this is a brand new part of the MIDI 2 specification, there are very, very few synthesizers that can take advan-

tage of it. MPE means you can send touch data for every single note, rather than a single value shared for every note being pressed. Surge can process this data to change a value specific to a single note.

MPE is commonly used to adjust per-note filter values, but it can be equally applied to pitch, LFO speed, or any other modulation source supported by Surge. This ability has even attracted the attention of electronic music pioneer, Roger Linn, inventor of the 1980s-defining LinnDrum; he's now the contributor of a complete set of MPE-inspired Surge presets. Surge is equally flexible in the types of sounds it can make, because it doesn't simply focus on a single type of synthesis, which is what Helm does. It's



**While the skin could do with an overhaul, the new darker theme helps Surge better integrate with all kinds of hosts.**

equally good at both subtractive and FM sounds, for example. The new release improves the sine oscillator for incredibly powerful FM, including wavetables from the classic TX hardware synths and extended feedback modes. There are new reverb, ring modulator, and rotary speaker effects, which can be used separately from a standalone effects plugin, and best of all, an ARM-build, which literally lets you turn a humble Raspberry Pi into a full-fledged modular synthesizer. Surge 1.7 offers all of this while using the new and more open VST3 standard.

**Project Website**

<https://surge-synthesizer.github.io>

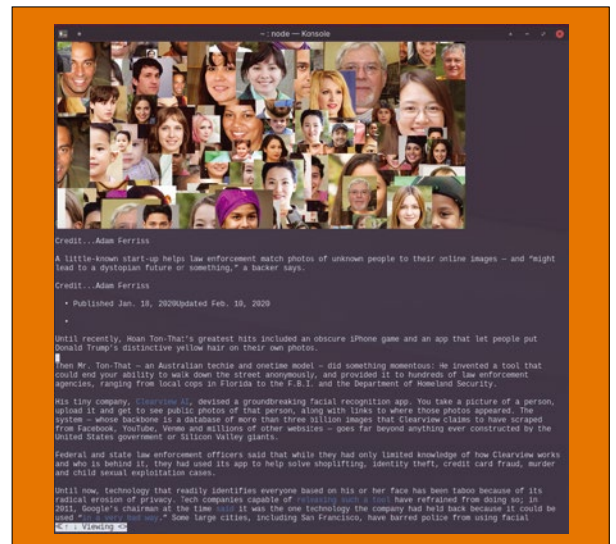
## Console-friendly HTML

# readability-cli

**I** was asked in a recent podcast which tools I'd like to see on the command line. At the time, without too much thought, I answered that I'm still looking for the perfect email client – something like Mutt but with native Vim bindings and a modern, semantic searchable back end. But what I really should have said was that I wanted a modern web browser for the terminal. The reason why I didn't think about this is because web browsers have become so far removed from what the command line offers that creating a fully functional command-line browser seems untenable. Of course, there are options like Lynx and w3m, but they're more useful when you can't access your desktop and need to download some web-based drivers, rather than being full-fledged

replacements. What we really need is a classic desktop browser built for the command line, and that possibility still seems far off.

Readability-cli *isn't* a classic desktop browser built for the command line, but it is an important step towards making it a reality. As keen readers will know, **Readability** is a Firefox function that removes all extraneous content from a web page to leave only the pure content. This content is then presented through the **Readability** interface much like an ebook, and it's a brilliant function for distraction-free reading. readability-cli performs the same function, using Mozilla's **Readability** library to generate distraction-free HTML from either a URL or the standard input, which you can then either save or pipe



**When combined with a console that can embed images, web browsing from the command line is almost better than with a GUI.**

into your favorite terminal web browser. It works brilliantly, because it's not directly interactive, it helps you avoid the temptation of clicking on other links – even when those links are maintained and passed through to the output – keeping your focus on the page you really wish to read.

**Project Website**

<https://gitlab.com/gardenappl/readability-cli>

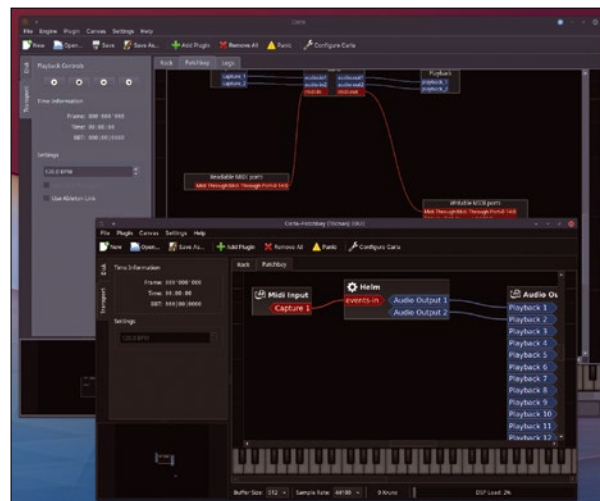
## Audio effect and instrument host

# Carla

The KXStudio project is the curator of some of the best audio and music production utilities on Linux. These utilities are often built to make sense of the Linux audio ecosystem and to provide essential functionality that would otherwise be missing on Linux. The Cadence front-end to JACK, for instance, is the best way of interacting with both JACK and PulseAudio on your Linux desktop. Similarly, Carla is one of the only ways you can host synthesizers and effects on your Linux desktop without having to resort to a full-fledged piece of audio software. This is important because not all audio software can host every kind of plugin – VST3 support is rare, for instance, and you don't always want to instantiate a piece of recording software to play a sound. Carla is the equivalent of a virtual studio, allowing you to load and route audio and MIDI data from any source to any destination. Vivality, all of this works without JACK, which is the normal layer for this kind of functionality on Linux. Carla will work natively with PulseAudio without requiring you to

make any sacrifices to the gods of real-time kernels.

The main application uses two main views, and they both lean heavily on the studio metaphor. The first is the rack. This is where you add your plugins, and the audio, or MIDI data, passes through them from the top down to the bottom. Which plugins are available depends on what you have installed. Carla supports more formats than any other application we can think of: LADSPA (including LRDF), DSSI, LV2, VST2 and VST3, plus SF2 and SFZ files for sample playback. There's even an experimental option to enable Wine bridges so that you can include Windows-based plugins natively on your Linux desktop. This breadth of support means you potentially have hundreds of open source and proprietary plugins at your disposal, all of which can be dropped into the rack to create anything from screaming guitar effects via cavernous echo chambers to prog rock synth stacks reaching to the sky. You can open up either the custom GUI for each plugin or the parameter list rendered within Carla. The latter allows you to save your own presets, which extends to the complete collection of plugins you



The patchbay allows you to connect anything to anything else within its own isolated container and save it as a preset.

add to the rack so that you can quickly retrieve exactly the same setup if you need to. You can also use this list to assign external MIDI controllers to each parameter, such as a knob or joystick, and even send software controller output to a control voltage device to control external analog synthesizers.

