

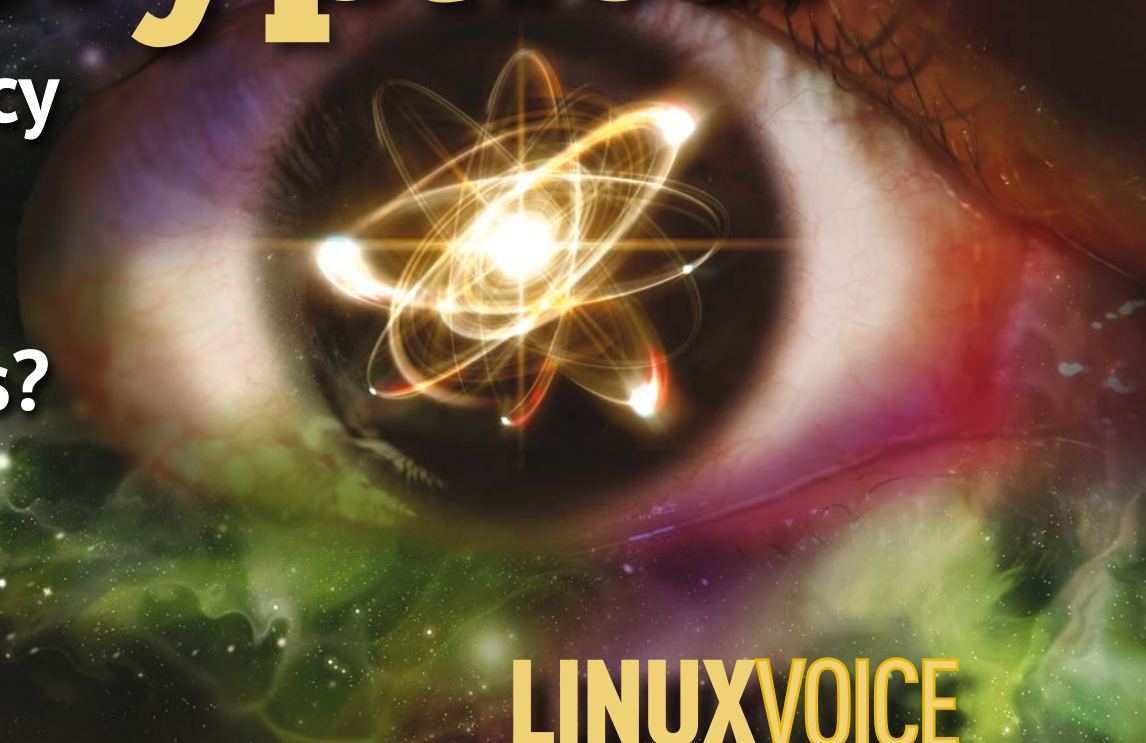


# LINUX PRO MAGAZINE

ISSUE 247 – JUNE 2021

# Post-Quantum Encryption

Will privacy survive quantum computers?



## LINUXVOICE

- maddog explores the history of text editors
- **Obsidian**: Nifty note and knowledge tool

**JSON Deep Dive**  
Get fluent in this important data format

**DTLS**  
Encryption over UDP

**10**

**FOSSPICKS**  
HEAVENLY FREE TOOLS!



LINUX NEW MEDIA  
The Pulse of Open Source



# LISA21

June 1–3, 2021 | Virtual Event  
[www.usenix.org/lisa21](http://www.usenix.org/lisa21)

LISA is the premier conference for operations professionals, where sysadmins, systems engineers, IT operations professionals, SRE practitioners, developers, IT managers, and academic researchers share real-world knowledge about designing, building, securing, and maintaining the critical systems of our interconnected world.

**Online registration and the full conference program are available now.**

Sign up to receive updates at [www.usenix.org/lisa/linuxpro](http://www.usenix.org/lisa/linuxpro), and follow @LISAconference on Twitter.



# NEWS FLASH: COBOL RISES FROM ITS OWN NON-ASHES

Dear Reader,

Of all the strange news this month, the one item that really caught my eye was the announcement of IBM's COBOL for Linux 1.1 compiler. I guess most of us are aware that millions of lines of business software around the world are still written in COBOL, and for that reason, old-school COBOL programmers still have work. But think about this for a minute: COBOL has existed for 62 years, the x86 architecture has been around for 43 years, and Linux has been with us for 30 years. Yet at this late date, the (97-year-old) IBM steps up now to announce a Linux COBOL compiler for the x86?

It is easy to see why IBM, whose clients include many old-school companies running old-school software, would continue to maintain their COBOL compilers. But a *new* compiler for x86 Linux? That's not just maintaining – that's investing!

Of course, COBOL has been counted out in the past. In fact, I can safely say that people have been counting out COBOL for as long as I have known about it. The Common Business Oriented Language (COBOL) dates back to the late 1950s, when a group of corporate and academic computer scientists got together to develop a coding language specifically designed to address the problems of business. The early birth of COBOL, and the fact that it was designed for efficient programming, not efficient execution, meant that it gained a reputation for being slow and inelegant. COBOL was a way to grind out steady and unglamorous business software. Some of that business software is still humming along, rolling out reports, inventories, and payroll. It could have cost millions to create, and it would cost even more now to replace, so, as the saying goes, "If it ain't broke, don't fix it."

Over the years, COBOL has seen many updates – to be fair, it really has evolved since 1959, with support for structured programming techniques, as well as XML and inter-language communication with C/C++. But it still has a reputation for being quite retro, and I guarantee that few students enroll in the world's leading computer science programs with the cherished goal of one day being a great COBOL programmer.

Clunky old software lives forever in its bits and bytes, but clunky old hardware fades away. Companies like IBM keep COBOL alive by continually providing a means to transition it to other hardware and operating systems. For years, IBM has supported COBOL compilers for z/OS and AIX – systems that

run on some of IBM's modern-day mainframe platforms. But why x86 and Linux? And why now?

Part of the answer is that the hardware of today is no hardware at all. Development environments increasingly live in the cloud, and virtual and containerized x86 systems are the stock and trade of cloud infrastructure. Another reason for supporting COBOL on the x86 could be a wager that, by lowering barriers to entry, they will eventually channel those x86-based users to the heavy mainframe systems that IBM is still selling. (They got rid of their PC division years ago.)

The other thing to remember is, IBM probably wouldn't be doing this if they didn't think they could make some money – or at least break even. Just because the COBOL x86 compiler runs on Linux doesn't mean it is free. According to the announcement, "COBOL for Linux on x86 is required on all systems on which COBOL applications are developed and where the applications are deployed and executed." You're supposed to "...consult your IBM representative or IBM Business Partner" for pricing.

Whatever the reasons, it does appear that the planets have aligned to give COBOL another burst of energy that could propel it even further beyond the long-past predictions of its demise.



Joe Casad,  
Editor in Chief



## ON THE COVER

### 34 DTLS

The connectionless UDP transport won't work with the popular TLS privacy protocol. DTLS is an alternative that will protect your UDP traffic from prying eyes.

### 38 JSON Deep Dive

JSON is a leading data format for websites, mobile devices, and even ordinary desktop applications. We take you down inside JSON format and describe some tools for working with JSON data.

### 56 Hands-On Guide to the GPIO

The GPIO is a powerful interface from your Raspberry Pi to outside hardware. We'll show you how to program for the GPIO.

### 88 Obsidian

Organize your thoughts with this cool knowledge app built on Markdown.

## NEWS

### 08 News

- Apple M1 Hardware Support To Be Merged into Linux Kernel 5.13
- KDE Launches the Qt 5 Patch Collection
- Linux Creator Warns Next Kernel Could Be Delayed
- System76 Updates Its Pangolin Laptop
- New Debian-Based Distribution Arrives on the Market
- System76 Releases New Thelio Desktop
- AlmaLinux Is Officially Available

### 12 Kernel News

- "Welcoming" a New Kernel Developer
- An Ancient Feature Goes Belly Up

## COVER STORY

### 16 Quantum Computing and Encryption

The encryption methods we use today are no match for tomorrow's quantum computers. We'll show you why and what's ahead for cryptography in the post-quantum era.

## REVIEW

### 22 Distro Walk – Knoppix

Knoppix, a portable operating system and rescue disk, continues to evolve.

## IN-DEPTH

### 26 distri

The experimental distri research project investigates ways to speed up package management.

### 30 Command Line – Installers

Testing cutting-edge applications may require learning about the installer first.

### 34 DTLS – Encryption for UDP

TLS encryption is wonderful if it is running over a reliable transport protocol like TCP; but if your needs call for the less reliable UDP transport, you'd better start learning about DTLS.

### 38 JSON Deep Dive

JSON data format is a standard feature of today's Internet. We'll take a close look at JSON format and explore some tools for reading and manipulating JSON data.

### 46 dgamelaunch

Set up a server to play Roguelike games and preserve a piece of gaming history.

### 51 Charly's Column – Zint

Doing a hardware inventory in a data center is never easy. Charly uses Zint to provide each newly acquired system with a QR code sticker.

### 52 Programming Snapshot – fsnotify

Inotify lets applications subscribe to change notifications in the filesystem. Mike Schilli uses the cross-platform fsnotify library to detect what's happening.

## 16 Post Quantum Encryption

Quantum computers are still at the experimental stage, but mathematicians have already discovered some quantum-based algorithms that will demolish the best of our current encryption methods. What better time to look for quantum encryption alternatives?

### MakerSpace

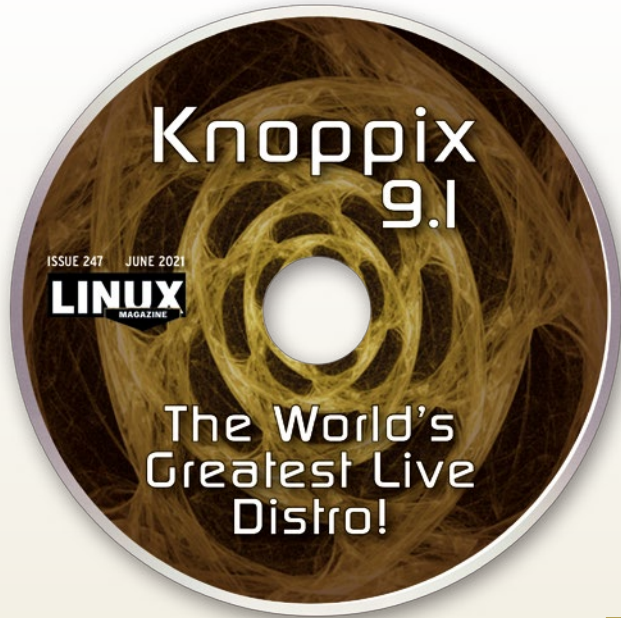
- 56 ARM64 Assembly and GPIO**  
Reading, writing, and arithmetic with the Raspberry Pi in ARM64 assembly language.
- 66 Pi OS 2020-12-02**  
The latest Raspberry Pi OS adds the PulseAudio sound server and a print manager.
- 70 Zenity**  
Add a dialog box to your command-line app.

### LINUXVOICE

- 73 Welcome**  
This month in Linux Voice.
- 74 Doghouse – Code Longevity**  
Maddog discusses the long history of text editors and the RAND message handling system.
- 75 EdUBudgie**  
EdUBudgie Linux is an Ubuntu clone created by a teacher and aimed directly at the education market.
- 78 Kit Scenarist**  
Creative writers take note! Kit Scenarist is a free application designed to simplify the process of writing a screenplay.
- 82 FOSSPicks**  
This month Graham looks at SonoBus, NewsFlash, Kinto.sh, RetroShare, Emilia Pinball, and much more!
- 88 Tutorial – Obsidian**  
Obsidian helps you work more effectively by giving you a tool to record, connect, and catalog your ideas and notes.



## Knoppix 9.1 and Zorin OS 15.3 Core Two Terrific Distros on a Double-Sided DVD!



### Knoppix 9.1 32- and 64-bit

Knoppix was created by Klaus Knopper, who has also written for *Linux Magazine*. It was one of the first Debian-derivatives, as well as among the first Live CDs. This month's download is a DVD containing 9GB of compressed software, but it is also available as a much smaller CD.

Knoppix was briefly installable like any disk image, but that option is no longer supported. Instead, Knoppix can be used as a portable operating system. However, it is best known as a rescue drive, with low memory requirements, extensive hardware support, and an assortment of recovery tools. These tools are especially useful on a large flash drive, with extra space for storing files created while running as a Live drive. Lesser-known but equally as impressive is ADRIANE Knoppix, a desktop environment for the sight impaired collected and developed by Klaus Knopper for his wife.

Knoppix is especially popular in Germany, and some of its documentation can take a while to be translated into English and other languages. However, the DVD is worth having around for emergencies and an essential tool for those who need extra assistance in order to interact with their computer. And if you are having trouble with Linux detecting your hardware, Knoppix just might help with your troubleshooting.



### Zorin OS 15.3 Core 64-bit

Zorin is a Debian-derivative known for its attention to aesthetics, like deepin or elementary OS. On the DVD, you'll find Zorin OS Core, the basic distribution, which will satisfy most users. However, if you like what you see here, you might explore Zorin OS 15.3 Lite, Zorin OS 15.3 Education, or the paid version, Zorin OS Ultimate, and decide which best suits your needs.

All these versions share common features. All are based on Ubuntu, with a default Gnome desktop designed to run without the overview page. The result is a very simple desktop environment, reminiscent of Gnome 2, and easy for any computer user to learn. The software is mostly Gnome technology. Should you choose to go beyond Zorin OS Core, you can also change the look of your desktop by imitating Windows, macOS, and other alternatives. All in all, Zorin OS Core is a polished beginner's distribution, allowing those familiar with proprietary operating systems an easy entry into the world of Linux.

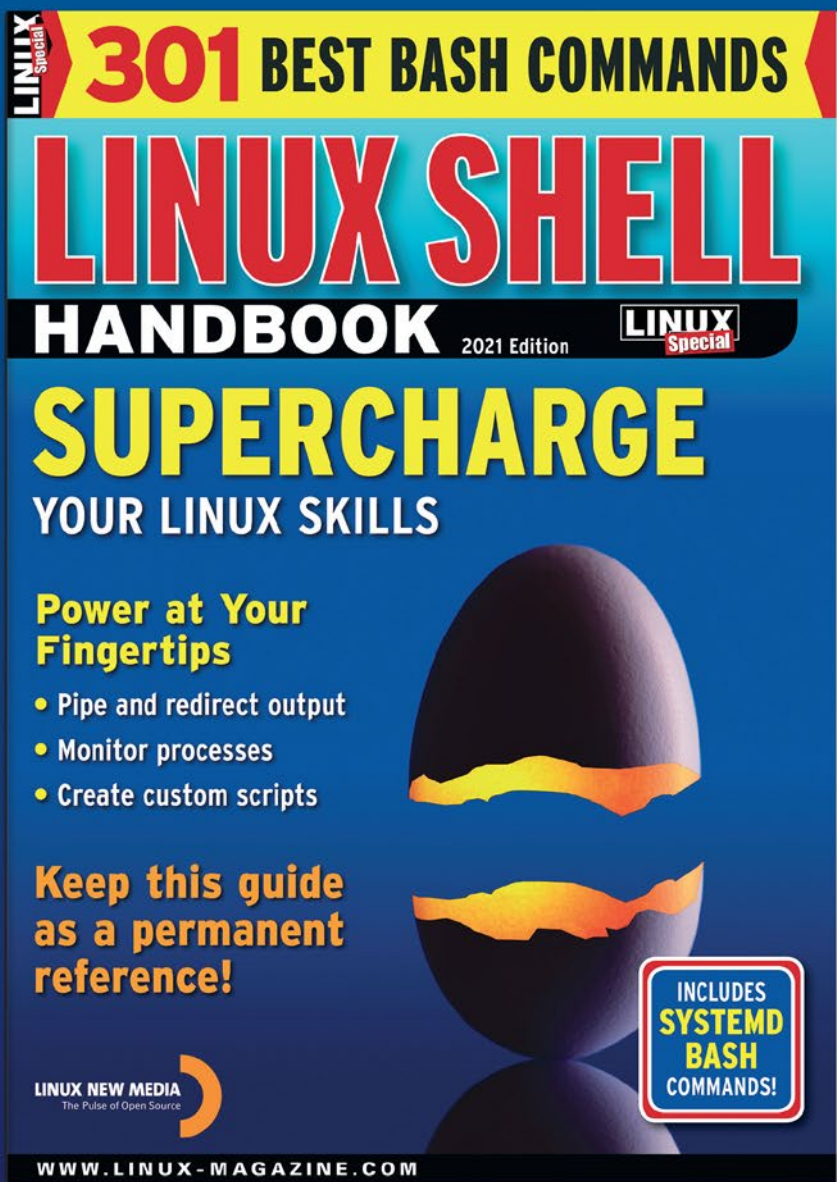
*Defective discs will be replaced.  
Please send an email to [subs@linux-magazine.com](mailto:subs@linux-magazine.com).*

*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.*

# THINK LIKE THE EXPERTS

Linux Shell Handbook 2021 Edition

This new edition is packed with the most important utilities for configuring and troubleshooting systems.



Here's a look at some of what you'll find inside:

- Customizing Bash
- Regular Expressions
- Systemd
- Bash Scripting
- Networking Tools
- 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 • Apple M1 Hardware Support To Be Merged into Linux Kernel 5.13
- KDE Launches the Qt 5 Patch Collection
- 09 • Linux Creator Warns Next Kernel Could Be Delayed
- System76 Updates Its Pangolin Laptop
- New Debian-Based Distribution Arrives on the Market
- More Online
- 10 • System76 Releases New Thelio Desktop
- AlmaLinux Is Officially Available

### Apple M1 Hardware Support To Be Merged into Linux Kernel 5.13

Hector Martin has merged the initial support for Apple M1 hardware into the Linux SOC (System On a Chip) tree. Martin is the founder of Asahi Linux, a project to port Linux to Apple Silicon Macs. The project was started in 2020, using the M1 Mac Mini, MacBook Air, and MacBook Pro hardware. The Asahi goal is “not just to make Linux run on these machines but to polish it to the point where it can be used as a daily OS.”

Now that M1 support has been merged into the tree, it should make it into the Linux kernel for the 5.13 release (which should come sometime this summer). That does not mean, however, you'll be able to run Linux on Apple Silicon this summer. In fact, at the moment there is no timetable for full support. The reason for this is porting Linux to Apple Silicon is a daunting task. Because Apple doesn't release any documentation for the M1 hardware, everything must be reverse-engineered and drivers must then be written.

But as of April 8, 2021, the arm/apple-m1 branch has been merged into Linux-next (the holding area for code expected for the next kernel merge window. To view the code that has been merged, take a look at this SOC commit (<https://git.kernel.org/pub/scm/linux/kernel/git/soc/soc.git/commit/?h=for-next&id=0d5fe4b31785b732b71e764b55cda5c8d6e3bbbff>). Although the Asahi Linux environment will now boot on the M1 hardware, it only provides serial and frame buffer console access. In other words, there's a long way to go. And, according to Martin, “we absolutely do not recommend buying M1 hardware for that purpose unless and until the Asahi project gets much, much farther down the road than it has managed so far.”

### KDE Launches the Qt 5 Patch Collection

At the end of 2020, Qt 6 was released to serve as the next-gen Qt application framework. This new iteration has made it possible to deliver more modern software and KDE has every plan to fully adopt this new release for the entire software stack.

However, KDE still very much relies on Qt 5 for both desktop and applications. With KDE's goal of migrating to Qt 6, they had to do something to ensure nothing falls by the wayside. To that end, KDE has decided (until Qt 6 adoption is finalized), to maintain a collection of patches for the Qt 5.15 release. These patches will include both security and standard fixes to make sure KDE continues to remain secure and stable.

Alex Pol, KDE e.V. President said of this, “To transition to great future technologies like Qt 6 we need to have the peace of mind that our current users are catered for. With this patch collection, we gain the flexibility we need to stabilize the status quo. This way we can continue collaborating with Qt and deliver great solutions for our users.”

As for Qt 6, the plan is to have support sometime in 2021.

To find out more about the KDE Qt 5 Patch Collection, read the official initiative (<https://community.kde.org/Qt5PatchCollection>). To find out where KDE stands with Qt 6, check out the Phabricator (<https://phabricator.kde.org/project/board/310/>).



## Linux Creator Warns Next Kernel Could Be Delayed

Never one to mince words, Linus Torvalds has released the latest RC (Release Candidate) of the Linux kernel, while expressing a slight bit of concern the size might hinder a timely release. Torvalds went so far as to say, "I'm not overly worried yet, but let's just say that the trend had better not continue, or I'll start feeling like we will need to make this one of those releases that need an rc8." Most Linux kernels go through 7 Release Candidates, which are made available every Sunday.

This is the same kernel that Torvalds warned users to avoid when the first release candidate was made available (which was delayed, due to an ice storm). To that, Torvalds said, "This merge window, we had a very innocuous code cleanup and simplification that raised no red flags at all, but had a subtle and very nasty bug in it: swap files stopped working right. And they stopped working in a particularly bad way: the offset of the start of the swap file was lost. Swapping still happened, but it happened to the wrong part of the filesystem, with the obvious catastrophic end results."

With that issue resolved, the Release Candidates continued, unabated. But due to the ballooning size, Torvalds has grown concerned about getting the final release out on time. If nothing of note happens (and the size doesn't grow out of hand), the 5.12 kernel should be released in late April or early May.

Read Torvalds' original statement about the 5.12 kernel size (<http://kml.iu.edu/hypermail/linux/kernel/2103.3/05182.html>).

## System76 Updates Its Pangolin Laptop

System76 does many things very well. One of those things is listen. The consumers have spoken and they wanted an AMD-powered version of their most popular laptop to date, the Pangolin. The Pangolin has the title of the first System76 laptop to be powered by the AMD Ryzen line of mobile processors. And with the Pangolin, you can configure a laptop all the way up to the Ryzen 7 4700 CPU and AMD Radeon graphics. As for memory and storage, the Pangolin can be spec'd up to 64GB of RAM and up to 2TB of NVMe storage.

The 15" laptop includes multi-colored backlit keys with tactile feedback, a large, multi-touch trackpad, and a 1080p matte display.

The Pangolin laptop sports 1 x USB 3.2 Gen 1 Type-A, 1 x USB 3.2 Gen 2 Type-C, 2 x USB 2.0 Type-A, and a MicroSD Card Reader. Other features include:

- 1.0M 720p HD webcam
- 49Wh Li-Ion battery
- Gigabit Intel Dual Band Wi-Fi 6 and Bluetooth 5
- Pop!\_OS operating system

Due to incredibly high demand, System76 has already sold out of their Pangolin stock, but the company has made it possible for those interested to sign up to be alerted (<https://system76.com/laptops/pangolin>) when the new inventory is ready for purchase.

The Pangolin starts at \$849.



## New Debian-Based Distribution Arrives on the Market

New Linux distributions pop up almost weekly, so it should come as no surprise that yet another Debian-based platform has hit the market. And like many others, TeLOS aims to be a bit different. How? Outside of being touch-screen friendly (using the KDE Plasma 5.20.5 desktop environment), TeLOS is ready to serve as your home theater center, thanks to the inclusion of the Kodi open-source media center software.

You'll also find a mixture of KDE and GNOME apps installed, giving you a sort of best-of-both-worlds on the desktop. On the KDE front, you'll find the standard

## MORE ONLINE

### Linux Magazine

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

### ADMIN HPC

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

#### Rethinking RAID (on Linux)

• Petros Koutoupis

Configure redundant storage arrays to boost overall data access throughput while maintaining fault tolerance.

### ADMIN Online

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

#### Secure Authentication with FIDO2

• Matthias Wübbeling

The FIDO and FIDO2 standard supports passwordless authentication. We discuss the requirements for the use of FIDO2 and show a sample implementation for a web service.

#### Identity and Access Management with OpenIAM

• Thorsten Scherf

Identity and access management plays a central role in modern IT infrastructures, with its local resources, numerous applications, and cloud services. We investigate how OpenIAM implements centralized user management.

#### Processing Streaming Events with Apache Kafka

• Kai Wähler

Apache Kafka reads and writes events virtually in real time, and you can extend it to take on a wide range of roles in today's world of big data and event streaming.

software included (such as Dolphin, Konsole, Okular, Ark, KTorrent, K3b, Kazam, KWrite, and KDE Connect). As for GNOME, the developers have added the Cheese webcam viewer, Disks disk utility, the Evolution groupware suite, the GNOME on-screen keyboard, the GNOME Clocks app, and the GNOME Sound Recorder.

But for open-source purists, TeLOS might have a couple of deal-breakers included, as both the Chrome web browser and the TeamViewer remote desktop software come pre-installed.

Be warned, TeLOS is pretty rough around the edges. Being a new distribution it has a long way to go before it's ready for the masses. And when you go to install the distribution, there is a password (ion) associated with the installer. Once installed, you'll also have to deal with a full-screen instance of the Chrome browser. Once you get past those things, you might find TeLOS an interesting take on the Linux desktop.

Download an ISO of the latest version of TeLOS (<https://sourceforge.net/projects/teლოსlinux/files/iso/>) and see if it might be your next Linux distribution of choice.

## System76 Releases New Thelio Desktop

System76 is best known for their high quality Linux laptops and their boundary-pushing desktops. And with their Thelio line of desktops designed and built-in house, with open source firmware, bios, and OS, they hold a very high appeal to Linux users.

The Thelio lineup is impressive and their newest entry to the lineup stands in between the base model, Thelio, and the next up, the Thelio Major. This new model, the Thelio Mira, has room for up to 2 GPUs, ships with either a 3rd or 4th gen AMD Ryzen, and can support up to 128GB of RAM. So even with a small footprint (17.18" x 9.96" x 13.03"), you're getting massive power to do important (or not so important) work.

Other standout specs include:

- NVIDIA Graphics (GT 1030, GTX 1650 Super: 1 x DisplayPort, 1 x HDMI, 1 x DVI, RTX 3070: 3 x DisplayPort, 1 x HDMI, Quadro RTX 4000: 3 x DisplayPort, 1 x DisplayPort over USB-C, Quadro RTX 5000: 4 x DisplayPort, 1 x DisplayPort over USB-C, Quadro RTX 6000: 4 x DisplayPort, 1 x DisplayPort over USB-C, Quadro RTX 8000: 4 x DisplayPort, 1 x DisplayPort over USB-C)
- Storage Up to 36TB, 2 x M.2 PCIe Gen4 NVMe and 4 x 2.5" SATA drives
- Rear Ports 1 x USB Type-C, with USB 3.2 Gen 2 support, 7 x USB 3.2 Gen 2, 1 x GbE RJ-45 port, 1 x 2.5G RJ-45 port
- Rear Audio 5 x Audio Jacks, 1 x Optical S/PDIF out

For more information, check out the official Thelio Mira (<https://system76.com/desktops/Thelio-mira>) page at System76.com.

## AlmaLinux Is Officially Available

AlmaLinux came into being after Red Hat decided to migrate CentOS to a pseudo-rolling release candidate. Unfortunately, several third-party tools (such as cPanel) decided to not support CentOS Stream, which left many admins and companies in the dark as to what they'd do.

That's when CloudLinux came to the rescue. CloudLinux was formed in 2009 to deliver a fork of RHEL/CentOS designed specifically for multitenancy hosting companies. But in 2020, the company realized they could pick up the pieces left behind by Red Hat.

Thus, AlmaLinux was born. This 1:1 RHEL binary compatible distribution is not only perfectly capable of being deployed in place of CentOS, you can even migrate your CentOS instances with a few quick commands. And, as of March 30, 2021, the official first release of AlmaLinux is available to download.

Of course, as if a drop-in replacement for CentOS wasn't enough, CloudLinux also formed a non-profit organization (AlmaLinux Open Source Foundation, <https://almalinux.org/>) dedicated to taking over the management of the distribution. And CloudLinux has committed a \$1 million annual endowment for the AlmaLinux project. So not only do users not have to worry that CloudLinux will take the same path Red Hat did with CentOS, AlmaLinux will have plenty of funding to continue operations for some time.

Download an ISO of this exciting new server distribution now ([https://repo.almalinux.org/almalinux/8/isos/x86\\_64/](https://repo.almalinux.org/almalinux/8/isos/x86_64/)).



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

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

**2020**  
Archives  
Available  
Now!

# 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.

<https://bit.ly/archive-bundle>

# Zack's Kernel News



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

By Zack Brown

## Author

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

## “Welcoming” a New Kernel Developer

Amy Parker, a newcomer to Linux kernel development, had an idea for a new filesystem. Ideally, she said, “once it’s completed, rich, and stable I’d try to get it into the kernel.” She asked what would be involved in such a process.

Andreas Dilger welcomed her and offered a few words of caution (a.k.a. doom ‘n’ gloom). First of all, he said, a new filesystem would need to have a unique value for users. If it only did something that other filesystems already did well, there wouldn’t be a need to include it in the source tree.

He added that filesystems had a particularly onerous burden of reliability, since users relied on them for their very lives. Unlike many software problems that could be fixed with a simple reboot, he said, if a filesystem lost user data, there was no way home. In light of this, Andreas added, “the general rule of thumb is 10 years before a new filesystem is stable enough for general use.”

Because of this, Andreas suggested that instead of writing a whole new filesystem, it could sometimes make more sense to take whatever idea Amy had in mind and add it to an existing filesystem, if that would be a good enough solution. He said, “Otherwise, users would have to stop using their existing filesystem before they started using yours, and that is a very slow process, because your filesystem would have to be much better at \*something\* before they would make that switch.”

In terms of Amy’s actual question about the process for submitting a new filesystem for consideration, Andreas said the first step would probably be to describe her idea and see if there were any existing filesystems that would be a better fit for those features.

Finally, after sufficient doom ‘n’ gloom had been dispersed, he concluded, “Note that I don’t want to discourage you from participating in the Linux filesystem development community, but there are definitely considerations going both ways wrt. [with regards to] accept-

ing a new filesystem into the kernel. It may be that your ideas, time, and efforts are better spent in contributing to an exciting project. It may also be that you have something groundbreaking work, and I look forward to reading about what that is.”

Amy thanked him for his feedback and said she’d think about all those things.

Meanwhile, Randy Dunlap suggested that Amy shouldn’t wait until her code was truly finished, but should release patches as she developed them, using a “release early, release often” philosophy. And Chaitanya Kulkarni suggested submitting patches and an overall description of the project as a Request For Comments (RFC), in order to get the discussion going.

Theodore Ts’o had some suggestions of his own:

*“File systems are also complicated enough that it’s useful to make the patches available via a git repo, and it’s highly recommended that you are rebasing it against the latest kernel on a regular basis.*

*“I also strongly recommend that once you get something that mostly works, that you start doing regression testing of the file system. Most of the major file systems in Linux use xfstests for their testing. One of the things that I’ve done is to package up xfstests as a test appliance, suitable for running under KVM or using Google Compute Engine, as a VM, to make it super easy for people to run regression tests. (One of my original goals for packaging it up was to make it easy for graduate students who were creating research file systems to try running regression tests so they could find potential problems – and understand how hard it is to make a robust, production-ready file system, by giving them a relatively well documented, turn-key system for running file system regression tests.)”*

And he concluded:

*“The final thing I’ll point out is that file system development is a team sport. Industry estimates are that it takes between 50 and 200 person-years to create a pro-*

duction-ready, general purpose enterprise file system. For example, ZFS took seven years to develop, starting with a core team of 4, and growing to over 14 developers by the time it was announced. And that didn't include all of the QA, release engineering, testers, performance engineers, to get it integrated into the Solaris product. Even after it was announced, it was a good four years before customers trusted it for production workloads.

"If you look at the major file systems in Linux: ext4, xfs, btrfs, f2fs, etc., you'll find that none of them are solo endeavors, and all of them have multiple companies who are employing the developers who work on them. Figuring out how to convince companies that there are good business reasons for them to support the developers of your file system is important, since in order to keep things going for the long haul, it really needs to be more than a single person's hobby."

Matthew Wilcox also had some advice of his own to offer. He said:

"Writing a new filesystem is fun! Everyone should do it.

"Releasing a filesystem is gut-churning. You're committing to a filesystem format that has to be supported for ~ever.

"Supporting a new filesystem is a weighty responsibility. People are depending on you to store their data reliably. And they demand boring and annoying features like xattrs, acls, support for time after 2038.

"We have quite a lot of actively developed filesystems for users to choose from already – ext4, btrfs, xfs are the main three. So you're going to face a challenge persuading people to switch.

"Finally, each filesystem represents a (small) maintainance burden to people who need to make changes that cross all filesystems. So it'd be nice to have a good justification for why we should include that cost.

"Depending exactly what your concept is, it might make more sense to make it part of an existing filesystem. Or develop it separately and have an existing filesystem integrate it."

Matthew concluded with some extra-doomy doom 'n' gloom, saying, "Anyway, I've been at this for twenty years, so maybe I'm just grouchy about new filesystems. By all means work on it and see if it makes sense, but there's a fairly low probability that it gets merged."

Rather than running for the hills, as I myself was at that moment doing out of sheer sympathy, Amy replied, "I'm bored and need something to dedicate myself to as a long-term commitment."

She thanked everyone for their advice. And since multiple people had suggested looking for existing filesystems to merge her idea into, she said she'd explore that possibility.

In response to Ted's suggestion that she should set up a Git repository, Amy replied that she had already been setting up the infrastructure for that.

As for Ted's further suggestion that Amy plan on doing some regression testing, Amy laughed into her sleeve, baiting him with a quote that actually came from Linus, "Regression testing? What's that? If it compiles, it is good; if it boots up, it is perfect." Though she immediately followed up with, "In all seriousness, though, yeah, already been planning for stuff like that."

And she remarked that she was already familiar with Ted's xfstests tool and had used it on a previous project.

And that was the end of the discussion.

So there you have it. Gone are the days of Linus Torvalds welcoming all comers with open arms, saying, "absolutely a filesystem would be a marvelous project, and here is the process for submitting patches; thank you for joining the community!"

Now it's, "hi, your project is utterly unrealistic, whatever it is, but we encourage you to give it a try anyway, sort of, not really, and in any case we're all so burnt out and bitter that we are sort of just speaking on autopilot. Welcome, whoever you are. The exit's over that way."

What conclusions can we draw? Is it possible that half-a-dozen big-time kernel hackers were simultaneously having a really bad day? Has COVID-19 fatigue caused a certain amount of brain atrophy or just outright depression? Or could it really be true that a newcomer can be told, sight-unseen, that their idea probably isn't really anything special and that she'd probably be better off working on someone else's project?

## An Ancient Feature Goes Belly Up

Way back in September, "when the grass was still green and the pond was still

wet and the clouds were still clean" (apologies to *The Lorax*), Linus Torvalds wrote, submitted, accepted, and applied a patch to remove the VGA soft scrollbar feature from the Linux kernel.

VGA soft scrollbar is what lets you scroll the console to see fleeting kernel messages as they flow past during bootup or crash down. You just hit Shift and Page Up to see whatever messages have scrolled past the top of the monitor.

Linus explained that VGA soft scrollbar, "turns out to have various nasty small special cases that nobody really is willing to fight. The soft scrollbar code was really useful a few decades ago when you typically used the console interactively as the main way to interact with the machine, but that just isn't the case any more."

Randy Dunlap said that with this patch going in, it should also be possible to remove the soft scrollbar documentation at the same time.

Linus also clarified the situation somewhat:

"Note that scrollbar hasn't actually gone away entirely – the original scrollbar supported by `_hardware_` still exists.

"Of course, that's really just the old-fashioned text VGA console, but that one actually scrolls not by moving any bytes around, but by moving the screen start address. And the scrollbar similarly isn't about any software buffering, but about the ability of moving back that screen start address.

"Do people use that? Probably not. But it wasn't removed because it didn't have any of the complexities and bitrot that all the software buffering code had.

"That said, I didn't check how much of the documentation is for the VGA text console, and how much of it is for the actual software scrollbar for fbcon etc. So it is entirely possible that all the docs are about the removed parts."

All seemed well until Pavel Machek cried out in anguish, "Could we pause this madness?"

Pavel went on to say:

"Scrollback is still useful. I needed it today... it was too small, so command results I was looking for already scrolled away, but... life will be really painful with 0 scrollbar.

"You'll need it, too... as soon as you get oops and will want to see errors just prior to that oops.

[...]

*“Kernel is now very verbose, so important messages during bootup scroll away. It is way bigger deal when you can no longer get to them using shift-pageup.*

*“fsck is rather verbose, too, and there’s no easy way to run that under X terminal... and yes, that makes scrollbar very useful, too.”*

Pavel put his money directly where his mouth was, saying, “If it means I get to maintain it... I’m not happy about it but that’s better than no scrollbar.”

Adam Borowski felt Pavel’s cry of pain. Adam unleashed his own tormented howl of “I concur,” lamenting, “this a serious usability regression for regular users.”

Adam pointed out that “without some kind of scrollbar, there’s no way of knowing why eg. your rootfs failed to mount (there was some oops, but its reason was at the beginning...). Or, any other problem the user would be able to solve, or pass the error messages to someone more knowledgeable.”

He also said to Linus:

*“I also wonder why did you choose to remove softscrollback which is actually useful, yet leave hardscrollback which doesn’t come to use on any non-ancient hardware:*

*\* on i386 there’s no vgacon at all*

*\* on x86, in-tree drivers for GPUs by Intel, nVidia and AMD (others are dead) default to switching away from vgacon*

*\* EFI wants its own earlycon*

*“... thus, the only niche left is nVidia proprietary drivers which, the last time I looked, still used CGA text mode.”*

Finally, to Pavel’s willingness to maintain the code in question, Adam remarked, “That’d be greatly appreciated. There are also some simplifications/re-writes that could be done, like getting rid of redundant 1-byte/4-byte storage (or even the code for 1-byte...). Hard scrollbar could be axed altogether (it provides only a small amount of scroll). Etc....”

Throwing his lot in with the rebellious Adam and Pavel, Maciej W. Rozycki confirmed that “For the record I keep using the console scrollbar all the time, and FWIW I have gone through all the hoops required to keep using VGA hardware emulation and its console text mode with my most recent laptop, which is a ThinkPad P51; no longer manufactured,

but still hardly an obsolete device by today’s standards I believe.” He therefore concluded that “no, it’s not that nobody uses that stuff anymore, and not with obsolete hardware either.”

At that point, the matter rested and several months passed. Then, as if no time whatsoever had passed, Phillip Susi replied to Pavel, “Amen! What self respecting admin installs a gui on servers? What do we have to do to get this back in? What was so buggy with this code that it needed to be removed? Why was it such a burden to just leave it be?”

To which Linus replied:

*“It really was buggy, with security implications. And we have no maintainers.*

*“So the scroll-back code can’t come back until we have a maintainer and a cleaner and simpler implementation.*

*“And no, maintaining it really doesn’t mean ‘just get it back to the old broken state’.*

*“So far I haven’t actually seen any patches, which means that it’s not coming back.”*

Phillip asked if there was any more information available. He said, “I can’t try to fix it if I don’t understand what is wrong with it. Are there any bug reports or anything I could look at?”

Meanwhile, Daniel Vetter was not going to let scrollbar return without a fight. In addition to the problems Linus had identified, Daniel said, “on anything that is remotely modern [...] there’s a pile more issues on top of just the scrollbar/fbcon code being a mess.” He continued:

*“Specifically the locking is somewhere between yolo and outright deadlocks. This holds even more so if the use case here is ‘I want scrollbar for an oops’. There’s rough sketches for how it could be solved, but it’s all very tricky work.*

*“Also, we need testcases for this, both in-kernel unit-test style stuff and uapi testcases. Especially the full interaction on a modern stack between /dev/fb/0, /dev/drm/card0, vt ioctls and the console is a pure nightmare.*

*“Altogether this is a few years of full time hacking to get this back into shape, and until that’s happening and clearly getting somewhere the only reasonable thing to do is to delete features in response to syzkaller crashes.”*

At this point, Greg Kroah-Hartman piled in, saying, “Along with what

Daniel has already pointed out, just look at all of the old syzbot reports for the code in this area. Try fixing one of those reports in an older kernel to give yourself an idea of the issues involved. Best of luck!”

Phillip was utterly unwilling to let this go, however. And when Geert Uytterhoeven offered some comments on the overall situation, Phillip said, “Judging from some of the comments in the code, it looks like you were one of the original authors of fbcon?” And Geert replied, “Indeed, a loooooong time ago....”

The two of them embarked on an implementation discussion. Phillip said he was willing to try to rewrite scrollbar from scratch if that was what it took, and he proposed some ideas about how to do that. And Geert replied:

*“There are multiple ways to implement scrolling:*

- 1. If the hardware supports a larger virtual screen and panning, and the virtual screen is enabled, most scrolling can be implemented by panning, with a casual copy when reaching the bottom (or top) of the virtual screen. This mode is (was) available on most graphics hardware with dedicated graphics memory.*

- 2. If a 2D acceleration engine is available, copying (and clearing/filling) can be implemented by rectangle copy/flip operations.*

- 3. Rectangle copy/flip by the CPU is always available.*

- 4. Redrawing characters by the CPU is always available.*

*“Which option was used depended on the hardware: not all options are available everywhere, and some perform better than others.”*

Several people joined the discussion, but no patches seemed to come out of it.

Reimplementing this feature seems, on the one hand, like something a fair number of people want badly enough to do just about anything for it and, on the other hand, like something that’s very hard to get right. And Linus doesn’t seem inclined to accept any patches that don’t actually get the thing right.

Will it come back? It seems like a fairly large mountain to climb for a feature that is only really useful for kernel developers debugging kernel code. And yet, it does seem to have a special place in the hearts of a fair number of those kernel developers. Time will tell. ■■■

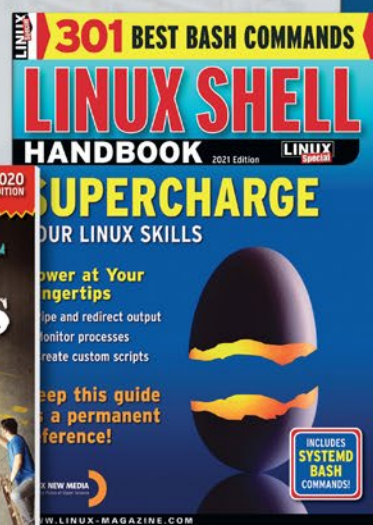
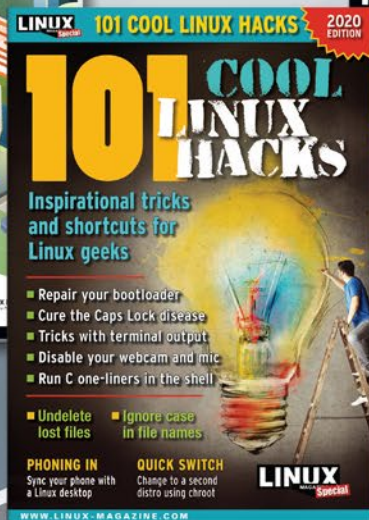
# Hone your skills with special editions!

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)





## Quantum computers and the quest for quantum-resilient encryption

# Entangled Secrets

The encryption methods we use today are no match for tomorrow's quantum computers. We'll show you why and what's ahead for cryptography in the post-quantum era.

By Stefan-Lukas Gazdag, Sophia Grundner-Culemann, Tobias Guggemos, Tobias Heider, and Daniel Loebenberger

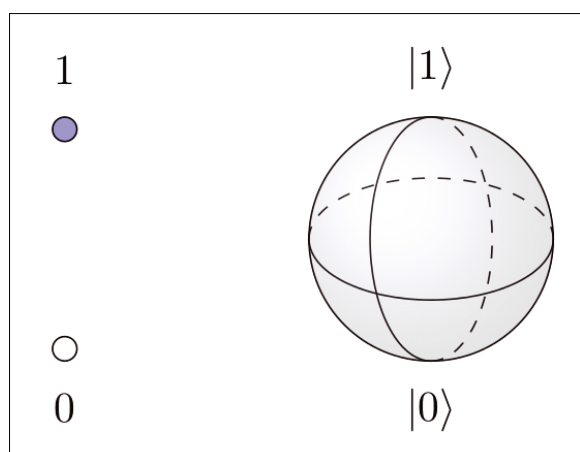
**E**ncryption is an everyday part of life on today's Internet. Encryption protocols facilitate virtual private networks (VPNs), protect corporate secrets, and validate banking transactions. Encryption is also the secret sauce behind technologies such as digital signatures and blockchain. The beauty of encryption is that, even if an observer intercepts the transmitted data, the original contents of the message remains hidden from view.

End users and corporations alike have come to depend on encrypted private communication over public networks, but many experts believe the way we think about encryption today will have to change if we want our secrets to stay secret. Cryptographers are looking ahead for a new form of encryption that will meet the needs of the post-quantum era.

## The Problem

A quantum computer is a computer that is designed to exploit the mysterious features of quantum mechanics. The basic unit of a conventional computer is binary (0 or 1). A quantum computer, on the other hand, is built around the quantum bit, or qubit, which assumes multiple states simultaneously.

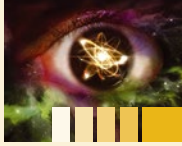
To fully understand the nuances of quantum computing [1], you would need a PhD in physics or computer science (or both), but as a quick illustration, Figure 1 shows a classical bit on the left with two states, zero or one. A qubit (right, represented by a Bloch sphere), can also represent values in between through what is known as superposition, such as a 1 with 65 percent probability and a 0 with 35 percent probability. If you mix up several qubits so that they influence each other, different results can occur, each with a specific probability. This strange but powerful feature lets quantum computers solve certain mathematical problems much more quickly than a conventional computer.



**Figure 1:** Classical bits (left) versus quantum bits (right).

Quantum computers have been theorized for many years, but the technology is still at the early stages of development. A few test systems exist today, but the kind of large-scale, production-ready quantum computing power necessary to implement the ideas discussed in this article are still a few years away. However, the experts believe the time of the quantum computer will come, and it make sense





Other encryption techniques rely on methods that mathematicians refer to with names like the *discrete logarithm problem* or the *elliptic-curve discrete logarithm problem*, both of which are also susceptible to attack using quantum techniques.

## Symmetric

Symmetric key encryption (where the data is encrypted and decrypted using the same key) is the most efficient means for achieving private communication on the Internet – if you can solve the key distribution problem. Even some procedures that begin with an asymmetric key exchange actually use symmetric encryption for communication and just employ an asymmetric process to communicate the symmetric session key. The most popular symmetric technique in use today is Advanced Encryption Standard (AES).

Symmetric encryption is subject to attack using quantum techniques; however, algorithms such as AES do appear to have some capacity to respond – at least in the near term. The most efficient generic attack by quantum computers on symmetric methods is the Grover algorithm, which was developed by Lov Grover in 1996. The Grover algorithm speeds up mindless brute-force checking of all possible keys. Classically, probability theory says that you have to test half of all possible keys on average. However, even in the worst case, the Grover algorithm requires no more tests than the square root of the number of all possibilities.

The Grover algorithm could thus reduce the bit security by half. In other words, a 128-bit encryption key would provide only the security level of 64-bit encryption in a quantum context. If you double the key length, you get back to the original security level – unattractive, but not technically difficult to implement. In practice, this means that wherever AES-128 is used, for example, you would need to upgrade to AES-256 for equivalent security; or, in the case of hash processes, you would need to upgrade from SHA-256 to SHA-512.

## Asymmetric

The security of asymmetric methods, on the other hand, is based on complex mathematical problems such as factoring large numbers or finding discrete logarithms. For sufficiently large numbers, this kind of encryption is practically impossible to solve using classical computers, but quantum computers will have a much better chance. Shor's algorithm provides exponential speed-up for several asymmetric encryption methods on a quantum computer. This speed-up allows the computation of keys of virtually arbitrary length in a meaningful amount of time and makes all widely used asymmetric algorithms, such as RSA, Diffie-Hellman, DSA, and variants based on elliptic curves (ECDH, ECDSA) vulnerable.

The potential obsolescence of the critical asymmetric algorithms that underpin today's Internet economy is one of the

for the IT industry to prepare.

A quantum computer could theoretically solve any problem you can solve with a conventional computer, but the theorized game-changing efficiency that speeds up solution by orders of magnitude will only work for certain types of problems that can be addressed using specialized quantum algorithms that exploit the power of superposition and qubits.

For example, some popular encryption methods use a key that is a product of prime numbers. The fact that the number 15 is the result of multiplying the two prime factors 3 and 5 is easy to deduce for humans as well as computers, but a number like the one in Listing 1 is difficult for even a computer to reduce to prime factors. The 309-digit number in Listing 1 once carried a US\$100,000 prize for anyone who could factor it. The contest ran from 2001 to 2007, and as of the time the contest ended, no one had claimed the prize [2]. Computer scientists have continued to work on the RSA Factoring Challenge numbers even after the contest ended, and to this day, no one has found the prime factors for the number in Listing 1, which is known as RSA-1024.

A message properly encrypted with a huge numbers like the one in Listing 1 poses a formidable challenge for a potential eavesdropper working with a conventional computer system. If you know the secret (in this case, the prime factors) the contents is quite trivial to decrypt, but if you don't know the secret, the message is effectively indecipherable.

However, encryption methods based on integer factorization problems of this type are far more vulnerable to attack using a quantum algorithm. According to cryptography experts, Shor's algorithm [3], which was created by the mathematician Peter Shor in 1994, could radically reduce the time needed to factor large numbers.

### Listing 1: RSA-1024 Challenge

```
1350664108659952233496032162788059699388814756056670275244851438515265106048595338
3394028715057190944179820728216447155137368041970396419174304649658927425623934102
0864383202110372958725762358509643110564073501508187510676594629205563685529475213
500852879416377328533906109750544334999811150056977236890927563
```



reasons why cryptographers are working ahead to ensure that quantum-resilient alternatives are in place before quantum computers emerge from the laboratory.

## Quantum-Resilient Alternatives

Production-ready quantum computers are still several years away. Stop-gap measures such as increasing key lengths and tweaking handshake procedures might work for a while, but eventually, the world will need a whole new class of encryption methods.

Today's security protocols are usually optimized for a specific procedure (e.g., a specific key exchange such as the Diffie-Hellman key exchange). Replacing this procedure was never intended; therefore, the protocol was not modularized accordingly. Designing the protocol around the encryption improves the efficiency of communication, and it also implicitly rules out certain attacks; however, the structure of the legacy protocol might complicate your efforts to migrate to a new encryption method.

Cryptographers are exploring several potential techniques for quantum-resilient encryption. One promising alternative is based on the concept of *lattices*. In a two-dimensional grid (Figure 2), it is comparatively easy to see which grid point (red) an arbitrarily set point (purple) is closest to. In a multidimensional grid, this becomes computationally very difficult. If the grid points symbolize all possible messages, a shifted point could represent the encrypted message. The receiver, who has precise knowledge of the grid, will find the next point with comparative ease. An attacker without sufficient information would have a hard time. Therefore, grids enable efficient encryption and signature procedures. Although it is very difficult to find reliable parameters, it is believed that this type of procedure is one of the most promising.

A quantum-resilient encryption method could also be built around error-correcting codes. McEliece's cryptosystem, for

example, has been known since 1978 and remains unbroken since then. However, the McEliece technique requires the exchange of a public key that weighs in at around 1MB, which has made the cryptosystem unattractive thus far. Other better alternatives exist for creating signatures: Hash-based signatures are already well understood and ready for use. One interesting area is isogenies [4] on supersingular elliptic curves [5] (not to be confused with classical cryptography based on elliptic curves). However, research on this approach has only been going for a few years, and so many questions still remain.

Some experts point to the benefits of a hybrid approach that would incorporate multiple procedures: If several of the procedures are used in parallel, the security of the overall system is based on all the procedures used, which spreads the risk if one of the methods is broken. In this context, experts speak of *crypto-agility*, a term that encompasses easy sharing of procedures, an easy way to respond to security incidents, and appropriate mitigation of incidents.

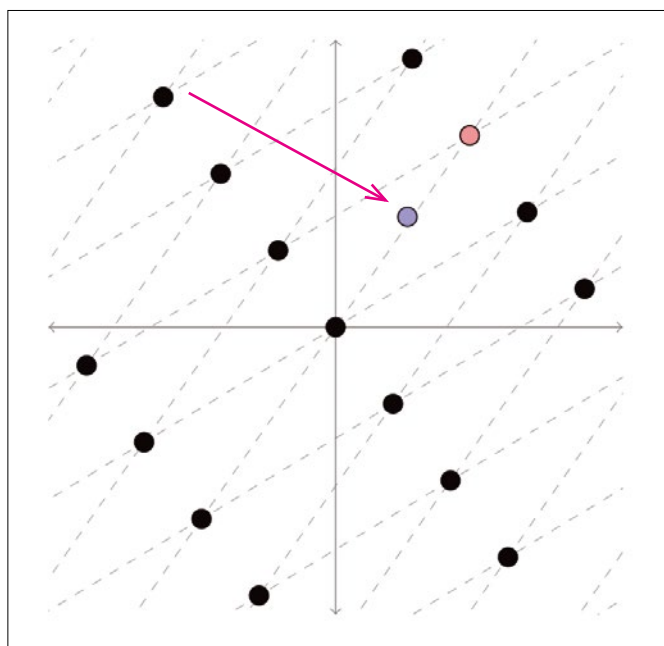
This hybrid approach seems relatively easy to implement for a task such as software updates. The update mechanism is modified such that, in addition to checking a classical signature, it also checks a second, quantum-resilient signature. The overhead is then limited to the additional signatures and verification times.

## VPNs

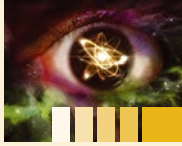
VPNs are an important part of today's networks, and encryption is essential to the privacy provided by a VPN. The protocols most commonly used with VPNs are not well prepared for attacks by quantum computers, and even some of the quantum-resilient alternatives are not well suited to use in VPNs. For example, the 1MB public key required by the quantum-resilient McEliece's method is 2,000 times bigger than what is currently required with today's protocols. This also affects software development – a programmer cannot allocate an arbitrary amount of memory on a system. In other words, software that was written for the data sizes that are common today (or in the past) may now need to be adapted for the new methods.

In the case of the IPsec VPN protocol, there is broad consensus that – as long as no suitably powerful quantum computer exists – you merely need to add methods that are quantum-resilient, but you do not yet need to replace the classical methods. Based on this approach, initial solutions have already been discussed for the Internet Key Exchange v2 (IKEv2) protocol, which is widespread in IPsec and laid down in the form of two Internet drafts currently under review. These drafts specify how additional messages can be exchanged in the protocol: The keys of most quantum-resilient alternatives turn out to be too large to be exchanged together with the underlying Diffie-Hellman exchange in a message without exceeding the maximum size of the initial message. (The maximum size is not just determined by the protocol but is also dependent on external parameters of the network, such as the Maximum Transmission Unit or MTU.)

Therefore, between the initial handshake, from which the connection is secured using a session key derived from Diffie-Hellman for AES, and the authentication step, you need to insert



**Figure 2:** Multidimensional lattices can serve as the starting point of quantum-resilient encryption schemes.



several messages for additional key negotiations. With the help of a quantum-resilient method (or several), additional secrets could be exchanged and the symmetric key could be updated. The connection would then be secured using at least two methods, which would also make it quantum-resilient.

But even that doesn't solve all the problems. The maximum size of an IKEv2 message is 64KB – too small for some procedures, including some McEliece configurations. Therefore, the key itself would have to be distributed over several messages and reassembled at the recipient's end by means of an internal counter. Possible solutions to this problem are currently in progress, but their security has not yet been conclusively demonstrated.