Before you hear anything, you first need to switch to the patchbay. This is the equivalent to the cabling behind the rack and allows you to route your system audio and MIDI into and out of Carla. To hear the output from your rack, for example, connect the audio outputs from the Carla module to the playback module. However, the super-clever aspect to this is that you can add patchbays to the rack and use these and separate, modular, setups to route audio between modules. It's like the *Inception* of audio solutions. While most users won't need to delve into this complexity, it's an unparalleled feature on any operating system and one of Carla's main strengths. If you've ever wanted to tinker with effects and/or synths without getting into the complexity of a DAW, Carla has to be your go-to choice.



Carla can be used to easily create an audio effects chain, such as for guitar effects, or for hosting synth plugins.

**Project Website**  
<https://kx.studio/Applications:Carla>

## RPG shooter

# BYTEPATH

It's perhaps understandable why so many game publishers choose not to release a game's source code after its commercial life has ended. You never know when the retrowave will wash over it and create a new source of revenue. But there's only ever a small chance this will happen, and releasing the source code today can help a whole new generation of game developers to create the games industry of tomorrow. BYTEPATH's developer has taken this pragmatic approach and not only released its source code, but has also published a brilliant tutorial on how the game was created. This takes you from the main game loop and libraries, through gameplay, enemies, and coding practice; it is one of the

most comprehensive tutorials we've seen.

The game itself is a unique kind of arcade shooter with a 2D side-view retro-vector style that feels at times like Asteroids, Tempest, and the much later Geometry Wars. What makes BYTEPATH unique is a crazily large and complex skill tree that allows you to control how your skills are upgraded. You start off by selecting your character/ship. Each ship has a different class tree and abilities, and these equate to different low-level presets in the skill tree, which can then be augmented with skill points (SP). You're given an amount of SP at the start, but you collect more when you're inside the shooter element of the game. This sees you flying



BYTEPATH has had Linux support from day one; its developer has blogged about its sales and comparatively low Linux sales numbers.

through space, collecting debris, SP, and upgrades while shooting things, and hopefully lasting long enough to collect the keys required to finish. The entire game is rendered through what feels like a broken terminal filter. It also features some beautiful 8-bit style music to go along with the visuals. Written in Lua with the LOVE game engine, the code has been released under MIT, with varying licenses for media. It's both a brilliant game and an important open source project.

**Project Website**

<https://github.com/a327ex/BYTEPATH>

## Driving simulation

# Trigger Rally

Thanks to Steam, Linux is now well-provided for when it comes to driving games. We can run best-in-class DiRT 4, it's older and more accessible cousin, Dirt Rally, and various incarnations of games in the F1 franchise. But there's also a surprising number of open source driving games. There are the driving simulations TORCS (which begat Torcs-NG and Speed Dreams) and VDrift, and of course there's the crazily addictive console-style SuperTuxKart. And if you want an arcade rallying experience, there's Trigger Rally. In Trigger Rally, you view your clunky kit car from above and behind its rear, using a joystick of the cursor keys to navigate the 3D terrain in front of you. What makes it unique is the brilliant physics engine that

not only handles collisions with aplomb, but provides a very satisfying driving experience. In many ways, it feels like TuxRacer with a car rather than a penguin, and that's no bad thing.

The car has weight and what feels like realistic inertia, which is different when driving either the front-wheel drive and rear-wheel drive vehicles in the game. You're forced to adopt the rallying driving style of turning into corners early and oversteering through their apex. This is counterintuitive at first, but hugely satisfying when you get it right. The rallying aspect in Trigger Rally comes from never really knowing where a course will take you. You steer after being prompted by arrows that appear as your virtual navigator and copilot. The variation is thanks to the



There's even a WebGL version of Trigger Rally that you can play online – after getting plenty of offline practice!

huge number of courses the game includes, from mountains to the desert, from tarmac to sand. It means you never really get a chance to learn a course in the same way you might learn Spa-Francorchamps. Even better, it's easy to design your own courses with little more than a graphics editor and a text editor. Each level is little more than an image map for height, color, and foliage, and an XML file for checkpoints and locations, but the generated output is fantastic.

**Project Website**

<https://trigger-rally.sourceforge.io/>



# Write Inkscape extensions that modify objects

# Trembling Text

Writing your own extension for Inkscape opens up a whole world of possibilities. Apart from creating new objects, you can modify existing objects and even animate them.

BY PAUL BROWN

In last month's issue [1], we saw how to write an extension that rendered an object (a circle) in an Inkscape document. But apart from creating new objects, Inkscape extensions can also be used to modify existing objects. Let's see how this can be done by creating an extension that will generate "wobbly" animation. The idea is to take a given path – say, a piece of text – and to move its nodes around a little bit. Then save the result as a frame, move the nodes a little more, save again, and so on. When you put the frames together, it will give the impression that the text wobbles. This video shows how to do it in After Effects [2], but it seems like a lot of work for something that could be done with a relatively simple script.

Unfortunately, documentation and tutorials explaining how to create this type of script are all but nonexistent. Again the only way forward is to wade through source code and comments within code [3] to try and figure out what tools are available to achieve our ends.

## What Is a Node?

Any object in Inkscape can be broken down into a bunch of paths, and paths, in turn, are made up by a bunch of nodes. So the key to modifying any object is modifying its nodes. But what is a node? You may think it is the point on a path where the path can change direction. In fact, that is just one control point, and a node is made up of three control points.

If you have used Inkscape to any extent, you will be familiar with this: Draw a Bézier line with several segments from top to bottom. Using the *Edit Paths by Nodes* tool, select all the nodes (Ctrl+A) and make the segments curve using the button in the toolbar. For added clarity, make the selected nodes symmetric, and you will end up with something like what you can see in Figure 1 (left).

The three control points, in order, are as follows: control point 0 is the control handle at the top, control point 1 is on the curve itself, and control point 2 is the handle at the bottom. If you drew the

### Listing 1: cps.inx

```
01 <?xml version="1.0" encoding="UTF-8"?>
02 <inkscape-extension xmlns="http://www.inkscape.org/namespace/inkscape/extension">
03
04 <name>CPS</name>
05 <id>org.linuxmagazine.inkscape.effects.cps</id>
06
07 <param name="CPS0" type="float" gui-text="CPS0:" min="-10" max="10">1</param>
08 <param name="CPS1" type="float" gui-text="CPS1:" min="-10" max="10">1</param>
09 <param name="CPS2" type="float" gui-text="CPS2:" min="-10" max="10">1</param>
10
11 <effect>
12 <object-type>path</object-type>
13 <effects-menu>
14 <submenu name="Modify Path"/>
15 </effects-menu>
16 </effect>
17
18 <script>
19 <command location="inx" interpreter="python">cps.py</command>
20 </script>
21 </inkscape-extension>
```

curve from bottom to top (Figure 1, right), the order would be inverted. Likewise if you drew from left to right, the first control point would be on the left, and drawing from right to left puts the first control point by default on the right.

Of course, you can grab the control points and move them anywhere you want, inverting their position, for example, but then you get loops and whorls in your path.

How do we know all this? Not through any documentation on Inkscape, but you can figure it out by experimenting with the extension shown in Listings 1 and 2.