Quantum-resilient encryption poses not only protocol-related questions, but also methodological ones. The Diffie-Hellman key exchange, which is used in almost all common protocols today, lets both communication partners participate equally and with equal communication overhead in the process of creating the final symmetric key. This is not always the case with key-encapsulation mechanisms (KEMs), which are common in the field of post-quantum cryptography, because here the key is transmitted from one communication partner to the other. For McEliece, this means, for example, that a great deal of data travels in one direction and very little in the other. This relationship, which is unusual for the previous protocols, might be exploitable by attackers even without a quantum computer. Similarly, it is important to clarify how best to achieve properties such as forward secrecy using KEMs in protocols or use cases.

Note that these changes are occurring with protocols that were deliberately kept simple to eliminate or mitigate the effects of a few specific attacks. Implementing hybrid or crypto-agile solutions dials up the complexity, which could make the protocol susceptible to other sorts of attacks, such as denial-of-service attacks. It would therefore be important for the developer to provide other safeguards, such as checking to see if an attacker is trying to fill the receiver's memory with unnecessary data disguised as key material.

## Next Steps

To enable adaptation of the existing infrastructure with all its protocols and the software on a wide variety of systems, standardization bodies are already addressing the problem. The challenge is to find a consensus on how to extend the protocols to communicate both practically and securely in the future. New solutions are needed in some cases, especially since the focus is now on the modularity of the mechanisms and hybrid application of the procedures.

The problem can definitely be solved. The US National Institute of Standards and Technology (NIST) is currently elaborating a standardization process [6]. Organizations like the German Federal Office for Information Security (BSI) advise migration and recommend quantum-resistant algorithms. However, it will probably be years before the standards are adopted and established in practice. For highly sensitive use cases, if you wait for the standard, you might already be too late.

## Conclusions

Recent estimates suggest that it takes about 20 million qubits to break a 2048-bit RSA key. Currently, the most powerful quantum

computers operate at about 70 qubits, and even optimistic estimates expect at most 1,000 qubits within the next three to five years. This means that current quantum computers are not nearly powerfully enough to break today's encryption algorithms. It is unclear if and when a cryptographically relevant quantum computer might exist.

However, if the pace of quantum development observed in recent years continues, the future could hold a realistic threat to secure communications. Some government intelligence agencies have already intercepted and stored vast volumes of encrypted data and might be able to start deciphering that data as soon as there are advances in crypto analysis or as soon as suitable quantum computers become available. This means that different encryption solutions are already useful today even if quantum decryption is not yet available.

At the end of the day, the problems related to quantum-resilient encryption are solvable, but there can be no doubt that the alternatives will not turn out to be as efficient and simple as the classical procedures.

When you migrate to quantum-resilient IT, it will be important to document where cryptography is used in your own enterprise – or in your own products. Then you can explore whether alternative options are available. Dependencies will always crop up, meaning that product manufacturers or open source projects will have to make appropriate adjustments. For example, the question might arise as to whether the crypto libraries used with the project can be replaced by quantum-resilient alternatives, or whether high-security requirements already force transitional solutions.

Some standardization bodies are at least trying to offer the option of securing today's communications against quantum computers without the use of post-quantum cryptography – for example, with the help of pre-shared keys.

The multitude of requirements makes a universal recommendation impossible. Only the use of experts and a broad exchange of knowledge can prevent isolated solutions, which – unfortunately – often occurred in the past. It is important to cover the widest possible range of use cases with the smallest possible number of standard solutions. Crypto and IT security experts rely on the experience of software and hardware developers, as well as administrators of small and large networks. This real-world testing with large volumes of data is the best way to identify vulnerabilities. Government agencies, standards bodies, and the crypto community need this input, and they welcome the participation of all of us in preparing tomorrow's digital world for the requirements of the near future. ■■■

## Info

- [1] Quantum computing: [https://en.wikipedia.org/wiki/Quantum\\_computing](https://en.wikipedia.org/wiki/Quantum_computing)
- [2] RSA Factoring Challenge: [https://en.wikipedia.org/wiki/RSA\\_Factoring\\_Challenge](https://en.wikipedia.org/wiki/RSA_Factoring_Challenge)
- [3] Shor's algorithm: [https://en.wikipedia.org/wiki/Shor%27s\\_algorithm](https://en.wikipedia.org/wiki/Shor%27s_algorithm)
- [4] Isogeny: <https://en.wikipedia.org/wiki/Isogeny>
- [5] Supersingular elliptic curves: [https://en.wikipedia.org/wiki/Supersingular\\_elliptic\\_curve](https://en.wikipedia.org/wiki/Supersingular_elliptic_curve)
- [6] NIST PQC mailing list: <https://csrc.nist.gov/projects/post-quantum-cryptography/email-list>

**ARCHIVE DVD** FREE DVD (\$29.90 VALUE!) **RASPBERRY PI GEEK** THE COMPLETE ARCHIVE 2,000 pages of maker projects and more!

# LINUX MAGAZINE

APRIL 2020

## STREAM TO YOUR TV

Cool tools for sound and video

Analyze File Metadata

Apache Cassandra Database for the Internet age!

Web Scraping

**NAVI** Interactive cheat sheet for the shell commands

**EDGE COMPUTING**

Get ready for the low-latency cloud

Bash Scraping Gather, parse, and filter web data with Bash

**FREE DVD** SystemRescueCd 4.4

# LINUX MAGAZINE

ISSUE 214 - MAY 2020

## EDGE COMPUTING

Reinventing home directories with the powerful systemd-homed

Vagrant Create a custom UI for managing virtual machines

Glances Keep up-to-date on system health

PlantUML Create diagrams using human-readable text

"The world around us may be going through strange times, but at least so far kernel development looks normal." - Linus Torvalds (see Kernel News)

Tangram Manage your social media accounts from the terminal

BerryLAN Set up a headless Rasp Pi server

**FREE DVD** OPENMANDRIVA LINUX 5.2

**DOUBLE-SIDED DVD INSIDE!** Knoppix 8.6.1

# LINUX MAGAZINE

ISSUE 220 - JUNE 2020

## WHAT'S NEW IN SYSTEMD

Reinventing the file storage metaphor with semantic tagging

OpenCart Build a secure online store

Design an Ebook with Free Tools

Monitoring Tricks Build a dashboard to keep watch on your old Linux computer

Water Your Plants Tending your garden with a Rasp Pi Zero

ProjectLibre Organize and visualize your next project

baobab All-in-one tool for managing Flatpak and AppImage packages

Helium Lean Linux distro for 32-bit systems

\* sncli: Sync and manage your notes

\* maddog: It's Time to Innovate

CALL IN WITHOUT LOCK-IN Get connected with the Jitsi videoconferencing tool

SPEED UP YOUR SYSTEM HOT TWEAKS FOR A FASTER LINUX

Steampunk Laptop A little copper tubing and a Raspberry Pi

Scary Tech This Halloween vending machine dispenses chocolate treats

Easy Tips for Optimizing Shell Scripts

Repository Arch 120 package repositories at once

Scary Tech This Halloween vending machine dispenses chocolate treats

**FREE DVD** fedora 32

**DOUBLE-SIDED DVD INSIDE!** ubuntu 20.04 LTS

# LINUX MAGAZINE

ISSUE 230 - JULY 2020

## SMARTER DIRECTORIES

Reinventing the file storage metaphor with semantic tagging

OpenCart Build a secure online store

Design an Ebook with Free Tools

Monitoring Tricks Build a dashboard to keep watch on your old Linux computer

Water Your Plants Tending your garden with a Rasp Pi Zero

ProjectLibre Organize and visualize your next project

baobab All-in-one tool for managing Flatpak and AppImage packages

Helium Lean Linux distro for 32-bit systems

\* sncli: Sync and manage your notes

\* maddog: It's Time to Innovate

CALL IN WITHOUT LOCK-IN Get connected with the Jitsi videoconferencing tool

SPEED UP YOUR SYSTEM HOT TWEAKS FOR A FASTER LINUX

Steampunk Laptop A little copper tubing and a Raspberry Pi

Scary Tech This Halloween vending machine dispenses chocolate treats

Easy Tips for Optimizing Shell Scripts

Repository Arch 120 package repositories at once

Scary Tech This Halloween vending machine dispenses chocolate treats

**FREE DVD** Bodhi Linux 5.1

**DOUBLE-SIDED DVD INSIDE!** Bodhi Linux 5.2

# LINUX MAGAZINE

ISSUE 228 - SEPTEMBER 2020

## SPEED UP YOUR SYSTEM

Hot tweaks for a faster Linux

PSI Discover the kernel's cool new performance monitoring feature

Easy Tips for Optimizing Shell Scripts

Repository Arch 120 package repositories at once

Scary Tech This Halloween vending machine dispenses chocolate treats

Easy Tips for Optimizing Shell Scripts

Repository Arch 120 package repositories at once

Scary Tech This Halloween vending machine dispenses chocolate treats

**FREE DVD** kubuntu 20.04 LTS

**DOUBLE-SIDED DVD INSIDE!** ubuntu 20.04 LTS

# LINUX MAGAZINE

ISSUE 227 - AUGUST 2020

## MS AND LINUX

Hardware wrinkles and QtCAM

Imaginary Teleprompter Polish up your video presence

Kitchen Timer Turn a Pi Zero into a DIY kitchen device

WIREGUARD Secure and simple VPN tool

WEBCAMS AND LINUX

**FREE DVD** debian 10.5

**DOUBLE-SIDED DVD INSIDE!** ZEVUAN

# LINUX MAGAZINE

ISSUE 239 - OCTOBER 2020

## BUILD AN IRC BOT

Customize a GTK3

DISPATCHER SCRIPTS Automate NetworkManager seamless location changes

Kernel Lockdown Mod Rein in root with this powerful new Linux security feature

Build a WiFi Sound System With a Rasp Pi speakers from

GARDEN TRICKS USE LONG-RANGE RADIO TO MONITOR YOUR PLANTS

BUILD AN IRC BOT

**FREE DVD** Kali Linux 2020.1

**DOUBLE-SIDED DVD INSIDE!** Kali Linux 2020.2

# LINUX MAGAZINE

ISSUE 238 - SEPTEMBER 2020

## SPEED UP YOUR SYSTEM

Hot tweaks for a faster Linux

PSI Discover the kernel's cool new performance monitoring feature

Easy Tips for Optimizing Shell Scripts

Repository Arch 120 package repositories at once

Scary Tech This Halloween vending machine dispenses chocolate treats

Easy Tips for Optimizing Shell Scripts

Repository Arch 120 package repositories at once

Scary Tech This Halloween vending machine dispenses chocolate treats

**FREE DVD** fedora 33

**DOUBLE-SIDED DVD INSIDE!** ubuntu 20.04 LTS

# LINUX MAGAZINE

ISSUE 242 - JANUARY 2021

## 3D PRINTING

From first steps to a finished creation

MOFO Linux Steer around censorship with this privacy-focused distro

Nyttig Share files and stream Internet radio from a Rasp Pi

Netdata Zero configuration monitoring tool

Nerf Dart Game Score points for your next neighborhood party

Costs and Benefits Choose the best alternative with the TOPSIS decision technique

GO PROGRAMMING Search for files on Google Drive

3D PRINTING

**20 YEARS ARCHIVE DVD** HUGE SAVINGS! \$29.90 VALUE!

**20TH ANNIVERSARY ISSUE** ALL 239 ISSUES ON A SEARCHABLE DVD

# LINUX MAGAZINE

ISSUE 240 - NOVEMBER 2020

## 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

Build a WiFi Sound System With a Rasp Pi speakers from

hub: A command-line tool for GitHub repository

maddog: Remembering the Linux PulseEffects: Equalizer and the PulseAudio

Tuxedo InfinityBook Light and lively Linux laptop

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

**FREE DVD** elementary OS

**DOUBLE-SIDED DVD INSIDE!** KDE neon 5.20.0

# LINUX MAGAZINE

ISSUE 241 - DECEMBER 2020

## SECURE YOUR SYSTEM

Expert tips for protecting your Linux

Clonezilla SE Mass cloning without the mess

Managing Servers with Cockpit

RebornOS Arch for the slightly less geeky

Christmas Tinkering Build a holiday music box

Git Utilities Cool tools for extending Git version control

Fun with Go Display song lyrics line by line

JAM.PY BUILD A WEB FRONT END FOR YOUR DATABASES

**FREE DVD** ubuntu 20.04 LTS

**DOUBLE-SIDED DVD INSIDE!** ubuntu 20.04 LTS

# LINUX MAGAZINE

ISSUE 242 - JANUARY 2021

## 3D PRINTING

From first steps to a finished creation

MOFO Linux Steer around censorship with this privacy-focused distro

Nyttig Share files and stream Internet radio from a Rasp Pi

Netdata Zero configuration monitoring tool

Nerf Dart Game Score points for your next neighborhood party

Costs and Benefits Choose the best alternative with the TOPSIS decision technique

GO PROGRAMMING Search for files on Google Drive

3D PRINTING

FOSSPicks

- DGIS spatial data environment
- PrettyEQ audio equalizer

Tutorial

- Track your investments with Portfolio Performance

www.LINUX-MAGAZINE.COM

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

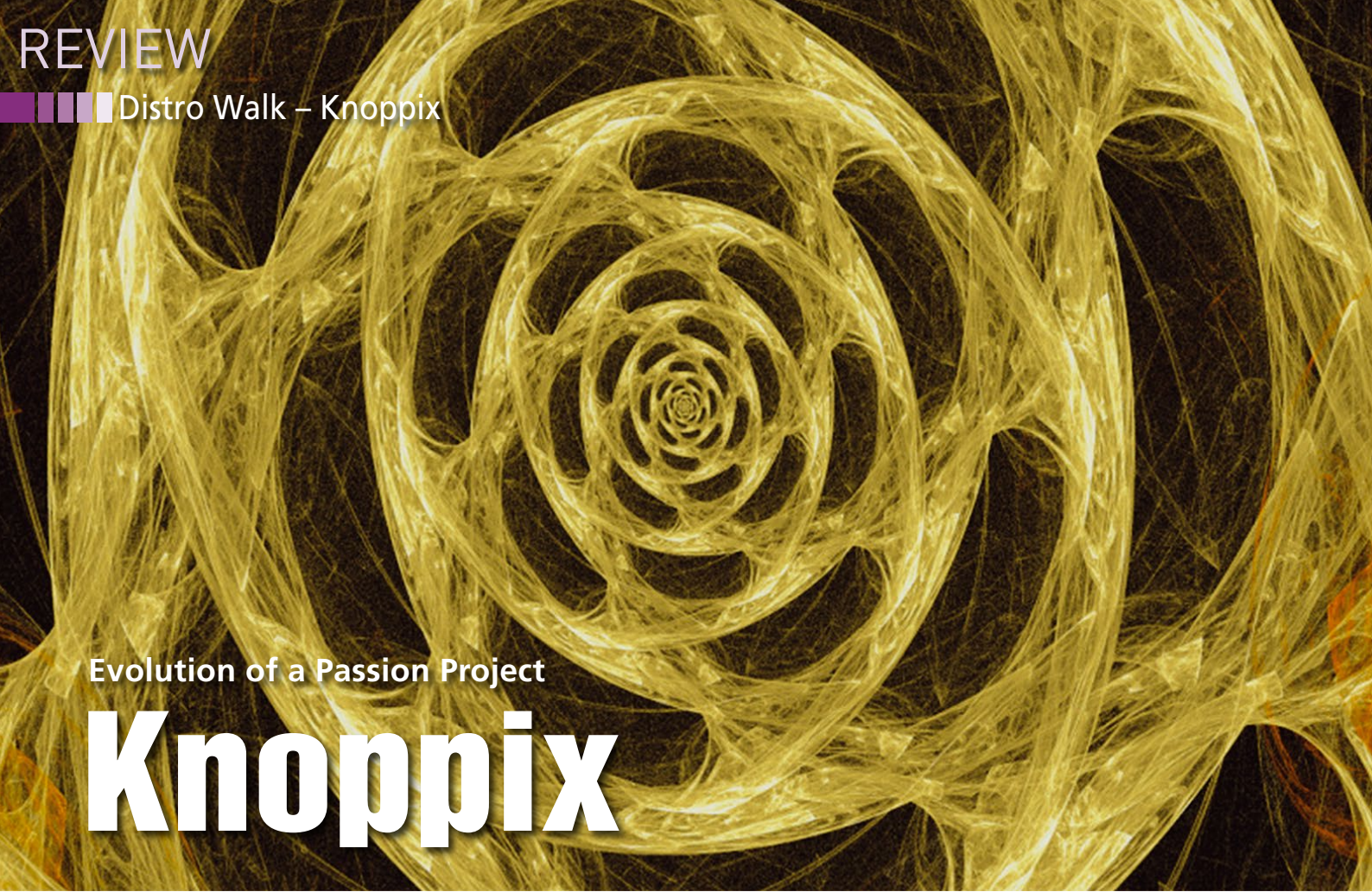
## Expand your Linux skills with:

- 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)





Evolution of a Passion Project  
**Knoppix**

**Knoppix, a portable operating system and rescue disk, continues to evolve.** *By Bruce Byfield*

**F**or over 20 years, Knoppix has been the premier portable operating system and rescue disk for Linux users. Although its packages are drawn from the Debian repositories, and contributors add to its hardware support, the bulk of the work on the distribution is done by German electrical engineer Klaus Knopper (Figure 1), an independent consultant and instructor at the Kaiserslautern University of



**Figure 1:** Klaus Knopper, Knoppix's core developer.

Applied Sciences. Over the years, Knoppix's hardware support has increased, features have been added and dropped, and its original purposes have been joined by ADRIANE (Audio Desktop Reference Implementation and Networking Environment), a desktop designed for the sight impaired with input from Klaus's wife, Adriane Knopper (Figure 2). All of which shows how this passion project is evolving with the times and is as important as ever.

"When I started studying electrical engineering in the late '80s," Klaus Knopper says, "my plan was to build electric cars and solar power plants. Apparently, this vision was just a little too early, so my interests turned more towards networking software and the possibilities that appeared with the Internet." At the time, free software was how Unix-like systems were taught, with ex-

ercises done with the GNU shell and compiler collection, using the GNU/Linux, BSD, and Hurd kernels. Knopper was among the students who founded a Unix working group, which eventually went on to organize the LinuxTag expo.

A few years later, Knopper encountered the Linuxcare Rescue CD, an 18MB business-card-sized CD with the Linux kernel and a command-line rescue tool. Knopper recalls, "I thought that a full CD-sized operating system with all the applications I use frequently, including a



**Figure 2:** Adriane Knopper, the inspiration for ADRIANE, Knoppix's desktop for the blind.

graphical desktop, data forensics, and TeX and other favorites would be extremely practical for travelling without a computer, using publicly available computers while still being able to use my personal software collection without installation.”

A Knoppix prototype debuted at the Atlanta Linux Showcase in 2000. After talking about his efforts, Knopper gave out several dozen CDs plus, a few weeks later, feedback about hardware detection. “At that point,” he says, “I decided that Knoppix should be a publicly available project for further development, because I could not possibly test on every available computer hardware constellation on my own. Detecting hardware correctly, and creating appropriate configuration for optimal support without any questions asked interactively is probably the biggest technical challenge.”

Today, Knoppix is downloaded 7,000 to 20,000 times per day from Knopper’s own website [1]. Although Knopper does not have statistics from mirrors, he notes that some Internet providers recommend downloading a Knoppix DVD just for measuring real bandwidth.

## Hardware Detection

From the start, Knoppix’s main challenge has been hardware support (Figure 3). However, support has become easier thanks to packages like the Linux kernel and `udev`, which have built-in detection for many types of hardware. “Writing scripts that probe a few thousand setup options is no longer necessary,” Knopper says. “One challenge left is finding the correct chipset to use in dual-chipset graphics on some boards. Depending on vendor and BIOS setup, only one of the tandem chipset parts works reliably, so there is only a 50/50 chance for finding the working setup for these automatically. As an example, from my experience, with the frequently installed Intel + NVidia combination, the Intel part works better out of the box, but in some cases booting with `knoppix64 xmodule=nouveau` is required to select the NVidia art instead.”

Knopper continues, “for only partly Linux-supported hardware, fallbacks to generic drivers are included. But these only come into action if the native drivers exist cleanly. Sometimes you

```
Linux Kernel 5.10.10-64, 1989 MB RAM.
CPU 0: AMD Opteron 63xx class CPU @ 4000MHz, 512 KB Cache
CPU 1: AMD Opteron 63xx class CPU @ 4000MHz, 512 KB Cache
CPU 2: AMD Opteron 63xx class CPU @ 4000MHz, 512 KB Cache
CPU 3: AMD Opteron 63xx class CPU @ 4000MHz, 512 KB Cache
CPU 4: AMD Opteron 63xx class CPU @ 4000MHz, 512 KB Cache
CPU 5: AMD Opteron 63xx class CPU @ 4000MHz, 512 KB Cache
CPU 6: AMD Opteron 63xx class CPU @ 4000MHz, 512 KB Cache
CPU 7: AMD Opteron 63xx class CPU @ 4000MHz, 512 KB Cache
Knoppix 9 found at: /dev/sr0
>>> Starting in Live-Mode. <<<
>>> Please do not remove medium until shutdown! <<<
Graphics 0: Red Hat, Inc. Virtio GPU
Sound 0: Intel Corporation 82B01FB/FB/FW/FW (ICH6 Family) High Definition A
udio Controller
[mod] [cg] [udev]
System Setup...
Detecting hardware [graphics, input]...
```

**Figure 3:** Knoppix’s hardware detection is both quick and extensive, although it sometimes needs the help of boot options.

have the case that an older Knoppix version works well on a specific graphics card using the `vesa` or `framebuffer` drive while a newer version which has a new experimental driver won’t, but can still be booted with the option `knoppix xmodule=vesa` to enforce the simplest driver.”

## Streamlined Features

In the past, some users have installed Knoppix as their main operating system, but Knopper warns that it is not a typical distro with frequent upgrades and multi-user setups. Nor does it work well with UEFI or upgrades. Because of these problems, the hard disk installer was removed in Knoppix 9. Instead, Knopper advises working from a flash drive, with the `/home` directory on a separate filesystem, so that upgrades do not cause problems. The exception is the `update-security` script, which according to Knopper can safely replace most packages.

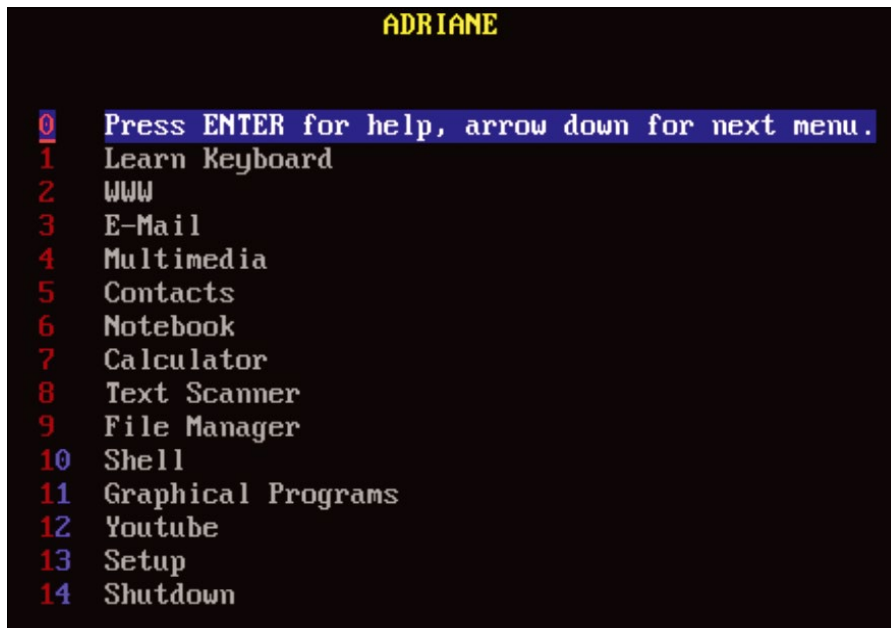
Another feature dropped from Knoppix is `systemd`. In fact, Knoppix was one of the first major distributions to remove `systemd`. “I try to keep the startup procedure as simple as possible, driven by shell scripts and only starting tasks in parallel when I’m confident this will work without conflicts,” Knopper says. “Systemd has made a Linux system very complex with dependencies at places where you would not expect them logically. You may have noticed that some Linux installations wait for a minute or two before successfully shutting down, just because there are leftover dependencies between network and services after a program removal, so the system has to wait until a timeout is reached when programs or libraries don’t send the ex-

pected answers. This won’t happen on Knoppix because processes are just shut down cleanly without waiting for system services to signal that they are ready for this now.”

Knopper adds, “Actually, for justice, I also dropped `SysVinit` and replaced it by the simpler `BusyBox` internal `init` and shell scripts which replace the `shutdown`, `reboot`, and `poweroff` commands by safe procedures. Removing `systemd` as well as (most of) `SysVinit` also removed a lot of dependencies which would otherwise have pulled in many libraries and services that are not actually needed to run Knoppix. To fulfill the dependency requirements in some Debian packages without messing with the package format, I added these as virtual dependencies, so you will find a package `no-systemd` installed on Knoppix which also avoids accidentally reinstalling `systemd` and possibly killing the boot system.”

## ADRIANE

One indication of how personal a project Knoppix remains is the inclusion of `ADRIANE`, a desktop designed for the blind (Figure 4). `ADRIANE` was developed with Knopper’s sight-impaired wife Adriane in mind, “so she can use a Linux system which she can control completely by herself without needing any proprietary software or surprises by incompatible updates or pop-up license requests, for using the Internet. The way which the audible desktop menu works and the easy-to-remember speech functions [are] her design. [It] tries to reflect the way a blind person starts learning to use a computer on their own with no or few sighted help.”



**Figure 4:** Knoppix’s ADRIANE is a desktop environment for the blind, working through sound and keyboard shortcuts.

Knopper further explains, “ADRIANE is not another add-on for graphical desktops but works with what’s commonly perceivable without vision: speech and text. For beginners who don’t know the whole lot of keyboard shortcuts which graphical screen readers need, [there’s] the simple talking menu which is controlled entirely by arrow keys, Escape, and Caps Lock as ‘Talking control’ keys. When configured with autostarting ADRIANE, the first thing you hear when starting the computer is ‘Enter for help, arrow down for next menu’, which tells you what you can do and how to get more information about the audible desktop usage. The JavaScript-capable text browser ELinks allows you to access and work with most websites easily, and email using the Mutt mail client (which I also use personally because it can handle extreme numbers of inbox mail smoothly) is also possible with only audio and (optionally) Braille devices.”

Knopper adds, “ADRIANE (started with boot option `adr iane`) is not an assistive technology per se; rather it is an alternative, non-vision-oriented desktop that uses assistive technologies like the SBL screenreader (with configurable profiles for different programs for speech and Braille), entirely made of Bash shell scripts with dialogs and accessible console-based programs. Knoppix can also use Orca on the graphical desktop, like many other distros (boot option `knoppix orca`), but that’s not a direct part of ADRIANE.”

Currently, with the help of ADRIANE, Knopper says you should be able to teach yourself to use a Linux system without vision. You can do things like access the Internet and email, use utilities like a calculator, take text notes, use an address book, and scan printed letters or books and then have the computer read them to you or save them as text

files – normal work in an easy-to-use interface. The KARL keyboard learning program teaches you keyboard layout and functions by having the computer read the meaning of a pressed key to you, enabling people unfamiliar with computers to learn how to use a keyboard before using applications. Experts would most likely use the shell with the screen reader, which is also a menu item in ADRIANE.

## Looking Ahead

Asked about future plans for Knoppix, Knopper’s first reply is “whatever is new and interesting in the FOSS world.” For example, with the increased use of videoconferencing, the forthcoming Knoppix 9.2 includes Open Broadcaster Software (OBS) Studio, with virtual camera support.

In addition, Knoppix tends to use 32-bit user space by preference, although it does support a 64-bit kernel. This tendency helps Knoppix to support as much hardware as possible. Some modern applications like Docker are installed as statically linked binaries, but they do not interfere with the rest of the 32-bit system. “I do get a lot of emails asking for a specific 64-bit version,” Knopper says, “but unfortunately I do not have enough free time to work on many versions in parallel.”

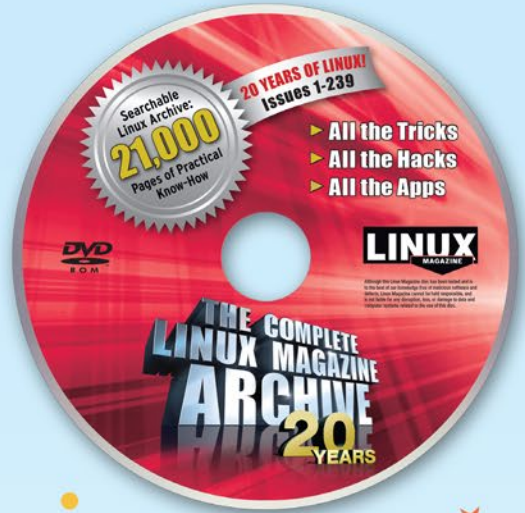
Knoppix will never rival Fedora or Ubuntu, but, then again, it is not meant to. Instead, it has found its own niches and learned to thrive in them. It remains an example of how much one developer can do in free software while borrowing the efforts of others in the community. ■■■

## Info

[1] Download Knoppix:  
<https://www.knopper.net/knoppix-mirrors/index-en.html>



# 20 YEARS LINUX MAGAZINE



## LINUX MAGAZINE 20 YEAR ARCHIVE DVD

ORDER NOW!

<https://bit.ly/Archive-DVD>





Improving Linux package management

# Delivery Service

Linux package managers work too slowly. The experimental distri research project investigates ways to speed up package management. *By Ferdinand Thommes*

**P**ackage managers differ from each other not only in terms of the package formats they use, but also in their execution speed. Developer Michael Stapelberg has been working on how to streamline package managers such as Debian's Apt or Fedora's DNF to make them faster. He has written blog posts on the subject, given talks, and created an experimental distribution, distri [1] to explore the problem.

Distri is a minimal, command-line distribution for reviewing package management concepts in Linux. This is purely a feasibility study and is not suitable for production use. Distri seeks to be the simplest distribution that is still useful.

## Criticism of Debian

Stapelberg, currently a Google developer, was a package maintainer at Debian from 2012 to 2019. Besides maintaining packages of Debian he wrote the i3 Window Manager and the Debian Code Search engine.

In March 2019, he announced in frustration his withdrawal from Debian development with a harsh criticism of the

Debian project [2]. He referred to the practices and tools used to develop, manage, and support the software in the distribution saying that they were often more of a hindrance than a help.

In Stapelberg's opinion, there is a lack of effective tools to implement comprehensive changes in a timely manner. Also, according to Stapelberg, the specifications laid down in the Debian guidelines and pushed by the quality assurance tool Lintian [3] unduly hinder the implementation of necessary technical changes.

## Too Much, Too Slow

In particular, Stapelberg vehemently criticizes the package management – not only in Debian, but Linux in general. Above all, he dislikes that package managers do too much and do it too slowly. To this end, he first conducted a series of tests with small and larger packages using the Apt, DNF, pacman, Nix, and apk package managers, contrasting the metadata downloaded and the time and bandwidth used [4].

What he discovered was that the metadata downloaded was often out of proportion to the package's size, and even more so the smaller the package. Any Debian user can easily check this by looking in the terminal at the end of an apt update to see how many megabytes of metadata the operation downloads. This amount often exceeds the total size of the actual packages to be updated (Figure 1).

The maintainer scripts that the package manager runs during installation also slow down the process, as well as prevent parallel installation of packages. In Stapelberg's view, these scripts can just as easily run the first time the application is launched. Debian processes maintainer scripts using files such as preinst and postinst when installing or updating packages; they contain distribution-specific customizations [5]. Debian has 8,620 maintainer scripts. The Debian Policy Manual web page provides flowcharts that exemplify the underlying complexity here [6]. In Fedora, scriptlets perform this task [7].

Photo by Mika Baumeister on Unsplash

```

ft@blue:~$ sudo apt update
[sudo] Password for ft:
Fetch:1 http://deb.anydesk.com all InRelease [5.588 B]
OK:2 https://deb.opera.com/opera-stable stable InRelease
OK:3 http://dl.google.com/linux/chrome/deb stable InRelease
Fetch:4 http://ftp.de.debian.org/debian unstable InRelease [153 kB]
OK:5 https://wire-app.wire.com/linux/debian stable InRelease
OK:6 http://ftp.uni-stuttgart.de/siduction/extra unstable InRelease
OK:7 https://apt.enpass.io stable InRelease
Ign:8 http://repo.vivaldi.com/snapshot/deb stable InRelease
OK:9 http://packages.siduction.org/extra experimental InRelease
OK:10 http://repo.vivaldi.com/snapshot/deb stable Release
Fetch:11 https://packages.rtot.im/debian sid InRelease [2.910 B]
OK:12 https://packages.siduction.org/fixes unstable InRelease
Fetch:13 http://incoming.debian.org/debian-builddd builddd-unstable InRelease [39,7 kB]
Fetch:15 https://packages.rtot.im/debian sid/main amd64 Packages [1.148 B]
Fetch:16 http://ftp.de.debian.org/debian unstable/main amd64 Packages.diff/Index [63,6 kB]
Fetch:17 http://ftp.de.debian.org/debian unstable/main Translation-en.diff/Index [63,6 kB]
Fetch:18 http://ftp.de.debian.org/debian unstable/main all Contents (deb).diff/Index [63,8 kB]
Fetch:19 http://ftp.de.debian.org/debian unstable/main amd64 Contents (deb).diff/Index [63,8 kB]
Fetch:20 http://ftp.de.debian.org/debian unstable/main amd64 Packages T-2021-01-13-0800.20-F-2021-01-13-0800.20.pdiff [6.351 B]
Fetch:21 http://ftp.de.debian.org/debian unstable/main Translation-en T-2021-01-13-0800.20-F-2021-01-13-0800.20.pdiff [477 B]
Fetch:22 http://ftp.de.debian.org/debian unstable/main all Contents (deb) T-2021-01-13-0800.20-F-2021-01-13-0800.20.pdiff [818 B]
Fetch:20 http://ftp.de.debian.org/debian unstable/main amd64 Packages T-2021-01-13-0800.20-F-2021-01-13-0800.20.pdiff [6.351 B]
Fetch:22 http://ftp.de.debian.org/debian unstable/main all Contents (deb) T-2021-01-13-0800.20-F-2021-01-13-0800.20.pdiff [818 B]
Fetch:21 http://ftp.de.debian.org/debian unstable/main Translation-en T-2021-01-13-0800.20-F-2021-01-13-0800.20.pdiff [477 B]
Fetch:23 http://ftp.de.debian.org/debian unstable/main amd64 Contents (deb) T-2021-01-13-0800.20-F-2021-01-13-0800.20.pdiff [2.236 B]
Fetch:23 http://ftp.de.debian.org/debian unstable/main amd64 Contents (deb) T-2021-01-13-0800.20-F-2021-01-13-0800.20.pdiff [2.236 B]
Fetch:24 http://ftp.de.debian.org/debian unstable/main amd64 DEP-11 Metadata [3.398 kB]
Fetch:25 http://incoming.debian.org/debian-builddd builddd-unstable/main amd64 Packages [248 kB]
Fetch:26 http://ftp.de.debian.org/debian unstable/main DEP-11 48x48 Icons [3.548 kB]
Fetch:27 http://ftp.de.debian.org/debian unstable/main DEP-11 64x64 Icons [7.266 kB]
Fetch:28 http://ftp.de.debian.org/debian unstable/main DEP-11 128x128 Icons [11,6 MB]
Fetch:29 http://ftp.de.debian.org/debian unstable/contrib amd64 DEP-11 Metadata [14,7 kB]
Fetch:30 http://incoming.debian.org/debian-builddd builddd-unstable/main Translation-en [134 kB]
Fetch:31 http://incoming.debian.org/debian-builddd builddd-unstable/contrib amd64 Packages [3.504 B]
Fetch:32 http://incoming.debian.org/debian-builddd builddd-unstable/contrib Translation-en [2.744 B]
26,7 MB were fetched in 12 s (2.175 kB / s).
Package lists are being read. ... Done
Dependency tree is built.
Status information is read in. ... Done
Update available for 20 packages. Run >> apt list --upgradable << to see them.

```

**Figure 1:** When installing and updating packages, the package manager also downloads metadata. The volume of metadata is particularly high in Fedora and Debian, which delays package installation.

## Inefficient

At Google, Stapelberg learned a great deal about effectively updating large amounts of data. Updates to distributions proved fairly ineffective in his research. He did a test using common systems to install one small package and one large package, recording times.

Although the test computer's network connection supported speeds of around 115MBps, none of the distributions achieved more than just under 11MBps of throughput, with most achieving around 3MBps. Alpine Linux performed best, being the fastest in both tests at 10.8MBps and also requiring the lowest volume of metadata to complete the task. Arch Linux and NixOS were in the middle, while Debian and Fedora performed worst.

As an example, for the small 75KB *ack* package, Fedora had to download a massive 114MB and the install took 33 seconds. Alpine, on the other hand, was content with 10MB installed in one second. When installing the virtualization software Qemu, Alpine managed with

26MB, while Fedora needed almost 10 times the amount of data at 226MB.

In view of such numbers, it is little wonder that Fedora is pretty sluggish when it comes to updates and installations. In both scenarios, Debian came in second to last, because downloading the large volume of metadata also delayed the process. Besides Alpine, Arch Linux also had one of the faster package managers in the test.

## Hooks and Triggers

Fedora is also slow because its unpacked package list alone is 60MB, while Alpine's list is a lean 734KB. Fedora offers over 20,000 packages, which is three times more packages than the very small Alpine, but the difference is still striking.

However, Stapelberg thinks even the best results in the test are too slow. He sees another reason for this in the frequently used hooks and triggers that the package manager executes during installation, which trigger the aforementioned maintainer scripts, create daemon user

accounts (such as an FTP or WWW account), or create cache files.

One of the most commonly used triggers, the *man* package trigger ensures that a man page for the package is included on the system with each package installation. In his blog, Stapelberg explains why all this should not happen during installation and how it prevents parallel installation of packages [8].

In Stapelberg's opinion, these interruptions of the actual installation should preferably take place when the app is first launched. If an application does not start between installation and the first or even further updates, the adjustments would only be executed once instead of several times, for example.

## Image Instead of Archive

In Stapelberg's opinion, a package manager should only do what is absolutely necessary to anchor a package in the system so that it is ready for use (i.e., start the program or load a kernel module). Unpacking during installation is not necessary if packages are available as filesystem images that the distribution mounts at startup, as is the case with Appliance or Snap.

According to Stapelberg, no package manager in a Linux distribution currently uses this scenario, although it could still increase the speed to above the level achieved by Alpine's apk, the fastest package manager in his test series. Images are currently only used by the Haiku operating system project.

In Stapelberg's experimental distribution distri, he seeks to experiment with reducing the complexity of package management. He concludes that distributions like Fedora or Debian could also run faster given less complexity. That doesn't mean it's technically easy to implement, but it would be feasible.

For example, distri uses read-only SquashFS images as the package format instead of the usual TAR archives (Figure 2). In addition to increased speed, this has the advantage that applications cannot be modified, which protects them from accidental or malicious modifications.

Distri organizes all files provided by a package under the */ro/* mount point, each in its own directory. The usual data exchange between software packages, which takes place via the speci-

## Index of /distri/supersilverhaze/pkg/

...			
<a href="#">accountsservice-amd64-0.6.55-12.meta.textproto</a>	14-May-2020	06:58	985
<a href="#">accountsservice-amd64-0.6.55-12.squashfs</a>	11-May-2020	09:22	905216
<a href="#">accountsservice-amd64-0.6.55-12.squashfs.gz</a>	11-May-2020	09:22	210022
<a href="#">accountsservice-amd64-0.6.55-12.squashfs.zst</a>	11-May-2020	09:22	174144
<a href="#">accountsservice-amd64.meta.textproto</a>	14-May-2020	06:58	985
<a href="#">ack-amd64-3.3.1-7.meta.textproto</a>	14-May-2020	06:58	149
<a href="#">ack-amd64-3.3.1-7.squashfs</a>	11-May-2020	08:33	258048
<a href="#">ack-amd64-3.3.1-7.squashfs.gz</a>	11-May-2020	08:33	79935
<a href="#">ack-amd64-3.3.1-7.squashfs.zst</a>	11-May-2020	08:33	66383
<a href="#">ack-amd64.meta.textproto</a>	14-May-2020	06:58	149
<a href="#">alsa-lib-amd64-1.1.8-7.meta.textproto</a>	14-May-2020	06:58	76
<a href="#">alsa-lib-amd64-1.1.8-7.squashfs</a>	11-May-2020	08:28	2228224
<a href="#">alsa-lib-amd64-1.1.8-7.squashfs.gz</a>	11-May-2020	08:28	617355
<a href="#">alsa-lib-amd64-1.1.8-7.squashfs.zst</a>	11-May-2020	08:28	555525
<a href="#">alsa-lib-amd64.meta.textproto</a>	14-May-2020	06:58	76
<a href="#">asciidoc-amd64-8.6.9-4.meta.textproto</a>	14-May-2020	06:58	224
<a href="#">asciidoc-amd64-8.6.9-4.squashfs</a>	11-May-2020	09:14	950272
<a href="#">asciidoc-amd64-8.6.9-4.squashfs.gz</a>	11-May-2020	09:14	260424
<a href="#">asciidoc-amd64-8.6.9-4.squashfs.zst</a>	11-May-2020	09:14	184096
<a href="#">asciidoc-amd64.meta.textproto</a>	14-May-2020	06:58	224

**Figure 2:** Distri uses the SquashFS package format, available as images and in packaged formats.

```
ft@blue:~/Images$ VBoxManage convertdd distri-disk.img distri.vdi
Converting from raw image file="distri-disk.img" to file="distri.vdi"...
Creating dynamic image with size 8589934592 bytes (8192MB)...
ft@blue:~/Images$
ft@blue:~/Images$ ls
distri-disk.img  distri-disk.img.zst  distri.vdi
```

**Figure 3:** If using VirtualBox, you first need to convert the downloaded image into a VDI.

fied directories of the Filesystem Hierarchy Standard (FHS) in conventional distributions, is handled by the system via exchange directories, which are provided by FUSE.

For example, the exchange directory `/ro/share/` provides the union of the `share/` subdirectory of all packages in the package store. The global exchange directories map the FHS with sufficient accuracy to allow third-party software, such as Google Chrome or Spotify, to work. Using `/ro/` also prevents conflicts when installing multiple versions of a package.

Distri also streamlines package building. Unlike conventional distributions' builders, the distri package builder does not install packages in the build environment. Instead, the system provides a filtered view of the package store in `/ro/` in the build environment. Even with large dependency trees, setting up a build environment this way takes a fraction of a second.

Distri's website provides information about the various ways to use the distribution [9]. There is no installer yet, but the maintainer has future plans for one. Distri can be started from a USB stick or in a Docker or LXD container, as well as in a virtual machine with VirtualBox or

Qemu. Since it is in IMG format [10] and not an ISO, you first need to convert it to a Virtual Disk Image (VDI) for VirtualBox (Figure 3).

First, you unpack the image. Since the developers have packaged it with the relatively new Zstandard (Zstd) compression algorithm, you will probably need to install `zstd` up front. On Debian, you can do this with:

```
apt install zstd
```

```
distri0 login: root
Password: 2021/01/18 09:43:32 mounting iptables-amd64-1.8.4-7
2021/01/18 09:43:32 mounting runc-amd64-1.0.0-rc9-8

2021/01/18 09:43:40 mounting emacs-amd64-26.3-15
2021/01/18 09:43:40 mounting at-spi2-core-amd64-2.30.1-8
Last login: Sun Jan 17 10:12:12 on tty1
2021/01/18 09:43:40 mounting zsh-amd64-5.6.2-9
```

**Figure 4:** At login, distri mounts a basic set of essential applications. At runtime, it brings in more apps, depending on the usage.

```
distri0# cd /
distri0# pwd
/
distri0# ls
bin  esp  include  lib64      proc  ro-tmp  root  sbin  srv  usr
boot etc  init     lost+found ro      rodebug rosrc  share sys  var
dev  home  lib      opt        ro-dbg roimg   run   src   tmp
```

**Figure 5:** Besides the usual suspects like `/etc` or `/usr`, you will also find distri-specific directories like `/ro` and `/roimg` in the root directory.

On Fedora, use:

```
dnf install zstd
```

Then extract the file using the call:

```
$ unzstd ~/Downloads/distri-disk.img.zst
```

In the process, the compressed file loses the `.zst` attachment and grows to 8GB. An attempt to start distri on an 8GB USB stick failed as expected; you need a stick with at least 16GB capacity. After starting distri, a login prompt opens where you can enter the password, which is *peace* for *root* (Figure 4).

You are greeted by a very simple Z shell prompt that lets you explore the system. At first, I thought the `cd` command had failed, because the prompt does not show the new location after the change – but `pwd` will help here. Entering `cd /ro/` takes you to the directory with all installed packages; switching to `/ro/share/` takes you to the exchange directory (Figure 5).

For common distri commands, and their equivalents in Debian, see distri's documentation [11]. This is also where you can learn more about Distri's package format and how to create your own packages. The `distri update` command (Figure 6) replaces

```
apt update && apt full-upgrade
```

and upgrades the entire distribution at an impressive pace. With a 1Gbps connection, the system downloads and in-

stalls around 275MB of data in less than four seconds.

Package installations are also completed in the blink of an eye (Figure 7). In the case of the Nano editor

```
distri install nano-amd64-4.9.5-2
```

it took just over a millisecond. You need to specify the package version because multiple versions can be installed in parallel. Available packages can be found in the distri repository [12].

## Conclusions

Because distri is an experiment in package management, it is likely to interest only a limited user base. If you want to dig deeper, you should read all of Stapelberg's blog posts on the topic [13] and watch his keynote at the Arch Developer Conference 2020 [14].

While there is no official support for distri, Stapelberg will answer questions on the

mailing list [15] and in the #distri chat room on the legacy.irc-robustirc IRC server. However, you may have to be patient. ■■■

```
distri [word ausgeführt] - Oracle VM VirtualBox
2021/01/17 10:18:49 resolved libxrender to [libxrender-amd64-0.9.10-3 glibc-amd64-2.27-3 libx11-amd64-1.6.6-3 libxau-amd64-1.0.8-3 xorgproto-amd64-2018.4-3 libxcb-amd64-1.13-3 libpthread-stubs-amd64-0.4-3]
2021/01/17 10:18:49 resolved libxpn to [libxpn-amd64-3.5.12-3 glibc-amd64-2.27-3 libx11-amd64-1.6.6-3 libxau-amd64-1.0.8-3 xorgproto-amd64-2018.4-3 libxcb-amd64-1.13-3 libpthread-stubs-amd64-0.4-3]
2021/01/17 10:18:49 resolved npc to [npc-amd64-1.1.0-3 glibc-amd64-2.27-3 gmp-amd64-6.1.2-3 mpfr-amd64-4.0.1-3]
2021/01/17 10:18:49 resolved xzutils to [xzutils-amd64-5.2.4-3 glibc-amd64-2.27-3]
2021/01/17 10:18:49 resolved zlib to [zlib-amd64-1.2.11-3 glibc-amd64-2.27-3]
2021/01/17 10:18:49 resolved xcb-util-renderutil to [xcb-util-renderutil-amd64-0.3.9-3 glibc-amd64-2.27-3 libxau-amd64-1.0.8-3 xorgproto-amd64-2018.4-3 libxcb-amd64-1.13-3 libpthread-stubs-amd64-0.4-3]
2021/01/17 10:18:49 resolved shadou to [shadou-amd64-4.6-5 attr-amd64-2.4.48-3 glibc-amd64-2.27-3 patch-amd64-1.3.1-10]
2021/01/17 10:18:49 resolved xcb-util-image to [xcb-util-image-amd64-0.4.0-3 glibc-amd64-2.27-3 libxau-amd64-1.0.8-3 xorgproto-amd64-2018.4-3 libxcb-amd64-1.13-3 libpthread-stubs-amd64-0.4-3]
2021/01/17 10:18:49 resolved xf86-input-synaptics to [xf86-input-synaptics-amd64-1.9.1-4 glibc-amd64-2.27-3 libudev-amd64-1.5.6-3 libx11-amd64-1.6.6-3 libxau-amd64-1.0.8-3 xorgproto-amd64-2018.4-3 libxcb-amd64-1.13-3 libpthread-stubs-amd64-0.4-3 libxext-amd64-1.3.3-3 libxi-amd64-1.7.9-2 libxfixes-amd64-5.0.3-3]
2021/01/17 10:18:49 resolved pixman to [pixman-amd64-0.34.0-3 glibc-amd64-2.27-3]
2021/01/17 10:18:49 resolved npth to [npth-amd64-1.6-3 glibc-amd64-2.27-3]
2021/01/17 10:18:49 resolved xorg-server to [xorg-server-amd64-1.20.3-4 expat-amd64-2.2.6-3 glibc-amd64-2.27-3 freetype-amd64-2.9.1-3 zlib-amd64-1.2.11-3 libdrm-amd64-2.4.94-3 libpciaccess-amd64-0.14-3 libepoxy-amd64-1.5.2-3 mesa-amd64-18.2.0-3 gcc-libs-amd64-8.2.0-3 gmp-amd64-6.1.2-3 npc-amd64-1.1.0-3 mpfr-amd64-4.0.1-3 libx11-amd64-1.6.6-3 libxau-amd64-1.0.8-3 xorgproto-amd64-2018.4-3 libxcb-amd64-1.13-3 libpthread-stubs-amd64-0.4-3 libxdamage-amd64-1.1.4-3 libxfixes-amd64-5.0.3-3 libxext-amd64-1.3.3-3 libxshmfence-amd64-1.3-3 libfontenc-amd64-1.1.3-3 libudev-amd64-239-10 cryptsetup-amd64-2.0.4-6 json-c-amd64-0.13.1-3 libgcrypt-amd64-1.8.3-3 libgpg-error-amd64-1.32-3 knod-amd64-25-5 libcap-amd64-2.25-3 lun2-amd64-2.03.00-6 libaio-amd64-0.3.111-3 pan-amd64-1.3.1-10 util-linux-amd64-2.32-6 popt-amd64-1.16-3 libfont2-amd64-2.0.3-3 nettle-amd64-3.4-3 pixman-amd64-0.34.0-3 xkbcomp-amd64-1.1.2-3 libkbbfile-amd64-1.0.9-3 xkeyboard-config-amd64-2.25-3]
2021/01/17 10:18:49 resolved xkeyboard-config to [xkeyboard-config-amd64-2.25-3]
2021/01/17 10:18:52 scanPackages in 40.278487ms
2021/01/17 10:18:52 done, 75.81 MB/s (260657361 bytes in 3.279034212s)
```

Figure 6: The distri update command updates the system. In a test, this took just under 3.3 seconds.

```
distri0# distri install cmake-amd64-3.12.4-8
2021/01/18 10:07:33 connecting to /ro-tmp/distri-fuse058504879/distri-fuse-ctl
2021/01/18 10:07:33 resolved cmake-amd64-3.12.4-8 to [cmake-amd64-3.12.4-8 gcc-libs-amd64-9.3.0-4 glibc-amd64-2.31-4 gmp-amd64-6.2.0-4 npc-amd64-1.1.0-4 mpfr-amd64-4.0.2-4 ncurses-amd64-6.2-9]
2021/01/18 10:07:33 installing package "cmake-amd64-3.12.4-8" to root / from repo https://repo.distr1.org/distri/supersilverhaze
2021/01/18 10:07:34 scanPackages in 2.089316ms
2021/01/18 10:07:34 done, 50.38 MB/s (76509431 bytes in 1.448248529s)
```

Figure 7: The installation of individual packages is blazingly fast. The cmake package, weighing in at about 75MB, was installed in about 1.5 seconds on a fast Internet connection.

## Info

- [1] distri: <https://distr1.org>
- [2] Stapelberg's critique of Debian: <https://michael.stapelberg.ch/posts/2019-03-10-debian-winding-down/>
- [3] Lintian: <https://en.wikipedia.org/wiki/Lintian>
- [4] Package manager comparison: <https://michael.stapelberg.ch/posts/2019-08-17-linux-package-managers-are-slow/>
- [5] Maintainer scripts: <https://www.debian.org/doc/debian-policy/ch-maintainerscripts.html>
- [6] Debian flowcharts: <https://www.debian.org/doc/debian-policy/ap-flowcharts.html>
- [7] Fedora scriptlets: <https://docs.fedoraproject.org/en-US/packaging-guidelines/Scriptlets/>
- [8] Hooks and Triggers: <https://michael.stapelberg.ch/posts/2019-07-20-hooks-and-triggers/>
- [9] Using distri: <https://distr1.org/getting-started/>
- [10] distri download: <https://repo.distr1.org/distri/supersilverhaze/img/>
- [11] distri documentation: <https://repo.distr1.org/distri/jackherer/docs/rosetta-stone.html>
- [12] distri repository: <https://repo.distr1.org/distri/supersilverhaze/pkg/>
- [13] Michael Stapelberg's blog: <https://michael.stapelberg.ch/posts/tags/distri/>
- [14] Stapelberg's keynote at Arch Developer Conference 2020: <https://media.ccc.de/v/arch-conf-online-2020-6387-distri-researching-fast-linux-package-management>
- [15] distri mailing list: <https://www.freelists.org/list/distri>



## Competing software installers

# Proliferation

With an increasing number of software installation methods, testing cutting-edge applications may require learning about the installer first. *By Bruce Byfield*

A well-known *xkcd* cartoon starts with 14 competing standards [1]. Someone resolves to simplify the situation by developing one standard that covers all the others. The result? Fifteen competing standards. The cartoon reflects a wry truth that has recently been illustrated in a proliferation of software installers that complicates matters for those who install cutting-edge applications. Increasingly, would-be users have to learn a new installer and install more software before they can get to the software they want to try.

When Linux first became popular in the late 1990s, software installation often ended in what was known as “dependency hell,” in which users had to track down the correct version of each

dependency and risked leaving a package half-installed. One way around this problem was to use static tarballs – archives that included the required dependencies. These were used in the first forays into commercial Linux software, like Loki Entertainment, and they were easy to create with the `tar` command. However, static tarballs were not always used, probably because they often meant that one hard drive might contain several versions of the same software, which was wasteful at a time when storage was limited.

Another early solution was to install software written in C or C++ from source, creating the necessary binaries. If you were lucky, the install scripts

would even list and install dependencies, although less conscientious developers might still strand users in dependency hell. Installing from source typically begins by switching into the directory that contains the source code and running `./configure` to check that all the required dependencies are already installed (Figure 1). For instance, one of the first checks that `./configure` usually does is to check whether the system has the GCC compiler installed and running correctly. In addition, `./compile` creates the files needed to compile, such as the makefile that contains the basic instructions for compiling. If all dependencies are present, you can then compile the binary using the `make` command. If the

### 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/>.

```
checking for curl-config... /usr/bin/curl-config
checking curl/curl.h usability... yes
checking curl/curl.h presence... yes
checking for curl/curl.h... yes
checking for libcurl header... no
checking whether pthreads work with -pthread... yes
checking searching for pthread library... using -lpthread
checking for joinable pthread attribute... PTHREAD_CREATE_JOINABLE
checking if more special flags are required for pthreads... no
checking for the Boost Version...
checking for boost header... checking for Boost libraries... -lboost_thread
-mt -lboost_program_options-mt
checking if -fvisibility-inlines-hidden is broken... no
checking whether g++ supports -fvisibility=hidden... yes
```

**Figure 1:** The `./configure` command is used to compile C and C++ source code, along with `make` and `make install`.

build is successful, the command `make install` installs the application. This system at least provides some minimal guidance in dealing with problems and is still used today, as well as variants like KDE's CMake.