Listing 1, `cps.inx`, is a very basic interface to the extension. Read the sister article to this one to get the details [1]. In essence, it reads in three parameters, `CPS0`, `CPS1`, and `CPS2` (lines 7, 8, and 9), one for each control point of each node. The parameters then get passed off to a Python script (line 19) that does the actual work.

The `cps.py` script (Listing 2) receives the parameters on lines 9, 10, and 11, and adds them to the coordinates of each of the control points of each node in the selected path (lines 24 to 29). Note that each node has an *x* and a *y* coordinate, so for example, the coordinates for control point 0 are `csp [0][0]` (*x* coordinate) and `csp [0][1]` (*y* coordinate).

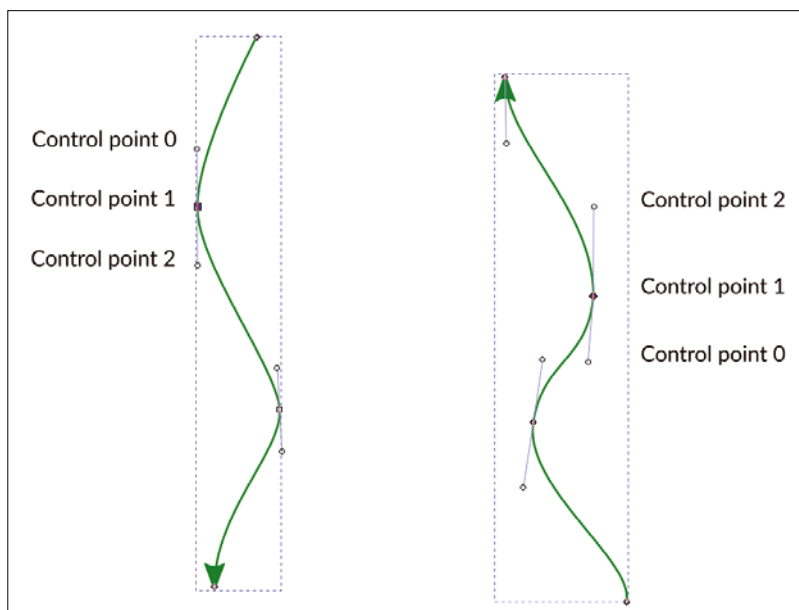
Save `cps.inx` and `cps.py` to your `$HOME/.config/inkscape/extensions` directory and the `CPS...` extension will appear under the *Extensions | Modify Path* menu next time you start Inkscape.

If, for example, you input 5 into the `CPS0` field in the extension's dialog, the first handles of all the nodes in the selected path will move down and to the right five pixels, millimeters, or whatever unit you are using, from their original position as shown in

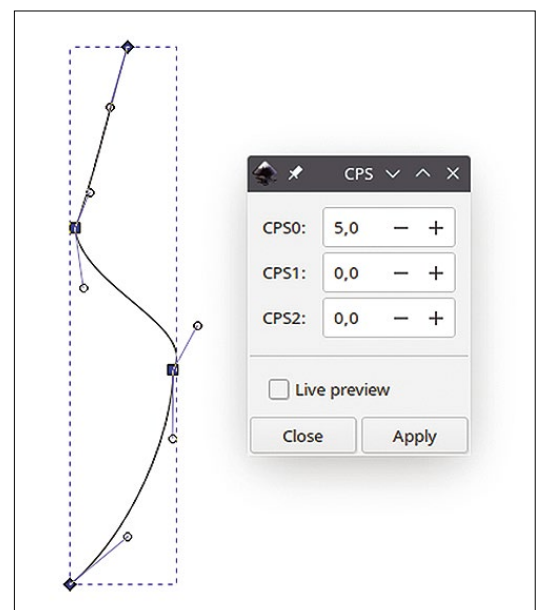
### Listing 2: `cps.py`

```
01 #!/usr/bin/env python
02 # coding=utf-8
03
04 import inkex
05
06 class CPS (inkex.EffectExtension):
07
08     def add_arguments (self, pars):
09         pars.add_argument ("--CPS0", type=float, default=1, help="CPS0")
10         pars.add_argument ("--CPS1", type=float, default=1, help="CPS1")
11         pars.add_argument ("--CPS2", type=float, default=1, help="CPS2")
12
13     def effect(self):
14         for node in self.svg.get_selected(inkex.PathElement):
15             path = node.path.to_superpath()
16
17             for subpath in path:
18                 closed = subpath[0] == subpath[-1]
19                 for index, csp in enumerate(subpath):
20                     if closed and index == len(subpath) - 1:
21                         subpath[index] = subpath[0]
22                     break
23                 else:
24                     csp[0][0] += self.options.CPS0
25                     csp[0][1] += self.options.CPS0
26                     csp[1][0] += self.options.CPS1
27                     csp[1][1] += self.options.CPS1
28                     csp[2][0] += self.options.CPS2
29                     csp[2][1] += self.options.CPS2
30
31             node.path = path
32
33 if __name__ == '__main__':
34     CPS().run()
```

**Figure 1:** A node is made up of three control points: the point where the path changes direction and the two handles that define the curvature at said point.



**Figure 2:** The CPS extension will help you figure out which control point is which among the nodes on a Bézier curve.



**Listing 3: tremble.inx**

```

01 <?xml version="1.0" encoding="UTF-8"?>
02 <inkscape-extension xmlns="http://www.inkscape.org/namespace/
inkscape/extension">
03
04 <name>Tremble</name>
05 <id>org.linuxmagazine.inkscape.effects.tremble</id>
06
07 <param name="radius" type="float" gui-text="Radius:">1</param>
08 <param name="accumulate" type="bool" gui-text="Accumulate
changes">false</param>
09 <param name="frames" type="int" gui-text="Frames:" min="1"
max="250"/>
10 <param name="folder" type="path" gui-text="Save frames in:"
mode="folder"/>
11
12 <effect>
13 <object-type>path</object-type>
14 <effects-menu>
15 <submenu name="Modify Path"/>
16 </effects-menu>
17 </effect>
18
19 <script>
20 <command location="inx" interpreter="python">tremble.py</command>
21 </script>
22 </inkscape-extension>

```

**Listing 4: tremble.py**

```

01 #!/usr/bin/env python
02 # coding=utf-8
03
04 import random
05 import inkex
06 from inkex import command
07
08 class Tremble (inkex.EffectExtension):
09
10 def add_arguments (self, pars):
11 pars.add_argument ("--radius", type=float, default=1,
help="Radius")
12 pars.add_argument ("--frames", type=int, default=1, help="Frames")
13 pars.add_argument ("--folder", type=str, help="Path")
14 pars.add_argument ("--accumulate", type=bool, help="Accumulate
deformation")
15
16 def effect(self):
17 for node in self.svg.get_selected(inkex.PathElement):
18 path = node.path.to_superpath()
19
20 for frame in range (1, self.options.frames):
21 for subpath in path:
22 closed = subpath[0] == subpath[-1]
23 for index, csp in enumerate(subpath):
24 if closed and index == len(subpath) - 1:

```

Figure 2. Note that position (0, 0) is located by default at the upper left-hand corner of the page in Inkscape, so x coordinate plus five is five units to the right and y coordinate plus five is five units down.

**Moving Nodes**

The implication of all of the above is that, to move a node, you have to move each of its three control points (handles and position on curve) all at the same time ... or not, if you want an even messier wobble.

This is what the *Jitter Nodes...* extension does. This extension is shipped by default with Inkscape and you can find its code in the `/usr/share/inkscape/extensions` directory. Indeed, `jitter.py` is the inspiration (read "I blatantly ripped it off") for Tremble, my own extension shown in Listings 3 and 4. Tremble is the extension that makes trembling, wibbly-wobbly animations out of selected objects.

The INX file, `tremble.inx` (Listing 3), is pretty standard. It allows you to decide the amount of wobbliness to apply to each node (line 7), how many frames to generate (line 9), and where to save the frames (line 10).

The latter is the only new parameter type in this interface; all the rest we saw in the previous article. The `path` parameters lets the user input or select a path to a file (`mode = "file"`) or a folder (`mode = "folder"`), either to load it into Inkscape or save something later to disk. You can give the user the option of opening several files (`mode = "files"`) or folders (`mode = "folders"`) at the same time or create a new file (`mode = "file_new"`) or folder (`mode = "folder_new"`).

The script that does the actual work, `tremble.py` (Listing 4), is also not terribly complicated. Up in the heading, you import Inkscape's `inkex` module, which contains many of the methods and attributes you need to write extensions. You will also need Python's `random` module. Finally you will be using something from Inkscape's `command` module.

The `command` module includes, among other things, some interesting methods, like `take_snapshot ()`, which saves a bitmap snapshot of the current SVG loaded into Inkscape; and `write_svg ()`, which saves the SVG generated by Inkscape to a file. You will use the latter a bit later in your script as you can see on line 37.

Inkscape's `add_arguments ()` method (lines 10 to 14) is what Inkscape's interpreter expects to find when it needs to read in the parameters coming from the INX file. The `--radius` parameter (line 11) is read into `self.options.radius`, the `--frames` parameter (line 12) is read into `self.options.frames`, and so on. Notice how there is not a special type for the path held in the `--folder` parameter: It is just a regular `str` type.

The real action starts in the `effect ()` method (lines 16 to 37), another of the standard methods



that Inkscape's interpreter expects and what it runs when it is done reading in parameters.

The first thing you need to do is to grab the information from the selected object. Your script can find the nodes of the currently selected object using the `get_selected()` method from INX's `svg` module (line 17). The parameter `inkex.PathElement` tells `get_selected()` what sort of object it should expect, in this case, a path.

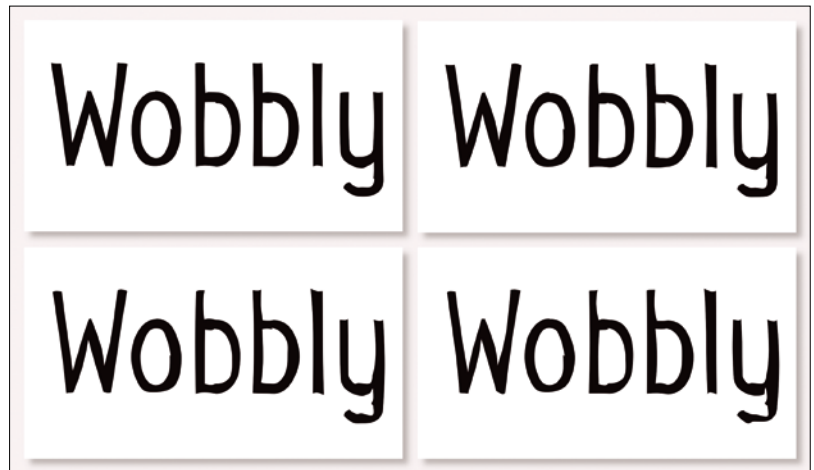
This allows you to loop over each path the nodes belong to and turn them into *superpaths* (line 18)!!! Okay ... to be fair, there is nothing that much to get excited about here, despite the name. Superpaths are just an internal construct that makes it simpler for Inkscape's interpreter to calculate the changes that you will inflict on the path by modifying the node.

Let's hold line 20 for a moment and move on to line 21.

As each object can be made up of several paths (an object created from the letter "i" for example, has two paths: the body of the letter and the dot), you have to iterate over each subpath (line 21) and iterate over every control point of each individual node (line 23).

First, though, you have to check to see if the object is closed (line 22). The last node in a closed body coincides in its location with the first one, but internally the first and last nodes are two different items in the SVG markup. You check by comparing the first subpath (`subpath [0]`) with the last subpath (`subpath [-1]`). If they are the same, you must treat the object as closed (lines 24 to 26). This means that when you reach the last node, you need to skip messing with it and quit the loop (line 26), because you are done.

Just to get back to the `to_superpath()` method a second, it splits all the nodes into two parts: the `index`, which you can see being used on lines 23,



**Figure 3:** Frames from a wobbly animation generated by Tremble.

24, and 25, is the number of the node (the first node is 0, the second is 1, etc.). This allows you to know when you have reached the last node in the subpath, as you can see in line 24.

The second part is the data regarding the control points, which we talked about above. Once you calculate the `delta` that you are going to apply to each node (i.e., the amount by which you are going to move the node), you add it to each of the `x` and `y` coordinates of the handles (lines 29, 30 and 33, 34) and the point on the curve itself (lines 31 and 32).

Once you have dealt with all the nodes in all the subpaths, you can dump the superpath `path`, with the modified values, back into `nodes.path`. The changes get written to the internal SVG structure, and you will see the changes appear in your Inkscape drawing.

The script then saves the `svg` object to the folder you chose on line 37.

Of course, you have to do this as many times as the number of frames you want. That is why you envelop the path-wobbling process in a loop on line 20.

#### Listing 4: tremble.py (continued)

```
25     subpath[index] = subpath[0]
26     break
27 else:
28     delta = random.uniform (-self.options.radius, self.options.radius)
29     csp[0][0] += delta
30     csp[0][1] += delta
31     csp[1][0] += delta
32     csp[1][1] += delta
33     csp[2][0] += delta
34     csp[2][1] += delta
35
36     node.path = path
37     command.write_svg(self.svg, self.options.folder + "/frame" + f'{frame:03}' + ".svg")
38
39 if __name__ == '__main__':
40     Tremble().run()
```

Copy both `tremble.inx` and `tremble.py` to your `$HOME/.config/share/inkscape/extensions` folder, and it will be ready to go the next time you start Inkscape.