Installation took a giant step forward when Debian became the first distribution to introduce package management. With package management, applications are presented along with an array of scripts that handle dependencies (as well as any alterations to the system before, during, and after installation) and draw from a standard set of online repositories. The efficiency, as well as the economy of memory, meant that package management soon spread to other major distributions. The commands may differ – `apt-get` or `dpkg` for Debian, `yum` or `dnf` for Fedora, and `pacman` for Arch Linux – but the structure of package managers is much the same in all distributions, with basic commands for installation, deletion, and repository searching and updating, as well as a centralized database for all installations. Several years ago, Ubuntu introduced `apt`, a convenient subset of `apt-get` functions (Figure 2), but the structure of package management has remained the same for over two decades. Most Linux users have likely

used a package manager, if only through a graphical interface.

## Universal Packages

Today, Linux computing is vastly different than it was in its infancy. Memory is abundant now, and the restrictions of earlier times no longer seem relevant to many. For instance, on many systems, package installation no longer needs to be part of system administration in order to manage limited system resources. Instead, ordinary users can install software for their own use. Similarly, there is room for multiple versions of resources, something that package management generally avoided. And, most importantly, many developers would prefer to build a single file for all distributions, in effect making software installation on Linux closer to what it is on Windows or macOS.

Currently, there are three main universal package systems: AppImage, Canonical's Snap, and Fedora's Flatpak. Announced in 2016, Snap and Flatpak helped to renew the interest in AppImage (Figure 3). All three are structured much like traditional package management systems and might be said to be a more sophisticated version of static tarballs. However, although all three are highly promising (e.g., Flatpak's home

page declares the format “The Future of Apps on Linux”), distributions differ enough in details like the location of files that one format that fits all is not always easy to create. Several distributions like openSUSE now maintain releases in a universal format, but all three universal package systems have become just more standards among many.

## The New Installers

Another growing trend is to use the installer included in a programming language. For instance, `Carp` is the package manager for the Rust programming language. Similarly, Ruby borrows `Homebrew` from Apple.

The most common of these programming language installers is `pip`, the Python installer (Figure 4). Until January 2020, a `pip` version existed for Python releases before 3.x and is undoubtedly still around. For current versions of Python, `pip3` is the command. However, somewhat confusingly, at the command prompt, `pip3` is referred to merely as `pip`. The command structure is further complicated by calling `pip` from Python3:

```
python3 -m pip install PACKAGE
```

The package can be defined by a local path, or the version control or Python

```
root@nanday:~# apt install tweak
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  tweak
0 upgraded, 1 newly installed, 0 to remove and 192 not upgraded.
Need to get 48.0 kB of archives.
After this operation, 133 kB of additional disk space will be used.
Get:1 http://ftp.us.debian.org/debian buster/main amd64 tweak amd64 3.02-4 [48.0 kB]
Fetched 48.0 kB in 1s (61.8 kB/s)
Retrieving bug reports... Done
Parsing Found/Fixed information... Done
Selecting previously unselected package tweak.
(Reading database ... 588357 files and directories currently installed.)
Preparing to unpack ../tweak_3.02-4_amd64.deb ...
Unpacking tweak (3.02-4) ...
Setting up tweak (3.02-4) ...
Processing triggers for man-db (2.8.5-2) ...
Processing triggers for menu (2.1.47+b1) ...
```

**Figure 2:** Typical of modern package manager commands, `apt` explains the results of requests and reports on progress.

```
bb@nanday:~/Applications$ chmod +x ^C
bb@nanday:~/Applications$ chmod +x Chrysalis-0.7.9_9c382eed7aaf860d9b28ecbf97084be.AppImage
bb@nanday:~/Applications$ ./Chrysalis-0.7.9_9c382eed7aaf860d9b28ecbf97084be.AppImage
```

**Figure 3:** AppImage requires only a download and a change in permissions to run.

```
bb@nanday:~$ python3 -m pip install "Django"
Defaulting to user installation because normal site-packages is not writeable
Collecting Django
  Using cached Django-3.1.6-py3-none-any.whl (7.8 MB)
Requirement already satisfied: sqlparse>=0.2.2 in ~/.local/lib/python3.7/site-packages (from Django) (0.4.1)
Requirement already satisfied: pytz in ~/.local/lib/python3.7/site-packages (from Django) (2021.1)
Requirement already satisfied: asgiref<4,>=3.2.10 in ~/.local/lib/python3.7/site-packages (from Django) (3.3.1)
Installing collected packages: Django
Successfully installed Django-3.1.6
```

**Figure 4:** Python's pip installer closely resembles standard package installers.

repository. An exact version can be specified by `PACKAGE==VERSION`, while specifying a version and adding `<HIGHER-VERSION` or `>LOWER-VERSION` specifies a range of installation candidates. In addition, `~=VERSION` specifies an installation candidate compatible with a certain version.

Almost alone among the other installation choices, this syntax is more complicated than that of other package managers discussed here. However, it is included, because, as its use of Debian syntax implies, pip seems designed as an improvement over traditional package management. Its syntax makes pip an installer for experts, providing them with advanced tools.

## Cloning from Servers

An increasingly common installation method is to copy the files from a version

control repository. The most common command for this purpose is `git`, using the command `git clone URL` (Figure 5), but `curl` and `wget` are also occasionally used. With all three commands, several scenarios are possible. You might copy only source code and find installation instructions in a README file. If the application is written in an interpreted language like Python, the copied files are ready for use. Alternatively, a developer may have included a compiled binary and all the dependencies. In all these cases, you only need to follow the provided instructions, although you may need to install a tool or two before you install.

## 15 Competing Standards

I have omitted some of the installation methods available for new applications on GitHub or GitLab, which are mostly

the installers for less common programming languages. And I admit that I have rarely seen many of these methods as anything other than a distraction from my browsing of newer applications. However, the choice of installation methods is not made to help users. Much of the time, the choice is made for the convenience of the developers, who would understandably rather be working on the code than making provisions for users before the code's general release. However, the situation remains reminiscent of the *xkcd* cartoon, and those who browse the latest code development need to be aware of how the choice of installers can become an obstacle. ■■■

### Info

[1] "Standards," *xkcd*: <https://xkcd.com/927/>

```
bb@nanday:~/git-test$ git clone https://github.com/keyboardio/Model01-Firmware
Cloning into 'Model01-Firmware'...
remote: Enumerating objects: 538, done.
remote: Total 538 (delta 0), reused 0 (delta 0), pack-reused 538
Receiving objects: 100% (538/538), 173.54 KiB | 1.04 MiB/s, done.
Resolving deltas: 100% (305/305), done.
```

**Figure 5:** Cloning a Git repository can be a quick way to install an application.



# IT Highlights at a Glance

The collage features several overlapping article thumbnails:

- ADMIN HPC:** HPC Up Close, Highlights, Further Reading, and a 'FULL PROGRAM FULLY VIRTUAL SC20' banner.
- ADMIN Update - Hottest Links:** Certificate Transparency, Going Worm Targeting Linux Servers, and a 'Most Read Article' section.
- LINUX UPDATE:** Exploring the World of Linux, Featured Articles, Further Reading, and a '20 Years of Linux Magazine' anniversary graphic.

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)

Secure communication over the unreliable UDP transport with DTLS

# Secret Delivery



TLS encryption is wonderful if it is running over a reliable transport protocol like TCP; but if your needs call for the less reliable UDP transport, you'd better start learning about DTLS. *By Andrei Kuzmenko*

**T**CP/IP is at the heart of the Internet, and the Transport layer is at the heart of TCP/IP. The Transport layer is responsible for end-to-end connections between the sender and receiver over a TCP/IP network. The two most common Transport layer protocols are Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). Nearly all Internet traffic uses either TCP or UDP.

TCP is connection-oriented, which means that a connection between the client and server is established before data can be sent. The TCP protocol provides reliable ordering and error-checked delivery. UDP is a connectionless protocol, which means it provides only minimal information and has no handshaking procedure. UDP does not offer a guarantee of ordering or delivery. Of course, the brevity of UDP makes it much faster than the steady and careful TCP, so applications that don't require a high level of reliability tend to use UDP.

TCP and UDP were created in more innocent days of an Internet, when networks did not face the security

challenges we deal with today. The Transport Layer Security (TLS) protocol (and its predecessor SSL) were developed to provide encryption for communication security at the Transport layer. TLS offers privacy and data integrity between two communicating network nodes; however, it requires a reliable transport protocol, which means it won't work with the simple and unreliable UDP. TLS assumes that the packets arrive in the correct order, which TCP ensures but UDP does not.

Does this absence of ordering control mean that you can't use UDP for encrypted communication? Not exactly. UDP is intended as a foundation, upon which developers can add new protocols and services. The Datagram Transport Layer Security (DTLS) protocol [1] was developed to bring TLS-like privacy and encryption to UDP. The creators of DTLS wanted it to be as much like TLS as possible, adding only the necessary services needed to make TLS-style encryption work correctly.

This article introduces you to DTLS and offers some thoughts on how you

can integrate DTLS into your own programs using OpenSSL or GnuTLS.

## How It Works

DTLS reuses most of the protocol elements from TLS, with minor but crucial modifications for it to work properly with datagram protocols like UDP. To cope with the unreliability of connectionless protocols, DTLS had to implement a solution for packet loss and packet reordering.

DTLS messages are grouped into a series of message flights. Although each flight may consist of a number of messages, the flight should be viewed as monolithic for the purpose of time out and retransmission. Figure 1 shows the DTLS 1.2 handshake procedure.

Because the DTLS handshake takes place over unreliable datagram transport, it is vulnerable to two types of Denial of Service (DoS) attacks. The first scenario is a standard resource-consumption attack. An adversary could start multiple handshake requests and cause the server to allocate system resources to burdensome cryp-

Lead Image © Daniel Villeneuve, 123RF.com

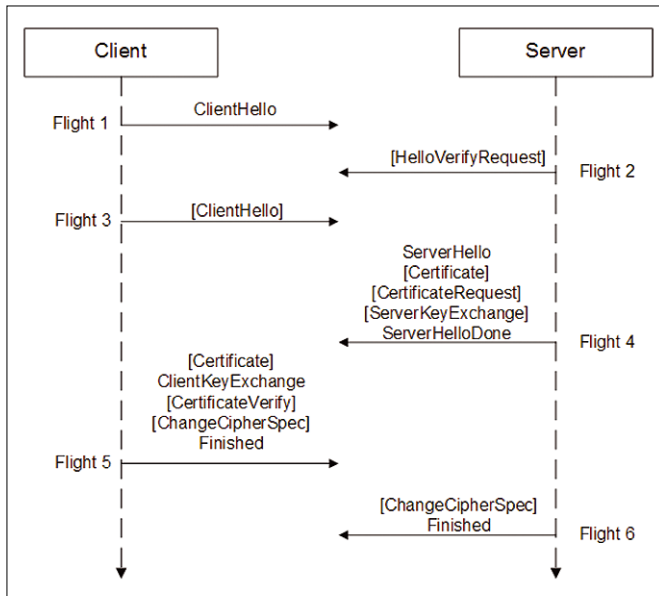


Figure 1: The DTLS 1.2 handshake.

topographic operations. This attack could slow or disrupt other network connections. The second attack is an amplification attack, in which an adversary sends a `ClientHello` message to the server and receives a large `CertificateMessage` response. To cause the damage, an adversary could employ IP spoofing to cause the DTLS server to send large `CertificateMessage` messages to the victim's IP address.

To protect against DoS attacks, DTLS adds a stateless cookie exchange to the handshake. The term *cookie* is universally used in computer science to describe a small packet of information that is sent and received without changes. The term “stateless” means that a cookie should be generated in such a way that it does not require keeping state on the server, which would require memory consumption.

To prevent DoS attacks, the DTLS server responds to the client's `ClientHello` message with a `HelloVerifyRequest` message, which contains a cookie. The client then has to repeat the `ClientHello` message with the cookie attached. The server verifies the cookie and proceeds with the handshake only if the cookie is valid.

The “Denial-of-Service Countermeasures” section of RFC 6347 [2] states the following:

*“DTLS servers SHOULD perform a cookie exchange whenever a new handshake is being performed. If the server is being operated in an environment where amplification is not a problem, the server*

According to RFC 6347, the cookie exchange must be enabled by default on the server side, and a user has no duty

to activate it. Because the cookie exchange is not a mandatory attribute of the handshake procedure, it can theoretically be turned off if the user really wants to remove it and understands all consequences of this decision. Network servers that are more sensitive to overall handshake latency can skip the `HelloVerifyRequest` message and instead respond with a `ServerHello` message, in which case the protocol behavior is the same as in the TLS protocol. Servers that choose to make this optimization can still be used as DoS amplifiers and should therefore not skip `HelloVerifyRequest` in environments where an amplification attack is a possibility.

RFC 6347 gives permission to disable the cookie exchange mechanism during the handshake procedure. The question is how this capability is realized in popular software libraries. The fol-

*MAY be configured not to perform a cookie exchange. The default SHOULD be that the exchange is performed, however.”*

Note the keywords “SHOULD” and “MAY.” The keyword “SHOULD” indicates items that can be omitted given valid reasons. The keyword “MAY” indicates features that can be arbitrarily omitted.

lowing sections look at implementing DTLS in OpenSSL (1.1.1i) and GnuTLS (3.6.15).

## OpenSSL

OpenSSL [3] is a very popular library and the de facto standard open source TLS implementation. The architecture of OpenSSL consists of three parts: the *context* (CTX), the *session* (SSL), and the *basic input/output subsystem* (BIO). The context is responsible for the protocol (SSL/TLS/DTLS), the session cache, and other global parameters. Each new session is represented by the SSL object created from the existing context. The SSL object holds the session state and a BIO object. The BIO object is used to communicate with a network socket. The scheme of the DTLS server is defined as shown in Listing 1.

The function `DTLSv1_listen()` waits for incoming `ClientHello` messages, responds with a `HelloVerifyRequest` message,

### Listing 1: Skeleton of a DTLS Server in OpenSSL

```

01 /* Functions to process cookies */
02 int gen_cookie(...){...};
03 int verify_cookie(...){...};
04 int main(void){
05 /* Library initialization */
06 SSL_load_error_strings();
07 SSL_library_init();
08 /* Create DTLS Context */
09 mtd = DTLSv1_server_method();
10 ctx = SSL_CTX_new(mtd);
11 /* Load SSL certificates */
12 /* Bind callbacks */
13 SSL_CTX_set_cookie_generate_cb(...);
14 SSL_CTX_set_cookie_verify_cb(...);
15 /* Create UDP socket */
16 fd = socket(...);
17 /* Process UDP packets */
18 for(;;){
19 /* Prepare new session */
20 BIO *bio = BIO_new_dgram(...);
21 SSL *ssl = SSL_new(ctx);
22 SSL_set_bio(ssl, bio, bio);
23 /* Waiting for ClientHello msg */
24 while(DTLSv1_listen(...) <= 0);
25 /* Encrypted data FROM client */
26 /* Process data */
27 /* Encrypted data TO client */
28 do_session(ssl, &client);
29 }
30 }
  
```

and returns 0, which means that no client has been verified yet and it must be called again to continue listening. When the client sends a `ClientHello` message with a valid cookie attached, the function will return 1 and the `sock_addr` structure of the verified client. There are two callback functions to provide cookie exchange operations on the server side: `gen_cookie()` and `verify_cookie()`.

When a cookie has to be generated for a `HelloVerifyRequest` message, the `gen_cookie()` function is called. After receiving an answer from the client with a cookie attached to a `ClientHello` message, the `verify_cookie()` function is used. Note that the programmer must implement these functions. I should emphasize that the implementation of `gen_cookie()` and `verify_cookie()` is a mandatory requirement within the existing architecture of the OpenSSL library. If the callbacks are not binded to the CTX by using `SSL_CTX_set_cookie_generate_cb()` and `SSL_CTX_set_cookie_verify_cb()`, or a NULL value is used instead of the function's address as a parameter of the binding function, the DTLS server will not respond to the client after receiving a `ClientHello` message.

By default the cookie exchange is enabled, and you do not have a way to change this part of the handshake procedure. There is an option called `SSL_OP_COOKIE_EXCHANGE` in the public API of the

OpenSSL library that could be used to regulate the cookie exchange mechanism. But this option is used by the OpenSSL library itself in the body of the function `DTLSv1_listen()`. Therefore, this option is useless for the programmer. Within the current implementation of the library, the cookie exchange is always on.

## GnuTLS

GnuTLS [4] is a C-based library that implements protocols ranging from SSL 3.0 to TLS 1.3, accompanied with the required means for authentication and public key infrastructure. The strong side of GnuTLS is a detailed documentation that is available online [5]. The documentation contains a brief introduction to the secure communication protocols and many examples of source code. GnuTLS provides essential means to write DTLS applications. One problem is that the library does not provide a function like `DTLSv1_listen()` from OpenSSL. You should construct the whole DTLS handshake procedure in your own program using functions provided by the library, which means that the text of the source code and the program structure of the application will be different depending on the programmer's decisions. If you want an RFC-compliant application, you must design and write the code as RFC-compliant.

Listing 2 shows the skeleton for a DTLS server in GnuTLS.

The GnuTLS approach is very similar to the OpenSSL scheme. The main difference is that the cookie exchange procedure is not a built-in part of `gnutls_handshake()` on the server side. From the documentation:

*“Because datagram TLS [DTLS] can operate over connections where the client cannot be reliably verified, functionality*

### GnuTLS Without Cookies

If you do not want to use the cookie exchange mechanism, you should remove all parts of the source code where cookies are used. In the files provided with this article [6], you will find a file named `cookie.diff` where you can see all steps to remove the cookie exchange from the source code of the DTLS server. You can obtain a version of the DTLS server without cookie exchange by executing the following command `patch -b server.c cookie.diff` in your terminal program. When you get this version, you should compile it and then collect the network traffic using `tcpdump`. After loading the dump in Wireshark, you will see a screen view similar to one in Figure 2.

As you can see, the traffic dump does not contain a `HelloVerifyRequest` message with attached cookie. The DTLS connection was successfully established without the cookie exchange during the handshake procedure.

### Listing 2: Skeleton of a DTLS Server in GnuTLS

```

01 /* Data structures for cookies */
02 gnutls_dtls_prestate_st prestate;
03 gnutls_datum_t cookie_key;
04 /* Init library */
05 gnutls_global_init();
06 /* Prepare key for cookie */
07 gnutls_key_generate(...);
08 /* Create socket */
09 fd = socket(...);
10 /* Wait for incoming udp packets */
11 for(;;){
12     /* Get udp payload */
13     ret = recvfrom(...);
14     if(ret > 0){
15         /* try to verify cookie */
16         ret = gnutls_dtls_cookie_verify();
17         if(ret < 0){
18             /* Send HelloVerifyRequest */
19             /* with attached cookie */
20             gnutls_dtls_cookie_send(...);
21             continue;
22         }
23     }
24     else continue;
25     /* Prepare session */
26     /* ... */
27     gnutls_dtls_prestate_set(...);
28     /* Do DTLS handshake */
29     gnutls_handshake(...);
30     /* Do DTLS data exchange */
31     for(;;){
32         /* Encrypted data exchange */
33         do_session(...);
34     }
35 }

```

in the form of cookies is available to prevent denial of service attacks to servers... If successful, the session should be initialized and associated with the cookie using `gnutls_dtls_prestate_set` before proceeding to the handshake. Note that the above apply to server side only and are not mandatory. Not using them, however, allows denial of service attacks. The client-side cookie handling is part of `gnutls_handshake`.”

Unlike the OpenSSL library, GnuTLS allows you to forego the cookie exchange without any limitations (see the box entitled “GnuTLS Without Cookies,”) but this option is best reserved for secure networks. If you plan to use DTLS on the

Internet, the cookie exchange is an important protection against denial of service attacks.

## The End

DTLS brings encryption capabilities similar to TLS to the connectionless UDP protocol. Both the OpenSSL and GnuTLS libraries provide a high level of security for DTLS connections. It is not a problem to perform the cookie exchange procedure during the DTLS handshake, but it is difficult to refuse this high level of protection. You can't use the only option to turn off the cookie exchange when configuring the DTLS object. You also can't use any stubs or NULL-pointer arguments.

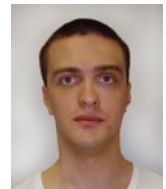
GnuTLS makes you write a part of the server-side handshake procedure manually, which is not a user-friendly approach. OpenSSL already has all the necessary pieces in place, but the current implementation of the library is rather strange. In the future, greater adoption of DTLS will depend upon developers having access to simple and reliable libraries and frameworks with predictable behavior. ■■■

## Info

- [1] DTLS: [https://en.wikipedia.org/wiki/Datagram\\_Transport\\_Layer\\_Security](https://en.wikipedia.org/wiki/Datagram_Transport_Layer_Security)
- [2] RFC 6347: <https://tools.ietf.org/html/rfc6347>
- [3] OpenSSL: <https://www.openssl.org/>
- [4] GnuTLS: <https://www.gnutls.org/>
- [5] GnuTLS documentation: <https://www.gnutls.org/documentation.html>
- [6] Code for this article: <ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/247/>

## Author

Andrei Kuzmenko is a professional software engineer and researcher. He is particularly interested in network technologies and Bash scripting. In the last 15 years, he has been working on Linux and its applications. C++ is his old passion.



No.	Time	Source	Destination	Protocol	Length	Info
1	0.000000	127.0.0.1	127.0.0.1	DTLSv1.2	257	Client Hello
2	0.151255	127.0.0.1	127.0.0.1	DTLSv1.2	160	Server Hello
3	0.151314	127.0.0.1	127.0.0.1	DTLSv1.2	1242	Certificate (Fragment)
4	0.151329	127.0.0.1	127.0.0.1	DTLSv1.2	289	Certificate (Reassembled)
5	0.151346	127.0.0.1	127.0.0.1	DTLSv1.2	652	Server Key Exchange
6	0.151359	127.0.0.1	127.0.0.1	DTLSv1.2	67	Server Hello Done
7	0.503026	127.0.0.1	127.0.0.1	DTLSv1.2	208	Client Key Exchange, Change Cipher S
8	0.828924	127.0.0.1	127.0.0.1	DTLSv1.2	56	Change Cipher Spec
9	0.829018	127.0.0.1	127.0.0.1	DTLSv1.2	103	Encrypted Handshake Message
10	0.880550	127.0.0.1	127.0.0.1	DTLSv1.2	85	Application Data
11	0.880940	127.0.0.1	127.0.0.1	DTLSv1.2	85	Application Data
12	0.881227	127.0.0.1	127.0.0.1	DTLSv1.2	81	Encrypted Alert
13	0.881558	127.0.0.1	127.0.0.1	DTLSv1.2	81	Encrypted Alert

Frame 1: 257 bytes on wire (2056 bits), 257 bytes captured (2056 bits)  
 Ethernet II, Src: 00:00:00\_00:00:00 (00:00:00:00:00:00), Dst: 00:00:00\_00:00:00 (00:00:00:00:00:00)  
 Internet Protocol Version 4, Src: 127.0.0.1, Dst: 127.0.0.1  
 User Datagram Protocol, Src Port: 51400, Dst Port: 8899  
 Datagram Transport Layer Security

Figure 2: DTLS connection without the cookie exchange.



## Working with the JSON data format

# DATA DOG

JSON data format is a standard feature of today's Internet – and a common option for mobile and desktop apps – but many users still regard it as something of a mystery. We'll take a close look at JSON format and some of the free tools you can use for reading and manipulating JSON data.

By Frank Hofmann and Veit Schiele

**O**ur world of web applications and fast, interactive mobile devices calls for the free exchange of data in easily accessible forms. Standard formats promote interoperability and minimize development time. Open formats also make it easy to import data into other applications. Over the years, several popular alternatives have emerged. CSV, XML, and YAML are well known and easy to adapt to different applications (see the box entitled “Comparing Formats” and Listings 1-3 for examples). One format that is used extensively for web applications, mobile applications, and even some conventional desktop tools is JavaScript Object Notation (JSON) [1].

JSON is wildly popular as a tool for passing information between web apps – for instance, it is currently the de facto standard for REST services – yet for many users, the details of JSON format are shrouded in mystery. This article takes a close look at JSON and some of the tools available for reading, manipulating, and importing JSON data.

### Understanding JSON

The notation of JSON is analogous to objects, records, or dictionaries – depending

on what that structure is currently called in your favorite programming language. Even though JSON format is based on JavaScript, parsers exist in almost all programming languages. In addition to Awk and C/C++, you can integrate JSON with Fortran, Go, Lisp, Lua, Python, and Visual Basic.

In everyday life, you will find the format in the data-sharing Jupyter Notebook app [2], in geographical specifications like GeoJSON [3] (Listing 4), and even in databases like MongoDB.

Taking a closer look at the JSON data structure, you will see that it is in an easy-to-read, text-based format. Parentheses, colons, and commas separate the individual elements; the data can be nested as desired. This means, for example, that you can map lists, arrays, or objects. Table 1 summarizes the elementary data types that JSON supports.

Mark the individual structure levels with brackets and indentation for better readability (Listing 5). Pairs of

#### Listing 1: CSV File

```
Stephen Fry; The Hippopotamus; 1994
Ian Rankin; Set In Darkness; 2009
Ken Follett; The Pillars of the Earth; 1989
```

#### Listing 2: XML File

```
<inventory>
  <book>
    <author>Stephen Fry</author>
    <title>The Hippopotamus</title>
    <publication>1994</publication>
  </book>
  <book>
    <author>Ian Rankin</author>
    <title>Set In Darkness</title>
    <publication>2009</publication>
  </book>
  <book>
    <author>Ken Follett</author>
    <title>The Pillars of the Earth</title>
    <publication>1989</publication>
  </book>
</inventory>
```

Lead Image © Fabian Schmidt, Fotolia.com

curly braces { and } each form a unit. Square brackets [ and ] are used to indicate fields (also known as arrays).

Individual field elements follow the form of an enumeration and are separated by commas. Each field element

consists of a key-value pair separated by a colon (:).

JSON was originally created in the early 2000s to exchange data between web applications. JSON worked quite well in the web context (even though you cannot always parse it unambiguously). To structure data, JSON falls back on conventions familiar to anyone who has programmed in a C-based language (C, C++, C#, Java, JavaScript, Perl, Python, and others).

JSON is specified according to RFC 8259 [4] and ECMA-404; common extensions are JSONP (JSON with padding), JSONPP (JSON with padding and parameters), and JSONML, which combines XML and JSON together. The character set for all JSON formats is Unicode (UTF-8), which eliminates the character-set guessing game that you will be familiar with from CSV.

You can use JSON to exchange smaller volumes of data between applications in an agile way. However, if the transferred data volume increases (e.g., if you have millions of measurements from a sensor), JavaScript-based Python libraries like Ipywidgets, Bokeh, and Plotly often fail. In the face of large data volumes, binary

```
dd@ubuntu:~$ jq . books.json
{
  "book": [
    {
      "author": "Stephen Fry",
      "title": "The Hippopotamus",
      "publication": "1994"
    },
    {
      "author": "Ian Rankin",
      "title": "Set In Darkness",
      "publication": "2009"
    },
    {
      "author": "Ken Follett",
      "title": "The Pillars of the Earth",
      "publication": "1989"
    }
  ]
}
```

**Figure 1:** The Jq tool puts in your JSON output and keeps the output readable.

**Table 1:** JSON Data Types

Data Type	Description
Strings	All Unicode characters except ", \, and control characters
Numbers	Numeric values including hexadecimal and exponential values, for example 0x42 and .2e-3
Boolean values	Logic values true and false
Arrays	Comma-separated, unordered lists of properties, although objects without properties are also allowed
Objects with properties	Notation as key-value pairs
Null values	null, NIL, or ()

## Comparing Formats

Using comma-separated values (CSV) ensures the data remains manageable for the most part, but the format is neither standardized nor particularly flexible. In a file like the one in Listing 1, neither the character set nor the separator between columns is fixed. In practical applications, spaces, tabs, hyphens, or semicolons are sometimes used instead of commas. Corresponding key data either has to be agreed upon or inferred from the file itself. Furthermore, the format does not allow nested records, arrays, or binary data.