### Workflow

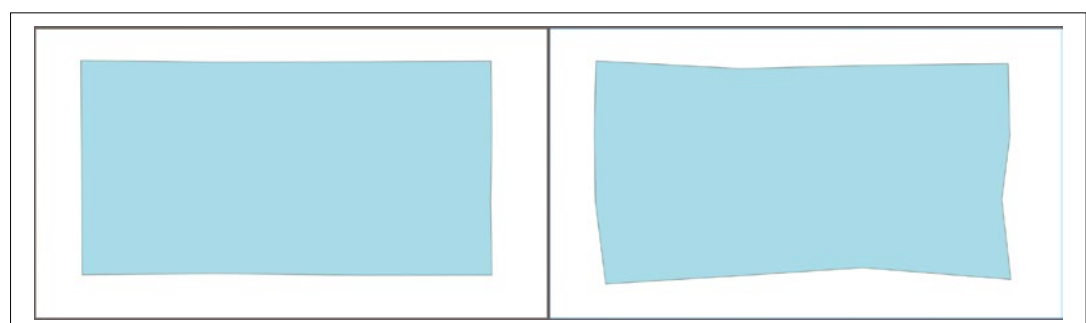
You make the most of this extension like this:

1. Draw the thing you want to animate and select it. Convert it to a path by picking *Path | Object to Path* from the menus or by pressing `Ctrl+Shift+C`. If the object is made up of different bits, like different letters in a piece of text, make sure to combine them all together into one path by selecting all nodes (`Ctrl+A`) and choosing *Path | Combine* from the menus or by pressing `Ctrl+K`.
2. If the object does not have many nodes, like a rectangle or a circle, you may want to add more to get more wobbliness. You can do this by hand by adding nodes at strategic points on the curves, or you can add nodes automatically using *Extensions | Modify Path | Add Nodes...*
3. Now is the moment to use Tremble. Make sure your object is selected, navigate to *Extensions | Modify Path | Tremble...*, fill in the form, and press *Apply*. A bunch of SVGs will pop up in the folder you chose (Figure 3). Note that, after running the extension, the object in Inkscape will be a version of your object that has been modified by the script. If you want to keep the original, *do NOT save it!* You will overwrite your original image. You can always press `Ctrl+Z` to undo the changes.
4. Most video editors will not load SVGs as a sequence, so you may want to convert your images into something they will accept, like PNGs. You can do that en masse by changing to the directory where your frames are located and running:

```
for i in *.svg; do convert $i ${i%.svg}.png; ?
rm $i; done
```

Note this will delete the original SVGs. Get rid of `rm $i` if that is not what you want.

**Figure 4:** The rectangles in the first frame (left) and the last frame (right) of this animation should be deformed to the same degree. Since deformation is accumulative, the rectangle in the last frame is noticeably more deformed.



5. Use a video editor like Kdenlive or FFmpeg to convert the sequence of images into a movie.
6. Impress your friends and family ... or not.

### Accumulations

There is one problem with the extension. I realized quite early on that the effect was accumulative, that is – when the script applies a new deformation to a path, it applies it to the previously deformed path. This means that, over time, your object will become more and more deformed (Figure 4).

This is not what I wanted, but I also thought “Interesting,” so I decided to make it an option, hence the `accumulate` parameter on line 8 in `tremble.inx` (Listing 3). However, I have not figured out how to avoid the accumulation, so, for the moment the extension acts as if it is always on.

That said, as long as the radius is relatively small and the clip short, it shouldn’t be too noticeable. When I hit on the solution, I will update the code. If you find this extension useful, you will be able to download the updated version from my GitLab account [4].

Either way, that is not the point of the article. The point was always to show off the power of Inkscape’s extension toolkit, and very powerful it is.

All it needs now is comprehensive documentation to match. Have fun! ■■■

### Info

- [1] “Magic Circle: Write your own extensions for Inkscape” by Paul Brown, *Linux Magazine*, issue 239, October 2020, pg. 88, <https://www.linux-magazine.com/Issues/2020/239/Magic-Circle>
- [2] How to make wobbly text using After Effects: <https://youtu.be/XXDxkMOCKgQ>
- [3] Inkscape’s available Doxygen documentation for extensions, which includes source code: <http://inkscape.gitlab.io/inkscape/doxygen-extensions/>
- [4] Paul Brown’s *Linux Magazine* GitLab repository: <https://gitlab.com/linux-magazine>



Most  
Extraordinary!

THE ULTIMATE

# raspberrypi GEEK

ARCHIVE



GET EVERY ISSUE OF  
**RASPBERRY PI GEEK**  
ON ONE  
FULLY-SEARCHABLE  
**DVD!**



**Order Now!**

<https://bit.ly/Archive-DVD>



# Celebrating 20 years of Linux Magazine Cheers!



Editor-in-chief Joe Casad reflects on the enchanting 20-year story of Linux Magazine.

BY JOE CASAD

I roll out of bed and start the coffee. The dog follows me around, expecting breakfast. I feed him; take a shower. It is still early and the first rays of sunlight are tangled in the trees over my neighbor's house. I pour a cup of coffee and sit at the desk in the small bedroom I use as my office...start my Linux system, call up Slack, check my notes: 9 o'clock Zoom call?

Like many companies around the world, our office has gone all-virtual due to the COVID-19 pandemic. We meet together online once per week for a roundup of company news, but information

passes between us all the time, all day, in email, texts, Zoom calls, and posts to workgroups. A publishing office is a frenetic place even on the slow days, and when it gets busy, it is impossible to imagine how it all can stay floating – text files, layout files, emails, author queries, and social media posts fly in every direction, and all the threads converge magically at a sacred moment when we upload the issue to the printer. I'm always amazed when nothing breaks, and the fact that we have smoothly navigated to a remote workday is a testament to our experience, versatility, and esprit de corps. But you see, it hasn't always been so virtual. Most of our years we have worked together face-to-face, and our vibrant office culture has always been a source of pride.

*Linux Magazine* launched way back in 2000. Our parent company at the time, Linux New Media AG, had been publishing a high-quality Linux magazine in Germany since the very early days of Linux. The founders had a vision for a network of magazines in different languages around the world – an international community like Linux itself – sharing information, expertise, and resources. At one point we had Linux Magazines in German, Polish, Romanian, Spanish, Brazilian Portuguese, and English – all published in tiny offices like ours with small budgets, but all cooperating and making the most of shared resources.

The original English edition was a joint venture with a company in the UK, and the founding editor was Julian Moss. At around the sixth issue, the torch passed to John Southern, who served as editor until I started with Issue 48. Somewhere around Issue 26, the publishing and layout gravitated back to the continent. The English team shared production and marketing resources with the parent company in Munich for a few years, but everyone knew that, when the time was right, they would look for a home in the English-speaking world.

When I first started, I was the only US employee and worked out of my basement in Lawrence, Kansas, editing and coordinating the production with the team back in Munich. The founders had originally wanted to put the U.S. headquarters on



Snapshot from the past: the team at ease (from left: Dena, Ann, Lori, Rita, Darrah, Emily, Joe).



Cindy, Rita, Dena, and Gwen toast to another successful deadline, along with Cindy's husband Dave.

one of the coasts – wondering, as do many, what was in between – but the CEO visited Lawrence and quite liked it, conceding that the beer at our local brew pub was more than acceptable to the German palate. (That’s a good sign, when people from Munich like your beer.)

After a couple years of toil, with circulation and ad sales trending up, we had gained enough momentum to open our first office in the US. At first it was just me and distinguished alumna Rikki Endsley, who started as managing editor and then became associate publisher when our publisher, Brian Osborn, a young American ex-pat with an infectious poker face and a rare knowledge of publishing on two continents, took on additional responsibilities at the corporate level. Our office was a one-room space above Buffalo Bob’s Smokehouse. We had a big bank of windows that looked out on Massachusetts Street. (The main downtown street in Lawrence is called Massachusetts Street because the town was settled by radical abolitionists from Massachusetts in the 1850s.)