The Extensible Markup Language (XML) is far more structured and flexible than CSV. A notable feature of XML is the data field enclosed in the field name with the form <fieldname>value</fieldname> (Listing 2). In practice, it makes sense to choose field names that let you infer the contents. The order of the fields is usually variable in a layer, and fields can be missing. One disadvantage of XML is that an XML file is significantly larger due to the need to continually repeat the field labels for each entry.

YAML is a recursive acronym for YAML Ain't Markup Language. The YAML specification describes a very compact way to serialize data. Hyphens and indentations using spaces serve as the basis for denoting fields. YAML borrows from XML, as well as from the format in which three programming languages (Python, Perl, and C) describe their data structures. Listing 3 shows the book inventory data as a YAML structure.

JSON [1] is based on JavaScript. The format is also very compact and flexible. In contrast to YAML, JSON explicitly identifies objects and their attributes, whereas in YAML, the assignment is derived from the context of the indentation depth.

## Listing 3: YAML File

```
---
book:
- author: Stephen Fry
  title: The Hippopotamus
  publication: '1994'
- author: Ian Rankin
  title: Set In Darkness
  publication: '2009'
- author: Ken Follett
  title: The Pillars of the Earth
  publication: '1989'
```

## Listing 4: GeoJSON File

```
{
  "type": "Feature",
  "geometry": {
    "type": "Point",
    "coordinates": [125.6, 10.1]
  },
  "properties": {
    "name": "Dinagat Islands"
  }
}
```

transport mechanisms are a better option for handling the load.

### Tool Overview

Several command-line tools are available for parsing, processing, and outputting JSON data. Table 2 summarizes some of the available tools. All of these tools are available as packages for Debian GNU/Linux, Ubuntu, Devuan, Linux Mint, and macOS.

Not all of the tools in Table 2 are intuitive, and some of them only develop their full impact in a specific context. You will find more information on these tools in the various documents and cheat sheets available online [12].

### Easily Readable JSON Output

When it comes to pretty printing, aeson-pretty, jc, jo, jq, and Jshon all have something to say. Some of the tools have a command-line parameter for printing, for example, -p for jo.

In Listing 6, cat and aeson-pretty work together for readable output via a pipe. Jq delivers the same results with the next call, but the output is in color (Figure 1):

```
$ jq . book_inventory.json
```

The dot in the call to jq is not immediately understandable. It stands for the expression to be processed; in this case, it denotes all objects specified as

#### Listing 5: JSON File

```
{ "book": [
  {
    "author": "Stephen Fry",
    "title": "The Hippopotamus",
    "publication": "1994"
  },
  {
    "author": "Ian Rankin",
    "title": "Set In Darkness",
    "publication": "2009"
  },
  {
    "author": "Ken Follett",
    "title": "The Pillars of the Earth",
    "publication": "1989"
  }
]
}
```

parameters in the JSON file. You can define colorizing of the output using two options, -C (--colour-output) and -M (--monochrome-output).

Some users prefer compact output with as few (space) characters as possible. In Listing 7, see aeson-pretty with the -c (short for --compact) option. This option reduces the number of characters in the output by 45 percent, from 428 to 236 bytes. Compared to Listing 5 and Listing 6, the results still convey the same information, but with only half the amount of data.

### Verifying JSON Data

The formatted output of a tool like jq or aeson-pretty usually helps to detect obvious errors in the JSON structure at a glance, but if you need a closer look, you can check your JSON data with JSONLint. The tool expects the file with the JSON data as a parameter. If everything is correct, it reports back that JSON is valid. If there is an error, it outputs the location that it identified as the error. Figure 2 shows JSONLint output for the book inventory – first with a correct JSON file, and then with

Table 2: JSON Tools

Tool	Language	Application
aeson-pretty [5]	Haskell	Output JSON in a readable way
jc [6]	Python	Convert output to JSON
jid [7]	Go	Interactively filter JSON
jo [8]	C	JSON output in the shell
jq [9]	C	Output and filter JSON in a readable way
Jshon [10]	C	Read and generate JSON
JSONLint [11]	PHP	Validate JSON data

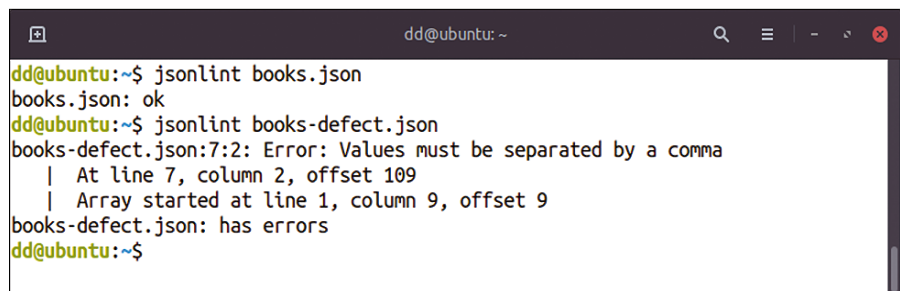


Figure 2: JSONLint helps you identify errors in JSON data.

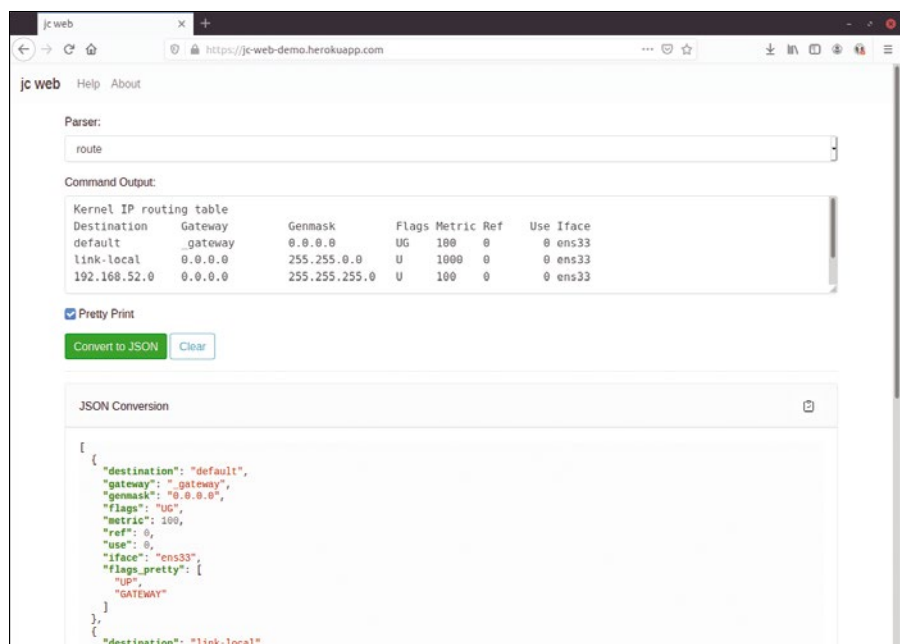


Figure 3: The demo app shows output converted into JSON format using route as an example.



**Listing 6: Printing with aeson-pretty**

```
$ cat book-inventory.json | aeson-pretty
{
  "book": [
    {
      "publication": "1994",
      "author": "Stephen Fry",
      "title": "The Hippopotamus"
    },
    {
      "publication": "2009",
      "author": "Ian Rankin",
      "title": "Set In Darkness"
    },
    {
      "publication": "1989",
      "author": "Ken Follett",
      "title": "The Pillars of the Earth"
    }
  ]
}
```

a variant that is missing a comma as a separator.

JSONLint was written in PHP. If you are looking for an alternative, you may also be able to get by with jq. If jq fails to parse the JSON data, it returns a number greater than zero, otherwise zero. Listing 8 shows the output for a deliberate error.

**Converting Output to JSON**

All Unix/Linux tools have their own specific output format. With a clever combination of grep, sed, and awk, you can break down the output and create the format you need for further processing. This approach sounds simple, but it often feels like walking up the stairs backwards while balancing a crystal vase on your head.

On the other hand, if every Unix/Linux tool had a --json switch and used it to create output in JSON format, the output could be parsed in a standardized way. However, the world still has a long way to go before this happens, so a workaround is needed. Jc and jo can both read the output from a tool, convert the output, and flip it back to the standard output in JSON format.

The list of output formats that jc understands is quite long and includes the output from df, du, lsblk, crontab, netstat, and lsof. Figure 3 shows output

**Listing 7: Compact Output with aeson-pretty**

```
$ cat book-inventory.json | aeson-pretty -c
{"book":[{"publication":"1994","author":"Stephen Fry","title":"The Hippopotamus"},
{"publication":"2009","author":"Ian Rankin","title":"Set In Darkness"},
{"publication":"1989","author":"Ken Follett","title":"The Pillars of the Earth"}]}
```

**Listing 8: Finding Errors in JSON Data with jq**

```
$ cat book-inventory-broken.json | jq .
parse error: Expected separator between values at line 7, column 5
$ echo $?
4
```

**Listing 9: Adding Environment Variables**

```
$ jo timeofday="$(date +%c" home=$HOME
{"time of day":"Mon 12 Oct 2020 17:06:30 CEST","home":"/home/frank"}
```

from the route command, as processed by the jc web demo page [13]. On the demo page, you can select the desired Unix/Linux command or data format at the top, and then copy the associated output into the input box. Click on Convert to JSON to create the output below – each entry is a JSON element. Use the Pretty Print checkbox to specify whether the output should be a compact one-liner or a prettied-up, longer version.

Building complicated JSON files yourself and counting parentheses – that was yesterday. Today, jo does it for you. Jo expects the key-value pairs as parameters and screws together a corresponding JSON output from them. Figure 4 shows the output for the two parameters magazine and issue.

Because jo receives the key-value pairs as parameters when called, variable content from the shell is no longer a problem. See Listing 9, which shows variables for today’s date and home directory.

**Interactive Filters**

Tools like jq, jid, and Jshon can filter the output if you only need part of the data. Earlier you learned that passing a . to jq outputs the entire dataset. With the appropriate call, you can filter the data and extract the author, title, and publication data from a JSON file with book publishing data. Using .book[], you first

narrow down the search to the book list; then you filter all items with the publication key using a pipe (|).

Figure 5 shows the results from jqplay – a simple playground tool in the web browser whose contents are passed to jq for processing. If you change the filter or the output in the two input fields on the left, the output on the right adjusts.

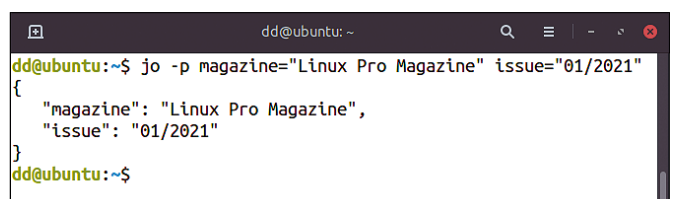
Jid stands for JSON Interactive Digger. A call to cat book\_inventory.json | jid lets you browse a JSON file interactively.

We have not yet discovered any special programs with graphical interfaces for editing JSON data. All text editors offer syntax highlighting and thus simplify editing. We were particularly impressed by the web-based JSON Viewer [14] editor, which offers a graph structure in addition to an object-based display. We fed JSON Viewer our book list for Figure 6.

**Downstream Processing**

In addition to the tools that output and format JSON data for the user are several tools that provide downstream processing of JSON as input in other programs (post-processing).

If you receive JSON data via an interface, it is good programming practice to



**Figure 4: Jo lets you compile JSON output without any serious overhead.**

sanity check the received data before further processing. The sanity check includes two stages:

- Syntactic correctness – is the spelling correct? Do all brackets (equal number of opening and closing brackets), commas, and quotation marks fit the bill?
- Correctness of data fields – does the received data structure match the data definition (JSON schema)?

For the first question, it is best to use JSONLint, which is described earlier in this article. For the second stage, you need the JSON schema that describes the data structure. You then compare this description with the received data.

On [json-schema.org](http://json-schema.org), you will find an overview of validators [15], sorted by the various programming languages in which they were developed. For example, consider the `validate-json` tool implemented in PHP [16]. If you are more into Python, `jsonschema` [17] serves the same purpose. The call to the two tools is identical.

### Defining the JSON Schema

Listing 10 shows the JSON schema with which you define the exact format of your data structure. The schema matches the book inventory used earlier in this article. The schema was stored in the `book-inventory-schema.json` file in the local directory.

The schema definition references the JSON standard used (the draft from September 2019, in this case) in the second line. The definition contains a number of keywords. Table 3 explains these keywords in more detail; a complete list of all supported keywords is available at [json-schema.org](http://json-schema.org) [18].

The next task is to validate the records by checking whether they match the specified schema. Listing 11 shows a single record from the book inventory in readable format. The compact version of the record contains all the parentheses and fields in a single line.

The `validate-json` tool expects two parameters in the call, the dataset and the schema (Listing 12). If everything goes well, the output does not cause any further feedback (line 2); otherwise, `validate-json` grumbles (lines 4 and 5). To provoke the error message starting in line 4, we turned the numeric specification for the year of publication (1994)

into a string "1994", which means that the data type in the dataset no longer matched the stored data type in the JSON schema. `validate-json` has every reason to complain.

Some programming languages also offer suitable helper libraries. In Python, for example, you can use `jsonschema`, and

for NodeJS, you can use the Express framework.

### Processing JSON

The list of command-line tools and helpers that read, search on, and modify JSON data is quite extensive. We stopped counting after more than 20 entries (see

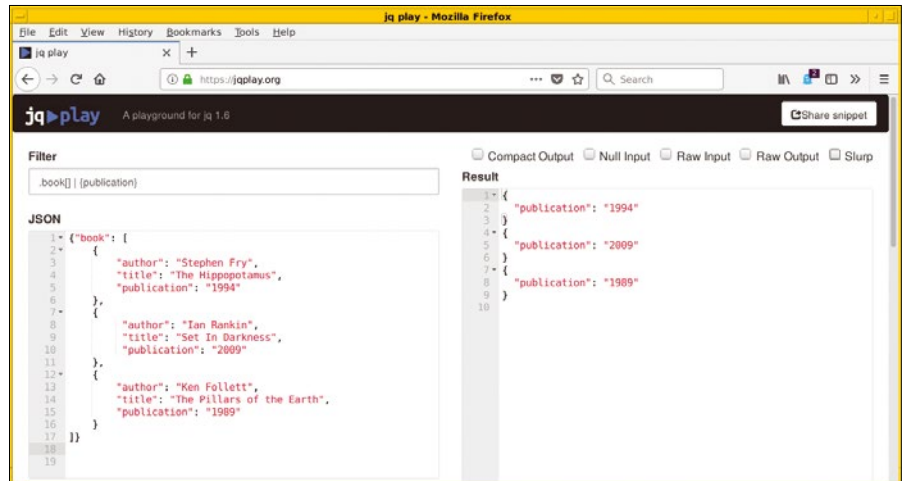


Figure 5: Jq can filter JSON output by arbitrary criteria.

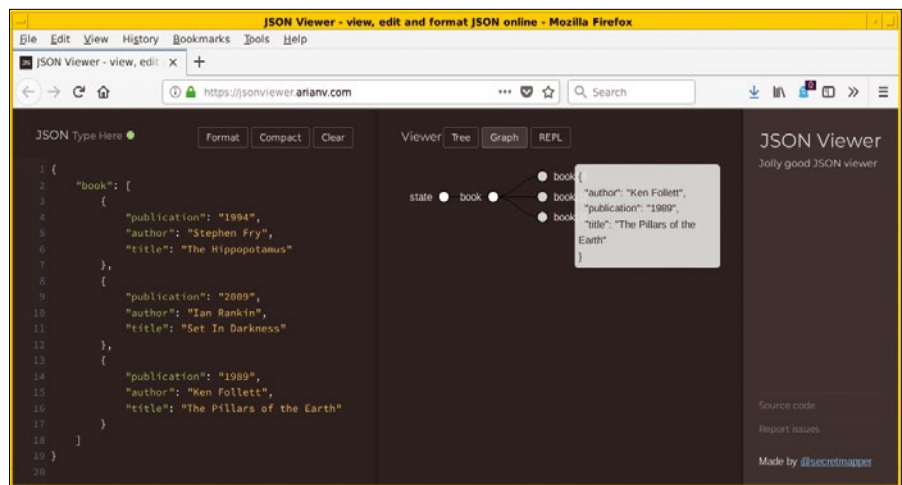


Figure 6: The web-based JSON Viewer shows data links as a graph.

Table 3: JSON Keywords

Keyword	Description
<code>\$schema</code>	Description of the schema specification
<code>title</code>	Title of the schema
<code>type</code>	Type of JSON data
<code>properties</code>	Properties of each value (key and values allowed for the field)
<code>required</code>	List of required properties
<code>properties.type</code>	Data type of an entry
<code>properties.minimum</code>	Minimum value of an entry
<code>properties.maximum</code>	Maximum value of an entry
<code>properties.minLength</code>	Minimum number of characters for an entry
<code>properties.maxLength</code>	Maximum number of characters for an entry
<code>properties.pattern</code>	Regular expression for a comparison with the value of an entry

Table 4 for a sample). Developer Ilya Sher maintains a useful, commented overview of options [19].

Jtbl, for example, takes JSON records and knits a pretty table from them. In Figure 7, you can see how this table looks for the book inventory. Each record is shown in a separate row. Jtbl can only cope with flat JSON structures. It cannot handle nesting so far.

## Element-by-Element Access

A tool like jq can dig out individual elements from the JSON data stream using expressions, but this approach is cumbersome for highly nested data structures; it is far easier to use a path specification. Tools like JMESPath [20] (pronounced “James Path”) and JSONPath [21] are similar to XPath for XML. JMESPath is available in Python, PHP, JavaScript, or even Lua; JSONPath is available in JavaScript, PHP, and Java.

These tools enable more complex expressions. Table 5 shows you a selection.

### Listing 10: JSON Schema

```
{
  "$schema": "http://json-schema.org/draft/2019-09/schema",
  "title": "Book",
  "type": "object",
  "required": ["author", "title", "publication"],
  "properties": {
    "author": {
      "type": "string",
      "description": "The author's name"
    },
    "title": {
      "type": "string",
      "description": "The book's title"
    },
    "publication": {
      "type": "number",
      "minimum": 0
    },
    "tags": {
      "type": "array",
      "items": {
        "type": "string"
      }
    }
  }
}
```

You read the expressions from left to right and name nodes or attributes in the order in which you want to work your way along the data structure. Two levels of nodes or attributes are separated by a period. Sets and patterns are specified in square brackets, for example, `book[*]` for all nodes of the book list. The specification `book[?author == `Ken Follett`]` takes all nodes from the dataset for which the attribute `author` has the value `Ken Follett`.

Please note the correct quotes when formulating the expressions. You need to quote values for comparison in the call in backticks (```), regardless of whether they are strings or numeric values.

Listing 13 shows the three expressions from Table 5 in action in a Python script. We used the JSON implementation of JMESPath here. Although the *Json* library is a fixed part of Python, JMESPath is one of the extras that you can install before using it, either via Pip or the package manager that comes with your Linux distribution. The corresponding

### Listing 11: JSON Record

```
{
  "author": "Stephen Fry",
  "title": "The Hippopotamus",
  "publication": 1994
}
```

```
dd@ubuntu:~$ cat books.json
{"book": [
  {
    "author": "Stephen Fry",
    "title": "The Hippopotamus",
    "publication": "1994"
  },
  {
    "author": "Ian Rankin",
    "title": "Set In Darkness",
    "publication": "2009"
  },
  {
    "author": "Ken Follett",
    "title": "The Pillars of the Earth",
    "publication": "1989"
  }
]}
dd@ubuntu:~$ cat books.json | jtbl
-----+-----+-----+
| book |
-----+-----+-----+
| [{"author": 'Stephen Fry', 'title': 'The Hippopotamus', 'publication': '1994'}, {'author': 'Ian Rankin', 'title': 'Set In Darkness', 'publication': '2009'}, {'author': 'Ken Follett', 'title': 'The Pillars of the Earth', 'publication': '1989'}] |
-----+-----+-----+
dd@ubuntu:~$
```

Figure 7: Representing records as a table.

### Listing 12: Calling validate-json

```
01 $ validate-json record.json bookinventory-schema.json
02 $
03 $ validate-json record.json bookinventory-schema.json
04 JSON does not validate. Violations:
05 [publication] String value found, but a number is required
```

### Table 4: Command-Line Tools

Tool	Application (selection)
faq, Xidel	Convert formats from and to JSON (BSON, Bencode, JSON, TOML, XML, YAML, etc.)
fx, gofx, jq, jid	Filter JSON data
jello	Filter JSON data with Python syntax
jtbl	Output to a table
Underscore	Processing via the command line

### Table 5: Expressions in JMESPath

Expression	Meaning
<code>book[*].title</code>	All book titles
<code>book[?author == `Ken Follett`].title</code>	All book titles by author Ken Follett
<code>book[?publication &gt; `1990`]</code>	All books published after 1990

**Listing 13:** find-json-path.py

```

01 import jmespath
02 import json
03
04 expression1 = jmespath.compile('book[*].title')
05 expression2 = jmespath.compile('book[?author == `Ken Follett`].title')
06 expression3 = jmespath.compile('book[?publication > `1990`].title')
07
08 with open("bookinventory.json") as jsonFile:
09     jsonData = json.load(jsonFile)
10
11 # book titles
12 print("Book title:")
13 bookTitles = expression1.search(jsonData)
14 for title in bookTitles:
15     print(title)
16
17 print(" ")
18
19 # all the books by Ken Follett
20 print("All books by Ken Follett:")
21 bookTitles = expression2.search(jsonData)
22 for title in bookTitles:
23     print(title)
24
25 print(" ")
26
27 # all books published later than 1990
28 print("All books published later than 1990:")
29 books = expression3.search(jsonData)
30 for item in books:
31     author = item["author"]
32     title = item["title"]
33     publication = item["publication"]
34     print("Author      : %s" % author)
35     print("Title       : %s" % title)
36     print("Published: %i" % publication)
37     print(" ")

```

package for Debian GNU/Linux and Ubuntu goes by the name of *python3-jmespath*.

After first loading the two Python libraries, *json* and *jmespath*, in the script (lines 1 and 2), three expressions or search patterns are defined as objects with the names *expression1*, *expression2*, and *expression3*. If you are familiar with the Python regular expression library *re*, you will already know the procedure.

Lines 8 and 9 read the book inventory as a JSON file and *load()* the contents of the file as a dictionary into the *jsonData* variable. Searches over the book inventory rely on the *search()* method from the search pattern object. For example, the call to *expression2.search(jsonData)* searches out all book titles that belong to the author Ken Follett.

*search()* returns a list of search hits that you can output one by one in a *for* loop. Figure 8 shows the output of the search matches for all three previously defined search paths.

## JSON Libraries

If you prefer some other programming language instead of Python, you can still connect to JSON. Table 6 shows a selection of libraries and modules. If you have different implementations available for a programming language, it is difficult to make a recommendation without knowing the volume and structure of the JSON data you wish to process. After a benchmark test, you will be smarter [22] about what best suits your case.

Listing 14 shows how to access JSON objects in the Go programming language. After importing the two modules *encoding/json* and *fmt*, you create a *Book* data structure that includes three variables: *Author*, *Title*, and *Publication*. You access this data structure in the *main()* function by declaring a *book* variable with this type in it.

The *bookJson* variable acquires the record for a book. Using the *Unmarshal()* method from the *json* module, you unpack the record byte by byte and assign the contents to the components from *book*. Then, using the *Println()* method, you output the contents of the components. For more information on processing, see Soham Kamani's blog [23], which is definitely a worthwhile read.



**Figure 8:** Selecting records and attributes by path.

## Listing 14: extract-json.go

```

package main

import (
    "encoding/json"
    "fmt"
)

type Book struct {
    Author string
    Title string
    Publication string
}

func main() {
    bookJson := `{"author": "Stephen Fry", "title": "The Hippopotamus",
                "publication": "1994"}`

    var book Book

    json.Unmarshal([]byte(bookJson), &book)
    fmt.Println("Author: ", book.Author)
    fmt.Println("Title: ", book.Title)
    fmt.Println("Publication: ", book.Publication)
}

```

Now save Listing 14 to the `extract-json.go` file and run the code. You'll see something like the output in Listing 15.

## Outlook

Using JSON is a good choice if the data conforms to the supported formats, the strings are not arbitrarily long, and you only need to implement the data exchange, while the documentation of the

data is managed elsewhere. Another strength of JSON is that many languages can process it.

With large volumes of data, however, the file size can have a detrimental effect on the processing speed. In those cases, a different format such as Google's Protobuf [24] could offer an alternative. For more information on serialization formats and the practical handling of JSON, see the examples in the Jupyter tutorial [25]. ■■■

## Listing 15: Output

```

$ go run extract-json.go
Author: Stephen Fry
Title: The Hippopotamus
Publication: 1994

```

## Thank you

The authors would like to thank Gerold Rupprecht for his criticism and suggestions during the preparation of the article.

## Info

- [1] JSON: <https://www.json.org/>
- [2] Jupyter Notebook: <https://jupyter.org/try>
- [3] GeoJSON: <https://geojson.org>
- [4] RFC 8259: <https://tools.ietf.org/html/rfc8259>
- [5] aeson-pretty: <https://github.com/informatikr/aeson-pretty>
- [6] jc: <https://github.com/kellyjonbrazil/jc>
- [7] jid: <https://github.com/simeji/jid>
- [8] jo: <https://github.com/jpmens/jo>
- [9] jq: <https://github.com/stedolan/jq>
- [10] Jshon: <http://kmkeen.com/jshon>
- [11] JSONLint: <https://jsonlint.com>
- [12] jq cheat sheet: <https://lzone.de/cheat-sheet/jq>
- [13] jc demo website: <https://jc-web-demo.herokuapp.com>
- [14] JSON Viewer: <https://jsonviewer.arianv.com>
- [15] JSON validators: <https://json-schema.org/implementations.html#validators>
- [16] validate-json: <https://github.com/justinrainbow/json-schema>
- [17] jsonschema: <https://github.com/Julian/jsonschema>
- [18] Validation of JSON data: <http://json-schema.org/draft/2019-09/json-schema-validation.html>
- [19] "List of JSON tools for command line": <https://ilya-sher.org/2018/04/10/list-of-json-tools-for-command-line/>
- [20] JMESPath: <https://jmespath.org>
- [21] JSONPath: <https://code.google.com/archive/p/jsonpath/>
- [22] "Choosing a faster JSON library for Python": <https://pythonspeed.com/articles/faster-json-library/>
- [23] "How to Parse JSON in Golang": <https://www.sohamkamani.com/blog/2017/10/18/parsing-json-in-golang/>
- [24] Protobuf: <https://developers.google.com/protocol-buffers>
- [25] Serialization formats/JSON (Jupyter Tutorial): <https://jupyter-tutorial.readthedocs.io/en/latest/data-processing/serialisation-formats/json.html>

## Author

**Veit Schiele** is founder and CEO of Cusy GmbH, which provides privacy-compliant tools for software development and a platform for research software and data. He is the author of Jupyter and PyViz tutorials. **Frank Hofmann** works mostly on the road as a developer, trainer, and author. His preferred work locations are Berlin, Geneva, and Cape Town. He is one of the authors of the Debian package management book.