We worked hard, talked about life, and watched a lot of time go by, not just in our lives, but in the evolution of Linux. Our friends in the local publishing community would stop by to see our hip office space. I think they were all a little jealous about our cool gig putting out our own Linux magazine in our funky downtown office. One of the reasons for starting the company in Lawrence was the indigenous reserve of IT-publishing veterans who got their start at local companies like R&D Publications and CMP. Over the years, we’ve enticed several former co-workers to come and join us.

### Dirty Baby

Our official mascot is a baby doll we found in the fork of a tree one day outside of our office. Dirty baby shows up for all our parties and rarely behaves well. Visit Dirty Baby on Facebook at @itsdirtybaby.



Oh, to be young again...



Amber, Lori, and Rikki share a moment for cappuccino and memories.



Brian and Gwen celebrate happy hour on the trade show floor.



Not sure what we were thinking on this one.



A couple times a year, Brian would come to town, often on his way to a tech conference or other event, and we would spend a week planning (more like scheming) about where we were going and what dragons we would be hunting next. With each issue, the whole thing felt more like a real company and less like an experiment. Gradually we brought in new services and capabilities, scaling up our operation. We launched our special edition series, added an off-site ad sales team, and increased our involvement with conferences and the open source community. At one particularly surreal moment, we had to occupy a temporary location at the back of the building so they could install a balcony on our office (how often does that happen?). We put out the first edition of *The Shell Handbook* crammed into a tiny windowless cubicle previously used by a psychotherapist while they crashed down the facade and installed the balcony. The balcony was super cool and made our friends even more jealous, but when our pop-

ulation rose to four in our little one-room space, we knew it was time to find another home.

Our next office was a house converted into an office building that was on a street a couple blocks off the downtown. The building had a big front porch and looked out on a park. The park had a large decommissioned steam locomotive that I could see from my desk at a second-story window. When I was a kid, they let you climb all over that train, poking around and exploring the mechanical crannies. Now there was a big fence around the train, so you couldn't just go poking around on it, but that was OK with me, I often thought, because now I get to poke around on Linux.

We expanded our scope considerably during the train park years, with newsletters, websites, custom content, and lots more stories to tell. Like the time a distributor went bankrupt and left three months of magazines stranded in a warehouse, or the time we were threatened by a patent troll who claimed he had a patent on attaching a DVD to a magazine. And then there was the episode when the DVD company accidentally put a porno film on some of the UK copies of the DVD. We have weathered every crisis together with a rare combination of aptitude and attitude that would be the envy of any small company.

Eventually, the global organization split up, and the English-language operation spun off to become a separate company. It's just us now, although we still collaborate with some of our former international counterparts. We've moved from the train park office to another location in a fancier part of town that isn't quite as bohemian, but it sure is easier to find a parking place.



A night out together (Dena, Joe, Rita, Darrah, Lori, Carson, Ann, Brian).



Amy and Megan are our stalwart final defense against the solecistic and labyrinthine.



Jessica tunes in on a customer in need.



I'm struck by how much Linux has changed since I started this job – and how the publishing industry has itself remained in a perpetual state of reinvention. It is one thing when the subject of the magazine is continually transforming – and quite another when the very context in which you operate is a moving target. There's no doubt in my mind that the secret to our success is our hard-working team of professionals, who stay calm under pressure and show up every day with ideas and good energy: They never say "no"; they just start thinking about how.

When I look back over my years at Linux New Media, I remember lots of great magazines and lots of grace under fire. I remember fabulous Christmas parties and afternoon escapes to Dempsey's tavern. I recall the kids who have been around the office through the years and adorned our walls with their art – including my own kids, who are now well into their 20s, as well as our pets, and even our parents. Birthday snacks, Valentine cards, meeting-day pizzas, and warm nights watching the 4th of July fireworks from the alley...

Through the years, I've had the privilege of watching the rise of the Linux community with a special little community of battle-worn pirates at Linux New Media. Happy 20 years everybody. Here's to 20 more. ■■■



We spent many formative years in our train park office. The ornamental lamp post in front always made me feel like I was crossing the border into Narnia.



No chore is too tough or too daunting for Dirty Baby and the Linux New Media team.



Our vintage collection of booth-tchotchke penguins peek out at the world from our cozy front window.



# IT Highlights at a Glance

**ADMIN HPC**  
HPC news, views, and technical articles delivered to your inbox every month. [Subscribe Now!](#)

**ADMIN Update - Hottest Links**

- **WAP and NAT**
- **Ubuntu 20.04 LTS** includes apt Snapshots
- **Simple & Secure**
- **Linux**
- **Articles in the Hybrid Cloud**

**Highlights**

**WAP and NAT**

We show you how to secure transparent IP address translations through NAT tunnels and gateways for Voice over IP (VoIP). (links) 20.04 LTS includes apt Snapshots

The next release of Ubuntu Desktop will include a handy new ZFS feature. (more)

**Simple & Secure**

The first stable release of the Debian 11 series improves the speed of encrypted file transfers. (more)

**Linux**

Ubuntu's LXD is a good replacement candidate for the Docker-based Logstash and Kibana combination in Kubernetes environments. (more)

**Articles in the Hybrid Cloud**

Extending your data center temporarily into the cloud during a customer rush might not be easy, but it can be done, thanks to Amazon's S3 buckets and some AWS tricks. (more)

**Most Read Articles**

- **Linux Systems Vulnerable to Attack**  
Millions of Linux systems are vulnerable to attack, due to in-risk firmware. (more)
- **Windows Subsystem For Linux**

**Further Reading**

- **SystemD is Coming to a Linux Distribution Near You**
- **Software RAID**
- **Open Source Software Compresses the Enterprise**
- **Container Management with Podman**
- **Nine Year Old Bug Found and Fixed in Sudo**

**ADMIN HPC Close**

- **More Best Practices for HPC Containers**
- **15th Annual Women in HPC Workshop**
- **Workshop**
- **Quantum Internet Blueprint**
- **Quantum Computing with AWS Lambda**

**Further Reading**

- **High-Performance Python 4**
- **Trusting the Data Cache for Fun and Profit**
- **Linux Whitepapers**
- **The New OpenSSH Version 4**
- **ROCm 3.0 Arrives**

**GET PRODUCTIVE!**

**ORDER NOW!**

**101 COOL LINUX HACKS**

**Get to Know ADMIN**

**ADMIN HPC**

**Linux Update**  
EXPLORING THE WORLD OF LINUX

- **Secure Online Passwords**
- **South Korean Government Considers Move to Linux Desktop**
- **Microsoft Defender ATP is Coming to Linux**
- **Check**
- **Designing Blockchains with FPGAs**