Table 6: Selection of JSON Libraries

Language	Libraries (Selection)
Go	encoding/json
LISP	CL-JSON
Lua	json.lua
NodeJS	Express
Perl	JSON::Parse, JSON::PP, JSON::XS
PHP	JSON
Python	simplejson, hyperjson, json, jsonschema, orjson, RapidJSON, UltraJSON, pandas
Ruby	JSON
Tcl	json



Setting up a dgamelaunch game server

# A Blast from the Past

If you are into retrogaming, dgamelaunch lets you set up a server to play Roguelike games and compete with friends, all while preserving a piece of gaming history. *By Rubén Llorente*

In the early '80s, Rogue, a game that would overhaul the gaming world, was released. According to Roguelike Gallery, Rogue first ran on a PDP-11 machine running Unix v6 [1]. Primitive by today's standards, Rogue popularized the Roguelike genre despite not being the first game of its kind. Rogue's influence can be seen in modern commercial titles. Rogue's source code is still available, as are many games that pre-date Linux.

In this article, I show how to set up a server to host antique terminal games. Why would you want to do this? Besides preserving a piece of history, playing these games is fun despite their age. In addition, running games on a specific server allows you to keep scoreboards, letting you compete against friends on a shared server. Last, but not least, setting up a game server is an instructive exercise.

## Service Architecture

This article assumes you are using Debian, which unlike early 16-bit Operating Systems, is capable of using modern connectivity protocols.

In the service architecture, *openbsd-inetd* takes an incoming Telnet connection from the user and passes it to a Telnet daemon (*telnetd*). The Telnet daemon calls *dgamelaunch*. *Dgamelaunch*

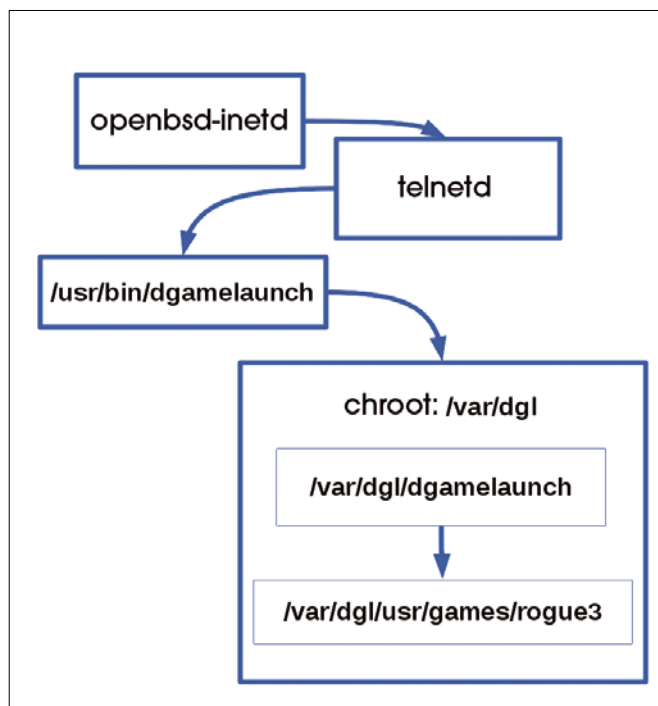


Figure 1: The game server's service architecture.

Photo by Kevin Borrill on Unsplash

chroots into `/var/dgl` and provides a limited shell that allows the user to play games (Figure 1).

The core of this game server is the game launcher, which is a limited shell that is used to authenticate the player and start games. The launcher I am using, `dgame launch` [2], is surprisingly capable for such a small C program. `Dgame launch` records games, allows users to watch other players, and relays messages from one person to another.

The game server will be available over Telnet, a ubiquitous protocol that most operating systems support out of the box. The idea is to let the player connect to the server using a Telnet client and offer him or her a `dgame launch` shell that will allow them to select which games to play. Beware: Telnet is not secure (see *Upgrading from Telnet* for a safer approach).

### Upgrading from Telnet

While the setup described in this article is safe to deploy on a home LAN, it is unsafe to use for a game server accessible over the Internet.

Armies of bots roam the Internet, storming Telnet services and trying to gain access to them via brute force attacks. Bots can also automatically create massive numbers of accounts in the hope of using them later for nefarious purposes. Although the real damage a bot can cause after registering or gaining access into an already existing account is negligible, `dgame launch` is not well equipped for dealing with these attacks directly, and neither is `telnetd` in this configuration.

Another issue associated with Telnet is that it lacks secure authentication – at least the way it is demonstrated here. Usernames and passwords are sent unencrypted over the line. That may be acceptable for a home network, but not for the Internet.

Organizations who offer retrogame servers over the Internet favor SSH instead, which offers encrypted connections and better support for dealing with bot attacks. SSH also does not require `inetd`. Using SSH, however, is outside of the scope of this article.

In order to mitigate the damages that may be caused by malicious users, `dgame launch` is designed to chroot into an isolated part of the operating system and drop privileges.

## Chroot Security

`Dgame launch` uses `chroot()` as a security layer. Roughly speaking, when you chroot into a directory, you are running a process inside that directory, with the process viewing the directory as the root (`/`) of the filesystem hierarchy. In essence, it means any process running within the chroot cannot access files placed outside of the chroot directory. This makes `chroot()` a poor man's isolation technique. A server running within a chroot won't be able to wreak havoc in the rest of the filesystem if the service is compromised.

However, chroots are breakable by design, making them less than ideal for isolating a service from the rest of the operating system. A root-owned process within a chroot can create block device nodes. If a chroot includes a hard link that points to an external resource, the resource can be abused from within the chroot. Also, a root-owned process can break out of the chroot directory.

Consequently, `chroot()` is no substitute for proper privilege separation. To avoid this problem, `dgame launch` drops privileges immediately after chrooting. Unprivileged processes can't escape from a chrooted environment so easily.

## Installing dgame launch

Unfortunately, `dgame launch` is not packaged by any major Linux distribution. At the time of writing, the project has had no official release since 2011. However, `dgame launch` is used by many current projects, including Roguelike

Gallery, Dungeon Crawl Stone Soup [3], and the public NetHack server NAO [4]. As a result, there is a constant influx of patches from the community should you ever need them, despite the official project appearing to be dead.

The following command (which must be run as superuser) installs everything you need in order to compile `dgame launch`:

```
# apt-get install automake autoconf \
  build-essential git bison sqlite3 \
  libsqlite3-dev curl unzip groff \
  libncurses-dev flex-old
```

You can get the source code from GitHub using the following command:

```
$ git clone \
  https://github.com/paxed/dgame launch.git
```

Then, compile the program. If you intend to enable `sqlite` for managing your users and plan to use `/var/dgl` as a chroot directory, you may configure and compile `dgame launch` as follows:

```
$ cd dgame launch
$ ./autogen.sh --enable-sqlite \
  --enable-shmem \
  --with-config-file=\
  /var/dgl/etc/dgame launch.conf
$ make
```

This prepares `dgame launch` to load its configuration from `/var/dgl/etc/dgame launch.conf` and to use `sqlite` as the user database.

### Listing 1: Dgl-create-chroot Configuration

```
CHROOT="/var/dgl/"
USRGRP="games:games"
SQLITE_DBFILE="/dgl_dir/dgame launch.db"

# Leave this variable empty to skip installing gzip in the chroot
COMPRESSBIN=""

# The script is Nethack centric. Leave some variables blank since
# we have no use for them.
NETHACKBIN=""
NH_PLAYGROUND_FIXED=""

# There Nethack related variables must be set even if we have no
# use for them.
NHSUBDIR="/nh343/"
NH_VAR_PLAYGROUND="/nh343/var/"
```

Next, build the chroot directory that dgame-launch will switch into when a user connects to the server. Dgame-launch provides a script, `dgl-create-chroot`, to do this. Open `dgl-create-chroot` with any text editor and change the configuration variables to your liking. The script is NetHack-centric, because NetHack is the most popular Roguelike game in existence, but you can leave the NetHack-related variables unmodified if you don't plan to run it at all. See Listing 1 for the variables you can set.

Once the configuration is done, save the script and run it as root:

```
# bash dgl-create-chroot
```

If you use the example values from Listing 1, the `/var/dgl` directory will be created and populated with all the necessary libraries and configuration files to run a chroot.

Finally, install dgame-launch in `/usr/bin`, with the `setuid` bit set.

```
# cp dgame-launch /usr/bin/
# chmod 4755 /usr/bin/dgame-launch
```

## Installing Games

Installing the original Rogue seems appropriate for this article. Roguelike Gallery hosts builds for many early Rogue-

likes. John “Elwin” Edwards, Roguelike Gallery’s creator, has done an amazing job of keeping and updating these antique games’ source code to ensure they can run on modern operating systems. Roguelike Gallery also provides precompiled binaries [5].

I keep a convenient copy of Elwin’s Roguelike collection on a personal server. You may download it with the following command:

```
$ curl -LO https://gopher.OperationalSecurity.es/Software/Early%20Roguelikes/ElwinR-r1-74351bf23e5e.zip
```

Compile Rogue v3 (the earliest version of Rogue that was widely available) with:

```
$ unzip ElwinR-r1-74351bf23e5e.zip
$ cd ElwinR-r1-74351bf23e5e/rogue3
$ autoreconf
$ ./configure --enable-savedir=/var/games/rogue3/save --enable-scorefile=/var/games/rogue3/rogue.scr --enable-logfile=/var/games/rogue3/rogue.log
$ make
```

The `enable-savedir`, `enable-scorefile`, and `enable-logfile` switches are necessary to compile a game for system-

wide installation. Keep in mind that Rogue will live in a chroot and won't be able to modify the rest of the operating system: For the game, the chroot directory will be all there is to the operating system.

Create the appropriate directories in the chroot and move the binary file to its final destination:

```
# cd /var/dgl
# mkdir -p var/games/rogue3/save
# mkdir -p usr/games
# mkdir -p dgl_dir/inprogress-rogue3
# cp $user_home/ElwinR-r1-74351bf23e5e/rogue3/rogue3 usr/games/
# chown -R games:games var/games dgl_dir
```

You can install additional games using similar steps. Keep in mind that you also must copy the libraries required by those games inside the chroot folder.

Rogue requires the appropriate ncurses library to live within the chroot.

```
# cp /lib/x86_64-linux-gnu/libncurses.so.6 /var/dgl/lib/x86_64-linux-gnu/
```

## Configuring dgame-launch

The game launcher’s main configuration file resides in `/var/dgl/etc/dgame-launch.conf`. Listing 2 shows an exam-

### Listing 2: dgame-launch.conf

```
chroot_path = "/var/dgl"
dglroot = "/dgl_dir/"
banner = "/dgl-banner"
shed_uid = 5
shed_gid = 60
commands[register] = mkdir "%ruserdata/%n",
    mkdir "%ruserdata/%n/ttyrec",
    mkdir "%ruserdata/%n/ttyrec/rogue3"
commands[login] = mkdir "%ruserdata/%n",
    mkdir "%ruserdata/%n/ttyrec",
    mkdir "%ruserdata/%n/ttyrec/rogue3"
menu["mainmenu_anon"] {
    bannerfile = "/dgl_menu_main_anon.txt"
    commands["l"] = ask_login
    commands["r"] = ask_register
    commands["w"] = watch_menu
    commands["q"] = quit
}
menu["mainmenu_user"] {
    bannerfile = "/dgl_menu_main_user.txt"
    commands["c"] = chpasswd
    commands["e"] = chmail
    commands["w"] = watch_menu
    commands["3"] = play_game "RogueV3"
    commands["q"] = quit
}
menu["watchmenu_help"] {
    bannerfile = "/dgl_menu_watchmenu_help.txt"
    commands["qQ "] = return
}
DEFINE {
    game_path = "/usr/games/rogue3"
    game_name = "Rogue V3 (3.6)"
    short_name = "RogueV3"
    game_args = "rogue3", "-n", "%n"
    inprogressdir = "%rinprogress-rogue3/"
    ttyrekdir = "%ruserdata/%n/ttyrec/rogue3/"
    commands = cp "/var/games/rogue3/save/%u-%n.r3sav" "/var/games/rogue3/save/%u-%n.r3sav.bak"
```



Listing 3: dgl\_menu\_main\_user.txt

```
##
## $VERSION - network console game launcher
## Copyright (c) 2000-2009 The Dgame launch Team
## See http://nethack.wikia.com/wiki/dgame launch for more info
##
## Games on this server are recorded for in-progress viewing and playback!

Logged in as: $USERNAME

c) Change password          e) Change email address
w) Watch games in progress

3) Play Rogue V3

q) Quit

=>
```

ple to get you started. You may find more example configuration files in dgame launch's source code tarball, with the meaning of the variables properly explained.

The `shed_uid` and `shed_gid` variables define the user ID and group ID that

dgame launch will drop privileges to after chrooting. In the example shown in Listing 2, this would be equivalent to `games:games` in a default Debian install. `commands[login]` and `commands[register]` tell dgame launch which actions to perform when a player logs in or registers,

respectively. In Listing 2, they build a directory tree when the user registers and rebuild it if the user logs in and the tree does not exist.

The `DEFINE` clause provides a configuration for loading Rogue, `game_path` defines the location of the `rogue3` binary within the chroot, while `commands` backs up each player's saved files each time the game is launched. `ttyrecdir` sets the directory in which the game session is recorded, just in case you want to watch your games later.

The final step is to configure the menus the users will see when logged into the game server. The default configuration suffices for testing, with the exception of the menu located at `/var/dgl/dgl_menu_main_user.txt`. By default, the menu is NetHack-centric. Listing 3 provides you with an appropriate alternative.

## Making the Service Available

You can test whether dgame launch works by invoking the `dgame launch` command as any regular user:

Shop the Shop → [shop.linuxnewmedia.com](http://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?

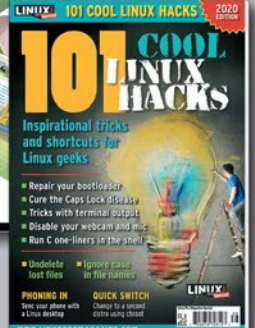
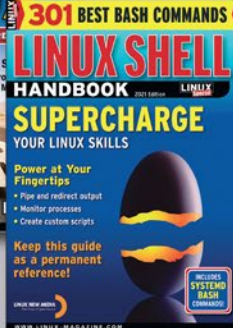
Searching for that back issue you really wish you'd picked up at the newsstand?

shop.linuxnewmedia.com

DIGITAL & PRINT SUBSCRIPTIONS



SPECIAL EDITIONS



```
$ /usr/bin/dgamelaunch
```

If everything works as intended, dgamelaunch will chroot into `/var/dgl`, and you'll be presented with a menu (Figure 2), from which you can create a user account for the game service, play games, and watch other players.

In order to make the game available over Telnet, you must install the appropriate Telnet daemon and configure it. Dgamelaunch's README file offers instructions to do this. Begin by installing an `inetd` daemon and a Telnet server:

```
# apt-get install openbsd-inetd telnetd
```

OpenBSD's `inetd` is a superserver that takes incoming connections and passes them to the appropriate server, in this case `telnetd`. In order to make this configuration work, edit `/etc/inetd.conf` and ensure the line shown in Listing 4 is its only content.

This line instructs OpenBSD's `inetd` to call the Telnet daemon when a Telnet connection is received. In turn, the Telnet daemon is configured to use `/var/dgl/dgamelaunch` as a shell. Remember to

reload the `inetd` daemon for the configuration to take effect:

```
# systemctl reload inetd
```

## Recording Games

The server automatically records the games under `/var/dgl/dgldir/userdata/$user/ttyrec/rogue3`. The recordings can be watched using a `tttyrec` player, such as `tttyplay`. It is common for public dgamelaunch servers to offer the files for download via a web server.

Any user who connects while other people are playing may watch the ongoing games. If the game is patched for proper dgamelaunch integration, it is possible for the audience to send messages to the player. Games that support this include NetHack and Dungeon Crawl Stone Soup. Trying to deliver a message when a game has no `spooldir` variable set will crash dgamelaunch, but thankfully it will not affect the instances of the people who are running games.

## Conclusions

Setting a server to play old terminal games is not easy, but it is certainly doable.

There are lots of games that can be run within a dgamelaunch service. Debian has the `bsdgames` package in its repository, with many terminal games from the pre-Linux era, including `phantasia`, `battlestar`, and `trek`. With a little tweaking, incorporating these games into the game server is possible.

You can see dgamelaunch in action using Roguelike Gallery's SSH service [6] (use `rodney` for the username and `yendor` for the password). Instructions for playing Rogue are in the source tarball. If you prefer to play a modern game with a retro feel, the Dungeon Crawl Stone Soup team keeps a list of servers for playing their games [7]. ■■■

## Info

- [1] History of Rogue: <https://rlgallery.org/about/rogue3.html>
- [2] dgamelaunch's GitHub repository: [launch">https://github.com/paxed/dgamelaunch](https://github.com/paxed/dgame<span style=)
- [3] Dungeon Crawl Stone Soup: <https://crawl.develz.org/>
- [4] NAO: <https://alt.org/nethack/>
- [5] Roguelike Gallery: <https://rlgallery.org>
- [6] Roguelike Gallery's dgamelaunch SSH service: <https://rlgallery.org:8080>
- [7] Dungeon Crawl Stone Soup online: [http://crawl.chaosforge.org/Playing\\_online](http://crawl.chaosforge.org/Playing_online)

## Author

Rubén Llorente is a mechanical engineer, whose job is to ensure that the security measures of the IT infrastructure of a small clinic are both law compliant



and safe. In addition, he is an OpenBSD enthusiast and a weapons collector.

```
##
## dgamelaunch 1.5.1 - network console game launcher
## Copyright (c) 2000-2009 The Dgamelaunch Team
## See http://nethack.wikia.com/wiki/dgamelaunch for more info
##
## Games on this server are recorded for in-progress viewing and playback!

Not logged in.

l) Login
r) Register new user
w) Watch games in progress
q) Quit

=>
```

Figure 2: The dgamelaunch menu.

## Listing 4: `inetd.conf`

```
telnet stream tcp nowait root.root /usr/sbin/tcpd /usr/sbin/in.telnetd -h -L /var/dgl/dgamelaunch
```

## The sys admin's daily grind: Zint

## Checklist

Doing a hardware inventory in a data center is anything but a piece of cake. In order to quickly assign devices to the appropriate database entry, Charly provides each newly acquired system with a QR code sticker with the help of Zint. *By Charly Kühnast*

**W**hen you need to manage large numbers of devices, there is no avoiding centralized data management. In the simplest case, this can be a wiki, with one entry per system. This will include, for example, the date of purchase, the length of the warranty period or maintenance contract, any repairs that have already been made, and the rack number where the device is installed (finding the hardware in a larger data center can be time-consuming). I then encode the URL of the wiki entry as a barcode or QR code, print it on self-adhesive film, and stick it on the device.

I generate the codes for this with Zint [1]. Many distributions have Zint

on board; if not, it is quickly compiled from the GitHub repository. You must have libpng in place; otherwise, Zint will not generate images. Those who now want to generate codes are spoiled for choice: Zint knows dozens of variants (Figure 1). With `zint -t`, I can display their names.

I know a few of these codes, like EAN and QR, from everyday life. PDF417 (Figure 2) and its relatives can be found on the boarding passes of many airlines. And there just happens to be a cold medicine bottle on the table in front

of me that has a PZN barcode. I can see from a web page [2] for generating barcodes that this is used on pharmaceuticals in Germany. On the same website – funnily enough, it uses Zint itself – there are examples of the other types of code.

For inventory purposes, I use a classic QR code. I can encode all ASCII characters in it, but I have to avoid nonstandard characters like accents and umlauts. Using the call from Listing 1, I create a QR code as a PNG that reveals the URL for *Linux Magazine's* website (Figure 3).

In Listing 1, I use `-b 58` to select QR as the code type. The parameter `-d` for data always has to be at the end: Zint blithely ignores all the options that follow. As long as I stick to this, the barcode generation routine works like clockwork, which gives me one less excuse to put off the pesky inventory process. ■■■

1: Code 11	52: PZN	96: DPD Code
2: Standard 2of5	53: Pharma Two-Track	97: Micro QR Code
3: Interleaved 2of5	55: PDF417	98: HIBC Code 128
4: IATA 2of5	56: Compact PDF417	99: HIBC Code 39
6: Data Logic	57: Maxicode	102: HIBC Data Matrix
7: Industrial 2of5	58: QR Code	104: HIBC QR Code
8: Code 39	60: Code 128-B	106: HIBC PDF417
9: Extended Code 39	63: AP Standard Customer	108: HIBC MicroPDF417
13: EAN	66: AP Reply Paid	110: HIBC Codablock-F
14: EAN + Check	67: AP Routing	112: HIBC Aztec Code
16: GS1-128	68: AP Redirection	115: DotCode
18: Codabar	69: ISBN	116: Han Xin Code
20: Code 128	70: RM4SCC	121: RM Mailmark
21: Leitcode	71: Data Matrix	128: Aztec Runes
22: Identcode	72: EAN-14	129: Code 32
23: Code 16k	73: VIN	130: Comp EAN
24: Code 49	74: Codablock-F	131: Comp GS1-128
25: Code 93	75: NVE-18	132: Comp DataBar Omni
28: Flattermarken	76: Japanese Post	133: Comp DataBar Ltd
29: GS1 DataBar Omni	77: Korea Post	134: Comp DataBar Exp
30: GS1 DataBar Ltd	79: GS1 DataBar Stack	135: Comp UPC-A
31: GS1 DataBar Exp	80: GS1 DataBar Stack Omni	136: Comp UPC-E
32: Telepen Alpha	81: GS1 DataBar Exp Stack	137: Comp DataBar Stack
34: UPC-A	82: Planet	138: Comp DataBar Stack Omni
35: UPC-A + Check	84: MicroPDF	139: Comp DataBar Exp Stack
37: UPC-E	85: USPS Intelligent Mail	140: Channel Code
38: UPC-E + Check	86: UK Plessey	141: Code One
40: Postnet	87: Telepen Numeric	142: Grid Matrix
47: MSI Plessey	89: ITF-14	143: UPNQR
49: FIM	90: KIX Code	144: Ultracode
50: Logmars	92: Aztec Code	145: rMQR
51: Pharma One-Track	93: DAFT Code	

Figure 1: Zint can generate these codes.

## Listing 1: QR Code with Zint

```
$ zint -o ~/qr/linmagurl-qr.png -b 58 -d https://linux-magazine.com
```



Figure 2: PDF417 is the barcode format often used in the transportation industry.



Figure 3: The *Linux Magazine* URL as a QR code

## Info

[1] Zint: <https://github.com/zint/zint>

[2] Barcode generator: <http://www.barcode-generator.org>

## 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.



Go library shows filesystem changes across platforms

# Motion Sensor



Inotify lets applications subscribe to change notifications in the filesystem. Mike Schilli uses the cross-platform fsnotify library to instruct a Go program to detect what's happening. *By Mike Schilli*

In a file manager, have you ever observed how newly created files by other applications immediately appear in the displayed directory and wondered how this works? As continuous querying of the filesystem is out of the question for performance reasons, these applications use the Linux filesystem's inotify interface instead.

Operating systems implement the mechanism in different ways: Linux uses inotify, the Mac uses kqueue, and Windows comes with an unpronounceable extra. Fortunately, the Go library *fsnotify* on GitHub abstracts this proliferation

```
$ go mod init watch
go: creating new go.mod: module watch
$ go build watch.go
go: finding module for package
github.com/fsnotify/fsnotify
go: found github.com/fsnotify/fsnotify
in github.com/fsnotify/fsnotify v1.4.9
```

```
$ mkdir -p /tmp/test
$ ./watch
"/tmp/test/foo": CREATE
"/tmp/test/foo": WRITE
"/tmp/test/foo": CHMOD
"/tmp/test/foo": REMOVE
```

Figure 1: Listing 1 listens for filesystem messages in /tmp/test/ ...

```
$ ls /tmp/test
$ touch /tmp/test/foo
$ echo "more" >>/tmp/test/foo
$ chmod +x /tmp/test/foo
$ rm /tmp/test/foo
$
```

Figure 2: ... which were triggered by user actions in another terminal.

Listing 1: watch.go

```
01 package main
02
03 import (
04     "fmt"
05     "github.com/fsnotify/fsnotify"
06 )
07
08 func main() {
09     watcher, err := fsnotify.NewWatcher()
10     if err != nil {
11         panic(err)
12     }
13     defer watcher.Close()
14
15     go func() {
16         for {
17             select {
18                 case event, ok := <-watcher.Events:
19                     if !ok {
20                         return
21                     }
22                     fmt.Printf("%+v\n", event)
23                 }
24             }
25         }()
26
27         err = watcher.Add("/tmp/test")
28         if err != nil {
29             panic(err)
30         }
31
32         done := make(chan struct{})
33         <-done
34     }
```

to create a simple interface. This means that programmers only need to write their applications once to cover all platforms.

## No Stress

About 15 years ago, I wrote an article on this topic in my regular column [1]. At the time I used Perl, and the article relied on FUSE, a special filesystem. Today filesystem notifications are part of the standard.

In Go, the whole thing can be done without much fuss; Listing 1 shows a simple example just to get you warmed up. Figure 1 visualizes how an executable binary named `watch` is created from the Go code in `watch.go`, which then starts monitoring a newly created directory `/tmp/test/`. In another terminal, the user now enters the commands shown in Figure 2. They first create a new file in the test directory, write data to it, change its execution privileges,

and finally delete it with `rm`. Figure 1 confirms that the Go program actually sees all changes in real time and logs the actions.

To do this, Listing 1 retrieves the library code from GitHub in line 5, creates a new watcher as the first step in the `main` program, and calls `defer` to tell it to shut itself down at the end of the program.

Since filesystem monitoring with `fsnotify` is an asynchronous process

### Listing 2: fswatch.go

```

01 package main
02
03 import (
04     "fmt"
05     "github.com/fsnotify/fsnotify"
06     "log"
07     "os"
08     "os/user"
09     "path/filepath"
10     "strings"
11 )
12
13 func main() {
14     cur, err := user.Current()
15     dieOnErr(err)
16     home := cur.HomeDir
17
18     watcher, err := fsnotify.NewWatcher()
19     dieOnErr(err)
20     defer watcher.Close()
21
22     watchInit(watcher)
23
24     err = filepath.Walk(filepath.Join(home, "go"),
25         func(path string, info os.FileInfo, err error) error {
26             dieOnErr(err)
27             if info.IsDir() {
28                 err := watcher.Add(path)
29                 dieOnErr(err)
30             }
31             return nil
32         })
33     dieOnErr(err)
34
35     done := make(chan bool)
36     <-done
37 }
38
39 func eventAsString(event fsnotify.Event) string {
40     info, err := os.Stat(event.Name)
41     dieOnErr(err)
42     evShort := (strings.ToLower(event.Op.String()))[0:2]
43     dirParts := strings.Split(event.Name, "/")
44     pathShort := event.Name
45     if len(dirParts) > 3 {
46         pathShort = filepath.Join(dirParts[len(dirParts)-3 :
47                                 len(dirParts)]...)
48     }
49     return fmt.Sprintf("%s %s %d", evShort, pathShort, info.
50                             Size())
51 }
52
53 func watchInit(watcher *fsnotify.Watcher) {
54     go func() {
55         for {
56             select {
57                 case event, ok := <-watcher.Events:
58                     if !ok {
59                         return
60                     }
61                     if event.Op&fsnotify.Rename == fsnotify.Rename ||
62                       event.Op&fsnotify.Remove == fsnotify.Remove {
63                         continue
64                     }
65                     log.Printf("%s\n", eventAsString(event))
66                     info, err := os.Stat(event.Name)
67                     dieOnErr(err)
68                     if info.IsDir() {
69                         err := watcher.Add(event.Name)
70                         dieOnErr(err)
71                     }
72                     case err, _ := <-watcher.Errors:
73                         panic(err)
74             }
75         }
76     }()
77 }
78
79 func dieOnErr(err error) {
80     if err != nil {
81         panic(err)
82     }
83 }

```

using Go channels, line 15 calls `go func` to launch a goroutine, which immediately starts an infinite loop with a `select` statement. The latter blocks the flow of the goroutine until messages arrive from the `watcher.Events` channel, sent by the library code from `fsnotify`, which gets its clues directly from the operating system.