**FEATURED ARTICLES**

- **Secure Online Passwords**  
Security storing passwords online can be a complex task. With a few simple techniques can offer better security, but users still need to choose their passwords wisely. (more)
- **South Korean Government Considers Move to Linux Desktop**  
South Korea's government is considering moving to Linux desktop. (more)
- **Microsoft Defender ATP is Coming to Linux**  
Microsoft is ramping up its efforts to fight malware across all platforms. (more)
- **Check**  
If you are looking for an application in AppImage, Flatpak or Snap app stores, Check how you perform a keyword-based search from the command line. (more)
- **Designing Blockchains with FPGAs**  
With LowCode Maker and Cofree, you can push your own ideas with better results than most online construction tools. (more)

**FURTHER READING**

- **Exploring the Sonoma Audio Player**
- **Mandatory Access Control with AppArmor**
- **Analysis (XSL) Usage with libxslt**
- **Linux Photo Tools**
- **Using Reading Functions in Bash**

**SOCAL LINUX EXPO**

March 5-6  
Frisco, CO  
Convent, CO  
Denver, CO

**MOST READ**

- **Little Long Range Radio**  
WiFi is convenient for devices that are in the same house. If you want to extend the distance, give LoRa a call. (more)
- **OpenSSH Now Supports FIDO/UAF Security Keys**  
SSH users can now add FIDO/UAF to the list of supported multi-factor authentication tools. (more)

**Raspberry Pi Geek Archive DVD**

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. Subscribe today for our excellent newsletters:

ADMIN HPC • ADMIN Update • Linux Update  
and keep your finger on the pulse of the IT industry.

ADMIN and HPC: [bit.ly/HPC-ADMIN-Update](https://bit.ly/HPC-ADMIN-Update)  
Linux Update: [bit.ly/Linux-Update](https://bit.ly/Linux-Update)

# LINUX NEWSSTAND

**Order online:**  
<https://bit.ly/Linux-Newsstand>

*Linux Magazine* is your guide to the world of Linux. Monthly issues are packed with advanced technical articles and tutorials you won't find anywhere else. Explore our full catalog of back issues for specific topics or to complete your collection.



**#239/October 2020**

## Build an IRC Bot

IRC bots do the essential work of coordinating and forwarding chat messages on the Internet. This month we show you how to build your own custom bot – and we give you an inside look at how to work directly with IRC.

**On the DVD:** Debian 10.5 and Devuan 3.0



**#238/September 2020**

## Speed Up Your System

Your Linux experience goes much more smoothly if your system is running at peak performance. This month we focus on some timely tuning techniques, including the kernel's new Pressure Stall Information (PSI) feature.

**On the DVD:** Bodhi Linux 5.1 and openSUSE Leap 15.2



**#237/August 2020**

## Webcams and Linux

This month we explore webcams, screencasting, and a cool teleprompter tool. We also look at PHP Building Blocks, Guacamole the clientless remote access tool based on HTML5, and a study of the handy MystiQ Audio/Video conversion tool.

**On the DVD:** Ubuntu Studio 20.04 and Kubuntu 20.04



**#236/July 2020**

## Smarter Directories

The rigid structure of nested files and directories used on computer systems around the world was created more than 60 years ago, and experts believe we can do better. This month, you'll learn about some scripts for semantic file tagging in Linux.

**On the DVD:** Fedora Workstation 32 and Ubuntu "Focal Fossa" Desktop 20.04 LTS



**#235/June 2020**

## What's New in Systemd

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

**On the DVD:** Knoppix 8.6.1 and OpenMandriva Lx Plasma 4.1



**#234/May 2020**

## Edge Computing

The Edge is a popular buzz word in high-tech news, but what does it mean really? We introduce you to an exciting new technology that could be changing the way we think about the cloud.

**On the DVD:** Manjaro 19.02 Gnome Edition and SystemRescueCd 6.1



# 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 <https://www.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](mailto:events@linux-magazine.com).

## NOTICE

Be sure to check the event website before booking any travel, as many events are being canceled or converted to virtual events due to the effects of COVID-19.

## The eighth year of All Things Open

**Date:** October 19-20, 2020

**Location:** Virtual Event

**Website:** <https://2020.allthingsopen.org/>

The eighth year of the All Things Open conference promises to be the best year yet as we will host a virtual event for the first time. Join 10,000 open source friends for two days focused on the tools, processes, and people that make open source possible.

## KubeCon + CloudNativeCon North America

**Date:** November 17-20, 2020

**Location:** Virtual Event

**Website:** [bit.ly/KubeCon-North-America](https://bit.ly/KubeCon-North-America)

KubeCon + CloudNativeCon North America is going online! The Cloud Native Computing Foundation's flagship conference gathers leading technologists from leading open source and cloud native communities to further the education and advancement of cloud native computing.

## Events

Arm Dev Summit	October 6-8	Virtual Conference	<a href="https://devsummit.arm.com/">https://devsummit.arm.com/</a>
World Summit AI	October 14	Online	<a href="https://worldsummit.ai/">https://worldsummit.ai/</a>
openSUSE + LibreOffice Conference	October 15-17	Virtual Conference	<a href="https://events.opensuse.org/">https://events.opensuse.org/</a>
All Things Open	October 19-20	Virtual Event	<a href="https://2020.allthingsopen.org/">https://2020.allthingsopen.org/</a>
Cloud Foundry Summit Europe	October 21-22	Virtual Event	<a href="https://bit.ly/cloud-foundry-summit">https://bit.ly/cloud-foundry-summit</a>
Open Source Summit Europe	October 26-29	Virtual Experience	<a href="https://bit.ly/open-source-europe">https://bit.ly/open-source-europe</a>
Embedded Linux Conference Europe	October 26-29	Virtual Experience	<a href="https://bit.ly/Embedded-Linux-Europe">https://bit.ly/Embedded-Linux-Europe</a>
AI Dev World	October 27-29	Virtual Conference & Expo	<a href="https://aidevworld.com/">https://aidevworld.com/</a>
KVM Forum	October 28-30	Virtual Experience	<a href="https://bit.ly/KVM-forum">https://bit.ly/KVM-forum</a>
Linux Security Summit	October 29-30	Virtual Experience	<a href="https://bit.ly/Linux-Security-Europe">https://bit.ly/Linux-Security-Europe</a>
Cloud Expo Europe	November 4-5	Frankfurt, Germany	<a href="https://www.cloudexpo-europe.de/">https://www.cloudexpo-europe.de/</a>
Data Centre World Frankfurt 2020	November 4-5	Frankfurt, Germany	<a href="https://www.datacentreworld.de/">https://www.datacentreworld.de/</a>
SC20	November 9-19	Virtual Event	<a href="https://sc20.supercomputing.org/">https://sc20.supercomputing.org/</a>
Open Source Strategy Forum (OSSF)	November 12-13	Virtual Experience	<a href="http://bit.ly/OS-Strategy-Forum">http://bit.ly/OS-Strategy-Forum</a>
KubeCon + CloudNativeCon North America	November 17-20	Virtual Experience	<a href="https://bit.ly/KubeCon-North-America">https://bit.ly/KubeCon-North-America</a>
Open Source Summit Japan	December 2-4	Virtual Experience	<a href="https://bit.ly/open-source-japan">https://bit.ly/open-source-japan</a>
DrupalCon Europe	December 8-11	Virtual Experience	<a href="https://events.drupal.org/">https://events.drupal.org/</a>

# 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](mailto:edit@linux-magazine.com).



## Authors

Paul Brown	84
Zack Brown	11
Bruce Byfield	22, 30
Joe Casad	3, 90
Mark Crutch	67
Chris Dock	34
Hans-Georg Eßer	14
Marco Fioretti	44
Jon "maddog" Hall	68
Charly Kühnast	53
Christoph Langner	74
Vincent Mealing	67
Martin Mohr	62
Graham Morrison	78
Mike Saunders	20
Mike Schilli	40
Mayank Sharma	26
Scott Sumner	54
Ferdinand Thommes	48, 70
Jack Wallen	8

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](http://www.linux-magazine.com/contact/write_for_us).

**NOW PRINTED ON** recycled paper from 100% post-consumer waste; no chlorine bleach is used in the production process.

## Contact Info

### Editor in Chief

Joe Casad, [jcasad@linux-magazine.com](mailto:jcasad@linux-magazine.com)

### Copy Editors

Amy Pettle, Megan Phelps

### News Editor

Jack Wallen

### Editor Emerita Nomadica

Rita L Sooby

### Managing Editor

Lori White

### Localization & Translation

Ian Travis

### Layout

Dena Friesen, Lori White

### Cover Design

Lori White

### Cover Image

© Maxim Basinski, 123RF.com

### Advertising

Brian Osborn, [bosborn@linuxnewmedia.com](mailto:bosborn@linuxnewmedia.com)  
phone +49 89 3090 5128

### Marketing Communications

Gwen Clark, [gclark@linuxnewmedia.com](mailto: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](mailto:cs@linuxpromagazine.com)  
Phone: 1-866-247-2802  
(Toll Free from the US and Canada)

For all other countries:  
Email: [subs@linux-magazine.com](mailto:subs@linux-magazine.com)

[www.linuxpromagazine.com](http://www.linuxpromagazine.com) – North America

[www.linux-magazine.com](http://www.linux-magazine.com) – Worldwide

While every care has been taken in the content of the magazine, the publishers cannot be held responsible for the accuracy of the information contained within it or any consequences arising from the use of it. The use of the disc provided with the magazine or any material provided on it is at your own risk.

Copyright and Trademarks © 2020 Linux New Media USA, LLC.

No material may be reproduced in any form whatsoever in whole or in part without the written permission of the publishers. It is assumed that all correspondence sent, for example, letters, email, faxes, photographs, articles, drawings, are supplied for publication or license to third parties on a non-exclusive worldwide basis by Linux New Media USA, LLC, unless otherwise stated in writing. Linux is a trademark of Linus Torvalds.

All brand or product names are trademarks of their respective owners. Contact us if we haven't credited your copyright; we will always correct any oversight.

Printed in Nuremberg, Germany by hofmann infocom GmbH on recycled paper from 100% post-consumer waste; no chlorine bleach is used in the production process.

Distributed by Seymour Distribution Ltd, United Kingdom

LINUX PRO MAGAZINE (ISSN 1752-9050) is published monthly by Linux New Media USA, LLC, 2721 W 6th St, Ste D, Lawrence, KS, 66049, USA. Periodicals Postage paid at Lawrence, KS and additional mailing offices. Ride-Along Enclosed. POSTMASTER: Please send address changes to Linux Pro Magazine, 2721 W 6th St, Ste D, Lawrence, KS 66049, USA.

Published monthly in Europe as Linux Magazine (ISSN 1471-5678) by: Sparkhaus Media GmbH, Ziebländstr. 1, 80799 Munich, Germany.

**Approximate**  
UK / Europe Nov 07  
USA / Canada Dec 04  
Australia Jan 04  
**On Sale Date**

Issue 241 / December 2020

# Harden Your System

The world is full of long discussions and exotic tools for locking down system security, but the journey can begin right now with simple and practical steps. Next month we show you the most important things you can do to keep your Linux system secure.



## Preview Newsletter

The Linux Magazine Preview is a monthly email newsletter that gives you a sneak peek at the next issue, including links to articles posted online.

Sign up at: <https://bit.ly/Linux-Update>

Lead Image © Nath Ting Feng, 123RF.com



# Hone your skills with special editions!

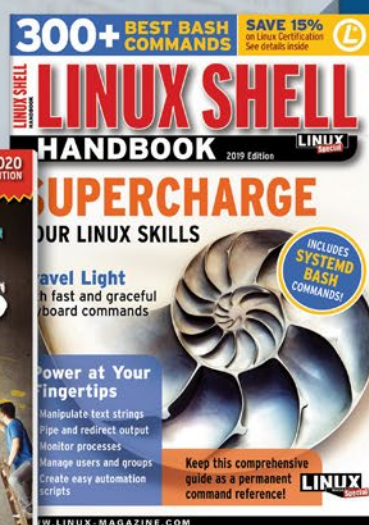
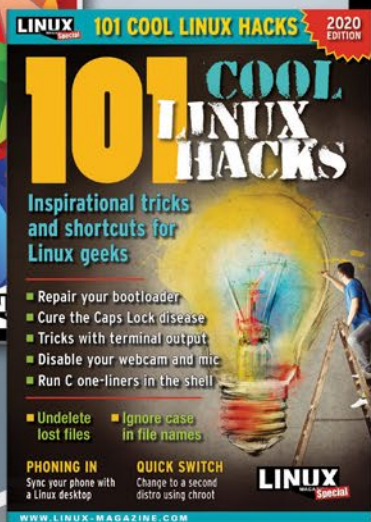
Get to know Shell, LibreOffice, Linux, and more from our Special Edition library.

The *Linux Magazine* team has created a series of single volumes that give you a deep-dive into the topics you want.

Available in print or digital format

**Check out the full library!**

[shop.linuxnewmedia.com](http://shop.linuxnewmedia.com)





# HETZNER

# MANAGED SERVERS DESIGNED FOR YOUR NEEDS



NEW

## e.g. Managed Server MX93

- ✓ Intel® Xeon® E5-1650 v2 Hexa-Core incl. Hyper-Threading Technology
- ✓ 64 GB DDR3 ECC RAM
- ✓ 2 x 500 GB SATA 6 Gb/s SSD (Software RAID 1)
- ✓ 100 GB Backup Space
- ✓ Traffic unlimited
- ✓ Location Germany
- ✓ No minimum contract
- ✓ Setup Fee \$105.50

monthly \$ **105.50**

## e.g. Managed Server MA120

- ✓ AMD EPYC 7401P 24-Core "Naples" (Zen) Simultaneous Multithreading
- ✓ 128 GB DDR4 ECC RAM
- ✓ 2 x 960 GB NVMe SSD (Software RAID 1)
- ✓ 100 GB Backup Space
- ✓ Traffic unlimited
- ✓ Location Germany
- ✓ No minimum contract
- ✓ Setup Fee \$141.00

monthly \$ **141.00**

All prices exclude VAT and are subject to the terms and conditions of Hetzner Online GmbH. Prices are subject to change. All rights reserved by the respective manufacturers.

[www.hetzner.com](http://www.hetzner.com)