## Routines and Blocking

Meanwhile, the `main` program continues to flow unimpeded, and line 27 tells `fsnotify` via `watcher.Add()` that it wants to monitor the `/tmp/test/` directory. That's all there is to it in the `main` program.

But since `main` is supposed to continue running and listening for events in the Goroutine launched earlier; line 32 creates an unused channel just before the end in line 33. Alas, no message will ever arrive from this channel: Its only job is to block the `main` program until the user cancels it by pressing `Ctrl+C`.

## Non-Recursive

The Go library `fsnotify` only adds one directory to the watch list with each call to `Add()`. Recursive integration of an entire file tree is supposedly on the `fsnotify` project's roadmap, but it doesn't work at the moment. Therefore, the application has to weave its own surveillance net by issuing recursive calls down the directory hierarchy.

For example, to track which files the Go compiler downloads or generates in the directory hierarchy below `~/go/` in the user's home directory during the work phase, Listing 2 first has to delve the depths of the directory structure using the `Walk()` function from the standard `filepath` package starting in line 24.

As a parameter, the function expects a callback function that it will call for each filesystem entry it finds with the name and the `FileInfo` structure including the metadata, such as the file or directory, size in bytes, or access permissions. If an error occurred during the traversal, the `err` variable is set to the corresponding error instead.

## Short Shrift for Want of Space

To shorten the eternal error checks after function calls with `err != nil` requiring countless `if` conditions to a magazine-friendly code length, Listing 2 defines the `dieOnErr()` function in line 77; this simply aborts on any error. Under production conditions, you would want to log the error here instead and eventually handle it upstream, but then you wouldn't have to worry about the shortage of space inherent in print products. Ha!

The callback function then uses `IsDir()` to check for each newly discovered entry below the `~/go/` directory. If it is not a file but a subdirectory, the func-

tion sets another watcher for it with `Add()`. Each of these watchers consumes a file descriptor on Linux, and the operating system does not have an unlimited supply of them. The `ulimit -n` command shows the number of file descriptors available and gives the administrator the ability to increase the number. Under normal circumstances, however, the default of 1,024 descriptors is fine.

The call to `watchInit()` in line 22 of the main program determines what happens on arriving notify events. The function starts in line 51, with an asynchronous goroutine waiting for events from all defined watchers up to this point. If the event belongs to a newly generated directory within the monitored file structure, line 67 also sets a watcher for it. Luckily, the program does not need to worry about watchers being defined twice for the same entry: `fsnotify` is smart enough to ignore duplicates.

## Race Condition

However, this method is not completely reliable: If the tracker notices the genesis of a new directory, it needs to quickly set up a watcher for it to track future changes in the directory. But if an application creates files in the directory immediately after it is created, it could beat the tracker to it, and the tracker would not notice the change.

Also, the `Walk()` function originally called to collect all the subdirectories

does not continue to track symbolic links. If you want to do that, you have to resolve them with the `EvalSymlinks()` function, but watch it carefully to prevent the walker getting stuck in an infinite loop.

Events reporting the renaming or deletion of an entry are filtered out by the `if` condition in line 59, because an `os.Stat()` for

```
$ ./fswatch
2021/01/31 14:17:49 cr mod/cache/download 4096
2021/01/31 14:17:49 cr cache/download/github.com 4096
2021/01/31 14:17:49 cr pkg/mod/github.com 4096
2021/01/31 14:17:49 cr mod/github.com/fsnotify 4096
2021/01/31 14:17:49 cr github.com/fsnotify/fsnotify@v1.4.9 4096
2021/01/31 14:17:49 ch github.com/fsnotify/fsnotify@v1.4.9 4096
2021/01/31 14:17:49 ch github.com/fsnotify/fsnotify@v1.4.9 4096
2021/01/31 14:17:49 cr cache/download/golang.org 4096
2021/01/31 14:17:49 cr download/golang.org/x 4096
2021/01/31 14:17:50 cr pkg/mod/golang.org 4096
2021/01/31 14:17:50 cr mod/golang.org/x 4096
2021/01/31 14:17:51 cr golang.org/x/sys@v0.0.0-20191005200804-aed5e4c7ecf9 4096
2021/01/31 14:17:51 ch x/sys@v0.0.0-20191005200804-aed5e4c7ecf9/windows 4096
2021/01/31 14:17:51 ch x/sys@v0.0.0-20191005200804-aed5e4c7ecf9/unix 20480
2021/01/31 14:17:51 ch x/sys@v0.0.0-20191005200804-aed5e4c7ecf9/unix 20480
2021/01/31 14:17:51 ch x/sys@v0.0.0-20191005200804-aed5e4c7ecf9/plan9 4096
2021/01/31 14:17:51 ch x/sys@v0.0.0-20191005200804-aed5e4c7ecf9/plan9 4096
2021/01/31 14:17:51 ch x/sys@v0.0.0-20191005200804-aed5e4c7ecf9/cpu 4096
2021/01/31 14:17:51 ch x/sys@v0.0.0-20191005200804-aed5e4c7ecf9/cpu 4096
2021/01/31 14:17:51 ch golang.org/x/sys@v0.0.0-20191005200804-aed5e4c7ecf9 4096
2021/01/31 14:17:51 ch golang.org/x/sys@v0.0.0-20191005200804-aed5e4c7ecf9 4096
```

**Figure 3:** While the Go compiler loads sources from GitHub to compile a binary, `fsnotify` keeps track of the new files created in the process.

such an entry would fail. Line 63 prints all other events in a nicely formatted way.

Figure 3 shows what the watcher finds while running the Go compiler with:

```
go build fswatch.go
```

The watcher reports the creation of quite a few cache directories that help compile and include the *fsnotify* code downloaded from GitHub. The *cr* abbreviation here stands for the “create” action. Other messages have an identifier of *ch* for *chmod* when the compiler manipulates the access bits. If compiling a long Go program with many dependencies drags on and on, this little helper tells you exactly what the compiler is doing and you can guesstimate how long it will continue to run.

To neatly log the events that occur, the `eventAsString()` function reformats them in what is still a pretty rudimentary way starting at line 39. Line 42 truncates the

event names to their first two characters and converts them to lowercase with `ToLower()`. Line 43 splits long directory paths into their components; line 46 shortens them to the last three partial paths (if it finds any more). Using the `[m:n]` array slice syntax, it extracts the last three with `len(dirPaths)-3` for *m* and `len(dirParts)` for *n*.

The element at index *m* is by definition included in the result, while the one for *n* is not. Since `Join()` from the *filepath* package joins a variable number of individual elements rather than an array of partial paths, the final three dots turn the array slice coming from the slice operator into a flattened list of individual elements.

The whole thing could now be nicely embedded in a UI that constantly entertains the user with updates for each compiler run, so that you can immediately see whether it is just the network hanging or if the process is taking so

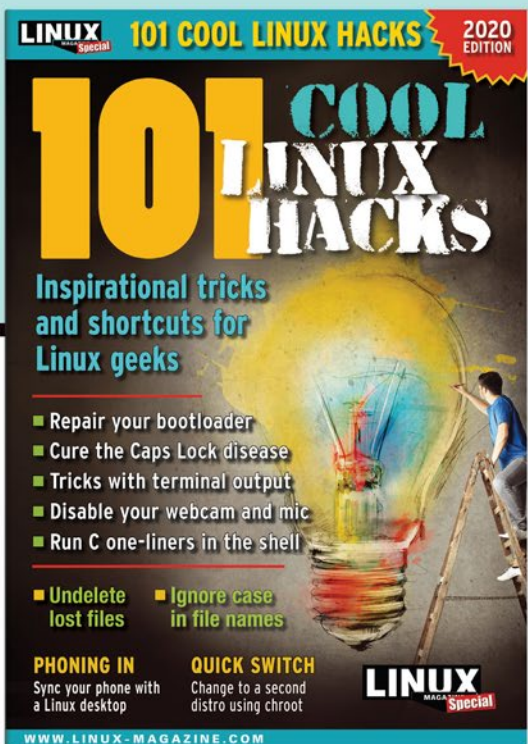
long because of code bloat or dependency hell. ■■■

#### Info

- [1] “Detecting system changes with Dnotify” by Mike Schilli, *Linux Magazine*, issue 63, February 2006, [https://www.linux-magazine.com/Issues/2006/63/Perl-noworries/\(language\)/eng-US](https://www.linux-magazine.com/Issues/2006/63/Perl-noworries/(language)/eng-US)
- [2] Listings for this article: <ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/247/>

#### 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.

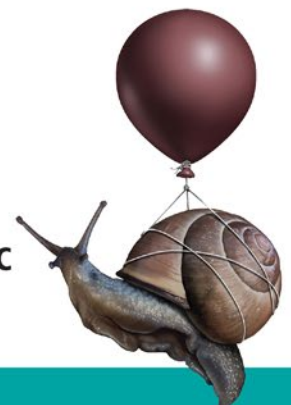


SHOP THE SHOP  
shop.linuxnewmedia.com

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



# MakerSpace

Access Raspberry Pi GPIO  
with ARM64 assembly

## The Three Rs

Reading, writing, and arithmetic with the Raspberry Pi in ARM64 assembly language. *By John Schwartzman*

In this article, I explore the Raspberry Pi's general purpose input/output (GPIO) system and look at how to use it to perform some basic input and output tasks with four separate programs that run simultaneously and communicate with each other. Table 1 lists the various programs discussed in this article [1].

### Recon

The first program, `gpiocount.asm`, counts up or down in various number systems: binary, octal, decimal, and hexadecimal – or really, any number system up to base 16. The count mode is changed with a switch that causes an interrupt to a loadable kernel module. That's the arithmetic portion of the project.

The `gpiocount` program writes its values to 4 bytes of memory that is shared with `gpiomux.asm`, which runs in the background, reads the 4 bytes written by

`gpiocount`, and writes the bytes into a four-digit, seven-segment display. All segment lines of the four displays are wired together inside the chip so that, to see the separate digits, you must turn on the four displays one at a time, faster than the eye can detect. That takes care of the reading, writing, and arithmetic.

Both `gpiocount` and `gpiomux` need to use the basic GPIO functions, so these functions have been placed into a shared library, `libgpio.asm`, which is assembled and linked to `libgpio.so` created by `Makefile`.

Each separate GPIO operation – reading a GPIO pin, writing a pin, setting a pin to input mode or output mode – is a function in `libgpio`. Mapping the GPIO's hardware registers into the memory spaces of `gpiocount` and `gpiomux` is another operation performed by `libgpio`. The `gpiocount` program creates a shared memory space and `gpiomux` reads the ad-

dress of this shared memory. Those two functions are also placed in `libgpio`.

Each of the three GPIO programs is placed in a separate ARM64 assembly language file. Both `gpiocount.asm` and `gpiomux.asm` have `main` functions, but they can be conditionally compiled to replace these functions with `countmain.cpp` and `muxmain.cpp` (the default in `Makefile`). Seeing the same code in both assembly language and C++ should make the code easy to follow.

Both `countmain.cpp` and `muxmain.cpp` print a lot of output to the console; however, I didn't bother with that in the assembly language versions. It's very easy to add `printf` calls to the assembly language mains, though: Simply place the string you want to print in the read-only data section (`.section .rodata`),

```
initstr: .asciz "countmain: initializing\n"
```

and place the following lines in the code section (`.section .text`):

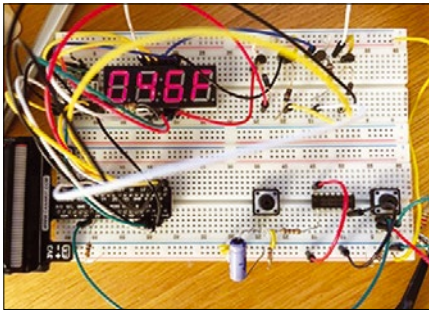
```
main:
    push2  x29, x30
    ...
    adr    x0, initstr
    bl    printf
    ...
    pop2   x29, x30
    ret
```

The fourth program is a loadable kernel module (LKM) written in C, `lkm_gpio.c`, that allows you to interrupt the kernel

**Table 1: Code Files**

File Name	Function
<code>gpiocount.asm</code>	Count up or down in different number systems
<code>gpiocount.h</code>	C++ header file with declarations for exports in <code>gpiocount.asm</code>
<code>countmain.cpp</code>	Optional C++ <code>main()</code> for <code>gpiocount.asm</code>
<code>gpiomux.asm</code>	Read and write four-digit, common-anode, seven-segment displays
<code>gpiomux.h</code>	C++ header file with declarations for exports in <code>gpiomux.asm</code>
<code>muxmain.cpp</code>	Optional C++ <code>main()</code> for <code>gpiomux.asm</code>
<code>libgpio.asm</code>	A shared library with GPIO functions used by <code>gpiocount.asm</code> and <code>gpiomux.asm</code>
<code>arm64_include.asm</code>	Contains constants and macros for ARM64 assembly language and describes the Raspberry Pi GPIO register map
<code>lkm_gpio.c</code>	An LKM that handles interrupts from user space





**Figure 1:** The breadboard and 40-pin header adapter with the Pi counting up in hexadecimal.

through a switch connected to a GPIO pin. It forces the current counter to end, whereupon the `gpioCount` function `main` starts a new task. The LKM module is a standalone program with a separate project directory and `Makefile`.

## Accessing the GPIO

The Raspberry Pi system-on-chip (SoC) contains 54, 32-bit hardware registers that provide access to the GPIO pins located in a 40-pin header. Although you could read and write to hardware registers, just as if they were memory locations, the register addresses are not within the range available to the programs.

Linux provides a way to map these addresses into virtual addresses that you can read and write. To do this, open the character device `/dev/gpiomem` as a file and call `mmap` to map the GPIO hardware register addresses. The code for this operation is located in the `mapOpen` subroutine in the file `libgpio.asm`.

Most of the code in this article is written in ARM64 Assembly Language. Assembly language gets you very close to the hardware and is often preferred over a high-level language when dealing with hardware. Figure 1 shows the breadboard connected to the Raspberry Pi through a 40-pin header and a breadboard adapter, and Figure 2 shows the wiring diagram for the device.

To save hardware, access to the GPIO registers is assigned to bits in a register, not the whole register. The `gpioMux.asm` program (Listing 1) provides a lookup table starting at line 279 that allows you to find the register addresses needed. For example, GPIO pin 6, which is assigned to segment e of the seven-segment display (lines 292-294), is in function select register 0, which manages GPIO pins 0-9. Bits 0-2 are assigned to GPIO pin 0,

bits 3-5 to GPIO pin 1, bits 18-20 to GPIO pin 6, and so on.

Why 3 bits? With 3 bits you can specify eight unique values. The value 000 means the pin is an input, 001 means the pin is an output, and the other six values 010-111 specify an alternative function for the GPIO pin. Here, I'll only deal with input and output. To write a 1 or a 0 to the GPIO output pin you have to write a bit into the set or clear register. The bit set is the same as the GPIO pin number, which is not the same as the 40-pin header number.

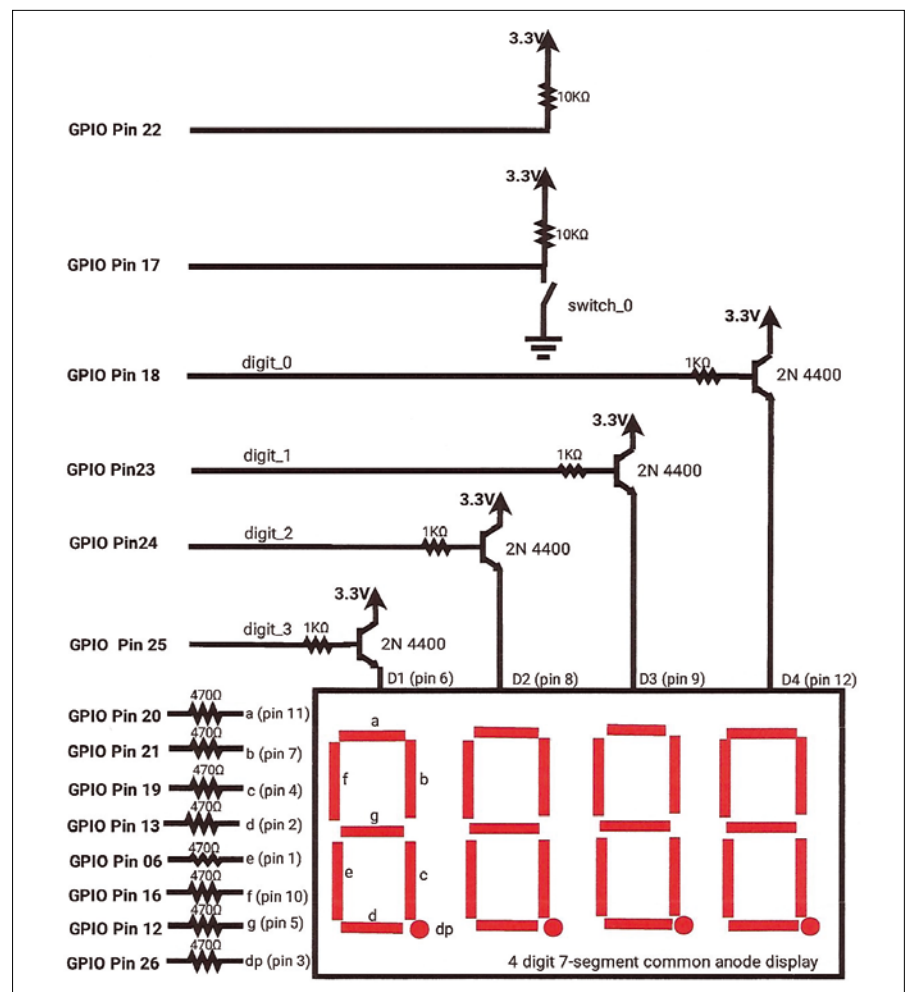
To set GPIO pin 6 to 0, write a 1 into bit 6 of the `GPCLR0` register. To make everything a little more complicated, if you only want to change one pin, you have to read a register first, change the bit that needs to be changed, and then write back the modified register.

Look at the `seg_e` entry in the lookup table and note the three values associated with that entry. The 0 means use `gpfsel0` (GPIO function select register 0) and 18 means write into bits 18, 19, and 20 of `gpfsel0` to set the register function. Fi-

nally, the 6 indicates the bit you have to set in `gpset0` to put GPIO pin 6 in a 1 state and the bit you have to set in `gpclr0` to put the pin in a 0 state. To set a GPIO pin high (1) or low (0), use `gpioSetState()` (in the `wroteDigit()` function). Please see `arm64_include.asm` for a description of the Raspberry Pi GPIO register map.

## Counting

The `gpioCount.asm` program has two methods that count in various number systems: `countUp(base)` and `countDown(base)`. Figure 3 shows a flow-chart of the `countUp` function. These routines write the count to four shared memory locations. The `gpioCount` program doesn't need to know much about the hardware; it needs to know just enough to test whether it has been interrupted. When the LKM is interrupted by `switch_0` pulling GPIO pin 17 low, it pulls GPIO pin 22 low, which `countUp()` and `countDown()` monitor to determine when they need to exit. They then restore GPIO pin 22 to a high state.



**Figure 2:** The breadboard wiring diagram.

## Listing 1: gpiomux.asm

```

001 //=====
002 // gpiomux.asm - Read and write 4-digit common-anode
    7-segment displays
003 // John Schwartzman, Forte Systems, Inc.
004 // 03/07/2021
005 // ARM64
006 //=====
007 .include "arm64_include.asm" // contains constants &
    macros
008
009 //===== CODE SECTION =====
010 .section .text
011
012 .ifdef OPT //===== use main from this file =====
013
014 .global main
015
016 //=====
017 main:
018     push2 x29, x30 // push fp & lr
019
020     mov    w0, #SIGHUP // prepare handleHangup - sig number
021     adr    x1, handleHangup // - function adr
022     bl    signal // invoke glibc signal() function
023
024     bl    initialize // set up gpio and shared memory
025     bl    readWrite // continuously display shared memory
026     bl    cleanUp // restore and unmap gpio
027
028     mov    w0, wzr // w0 = EXIT_SUCCESS
029     bl    exit // invoke glibc exit() function
030
031     pop2  x29, x30 // pop fp & lr
032     ret
033
034 //=====
035
036 .else // OPT != __MAIN__ //== use main from muxmain.cpp ==
037
038 .global initialize, cleanUp, readWrite, setExitFlag
039
040 .endif //=====
041
042 //=====
043 // Handle the SIGHUP signal: call cleanUp
044 handleHangup:
045     push2 x29, x30 // push fp & lr
046     bl    setExitFlag // tell readWrite to exit
047     pop2  x29, x30 // pop fp & lr
048     ret
049
050 //=====
051 setExitFlag: // tell readWrite to exit
052     adr    x0, exitFlag // write ONE to memory location
    exitFlag
053
053     mov    w1, #ONE
054     strb  w1, [x0]
055     ret
056
057 //=====
058 // Map virtual memory to /dev/gpiomem and set GPIO pins
    for input or output.
059 initialize:
060     push2 x29, x30 // push fp & lr
061
062     adr    x0, digits
063     bl    readSharedMemory
064     cmp    x0, #MINUS_ONE // success?
065     beq    fin // branch if no
066
067     adr    x0, memdev
068     adr    x1, gpiobase
069     bl    mapOpen // map the memory
070     cmp    x0, xzr // success?
071     bmi    fin // branch if no
072
073     adr    x29, gpiobase // save gpiobase
074     str    x0, [x29]
075
076     adr    x29, gpiobase // This step is necessary for all
077     ldr    x29, [x29] // GPIO activity. x29 => gpiobase
078
079     adr    x0, switch_0 // GPIO pin 17 used for switch
080     bl    gpioDirectionIn
081
082     adr    x0, seg_a // GPIO pin 20
083     bl    gpioDirectionOut
084
085     adr    x0, seg_b // GPIO pin 21
086     bl    gpioDirectionOut
087
088     adr    x0, seg_c // GPIO pin 19
089     bl    gpioDirectionOut
090
091     adr    x0, seg_d // GPIO pin 13
092     bl    gpioDirectionOut
093
094     adr    x0, seg_e // GPIO pin 06
095     bl    gpioDirectionOut
096
097     adr    x0, seg_f // GPIO pin 18
098     bl    gpioDirectionOut
099
100     adr    x0, seg_g // GPIO pin 12
101     bl    gpioDirectionOut
102
103     adr    x0, seg_dp // GPIO pin 26
104     bl    gpioDirectionOut
105
106     adr    x0, digit_0 // GPIO pin 18
107     bl    gpioDirectionOut
108
109     adr    x0, digit_1 // GPIO pin 23
110     bl    gpioDirectionOut
111
112     adr    x0, digit_2 // GPIO pin 24
113     bl    gpioDirectionOut
114
115     adr    x0, digit_3 // GPIO pin 25
116     bl    gpioDirectionOut

```

The `gpiomux.asm` program takes the numbers from shared memory and writes them to the seven-segment display. Each of the eight LED segments is connected to a GPIO pin designated as output. The four-digit driver pins (`digit_0`, `digit_1`, `digit_2`, and `digit_3`) have been designated as outputs, as well.

In `gpiomux.asm`, the `initialize` function (line 59) gets the shared memory region, performs the virtual GPIO mapping, and sets the GPIO pin directions as desired.

The `hex_numbers` lookup table (`gpiomux.asm`, lines 323-339) describes what segments should be on (0) or off (1) to form

the correct number on the seven-segment display. I know that sounds backward, but on a common-anode, seven-segment display, the anode of every LED in the device is connected through its driver pin and a general-purpose NPN transistor to 3.3V. Writing a 1 to a GPIO `seg_n` pin means putting 3.3V on the pin. With

### Listing 1: `gpiomux.asm` (continued)

```

117
118  adr  x0, pin_22      // GPIO pin 22
119  bl   gpioDirectionOut
120
121  mov  x0, xzr        // clear error flag
122
123  fin:
124  pop2 x29, x30      // pop fp & lr
125  ret
126
127 //=====
128 cleanup:           // cleanup has no parameters
129  push2 x29, x30    // push fp & lr
130
131  adr  x29, gpiobase // x29 => gpiobase
132  ldr  x29, [x29]
133
134  adr  x0, seg_a     // GPIO pin 20
135  bl   gpioDirectionIn
136
137  adr  x0, seg_b     // GPIO pin 21
138  bl   gpioDirectionIn
139
140  adr  x0, seg_c     // GPIO pin 19
141  bl   gpioDirectionIn
142
143  adr  x0, seg_d     // GPIO pin 13
144  bl   gpioDirectionIn
145
146  adr  x0, seg_e     // GPIO pin 06
147  bl   gpioDirectionIn
148
149  adr  x0, seg_f     // GPIO pin 16
150  bl   gpioDirectionIn
151
152  adr  x0, seg_g     // GPIO pin 12
153  bl   gpioDirectionIn
154
155  adr  x0, seg_dp    // GPIO pin 26
156  bl   gpioDirectionIn
157
158  adr  x0, digit_0   // GPIO pin 18
159  bl   gpioDirectionIn
160
161  adr  x0, digit_1   // GPIO pin 23
162  bl   gpioDirectionIn
163
164  adr  x0, digit_2   // GPIO pin 24
165  bl   gpioDirectionIn
166
167  adr  x0, digit_3   // GPIO pin 25
168  bl   gpioDirectionIn
169
170  adr  x0, pin_22    // GPIO pin 22
171  bl   gpioDirectionIn
172
173  mov  x0, x29
174  bl   mapClose      // unmap gpio
175
176  pop2 x29, x30     // pop fp & lr
177  ret
178
179 //=====
180 readWrite:         // readWrite has no parameters
181  push2 x29, x30    // push fp & lr
182  push x22
183
184  adr  x29, gpiobase
185  ldr  x29, [x29]    // x29 = gpiobase
186
187  adr  x22, digits  // get this from shared memory
188  ldr  x22, [x22]
189
190  continue:
191  ldrb w0, [x22, #THREE] // get lsd -- digit_0
192  adr  x1, hex_numbers // get lookup table hex_
                          numbers patterns
193  ldr  w1, [x1, x0, lsl #2] // point to correct num &
                          get pins to clr
194
195  adr  x0, digit_0   // write digit_0
196  bl   writeDigit
197
198  ldrb w0, [x22, #TWO] // get digit_1
199  adr  x1, hex_numbers // get lookup table hex_
                          numbers patterns
200  ldr  w1, [x1, x0, lsl #2] // point to correct num &
                          get pins to clr
201
202  adr  x0, digit_1   // write digit_1
203  bl   writeDigit
204
205  ldrb w0, [x22, #ONE] // get digit_2
206  adr  x1, hex_numbers // get lookup table hex_
                          numbers patterns
207  ldr  w1, [x1, x0, lsl #2] // point to correct num &
                          get pins to clr
208
209  adr  x0, digit_2   // write digit_2
210  bl   writeDigit
211

```

## Listing 1: gpiomux.asm (continued)

```

212  ldrb  w0, [x22, #ZERO]    // get digit_3
213  adr   x1, hex_numbers    // get lookup table hex_
                             numbers patterns
214  ldr   w1, [x1, x0, lsl #2] // point to correct num &
                             get pins to clr
215
216  adr   x0, digit_3        // write digit_3
217  bl    writeDigit
218
219
220  adr   x0, exitFlag        // read the exit flag byte
221  ldrb  w0, [x0]
222  cmp   w0, #ONE           // do we need to exit?
223  bne   continue          // branch if no
224
225  pop   x22
226  pop2  x29, x30           // pop fp & lr
227  ret
228
229 //=====
230 writeDigit: // x29 => gpiobase, w0 => active digit, x1 =
               print pattern
231  push2 x29, x30          // push fp & lr
232  push  x22
233
234  mov   x22, x0           // x0 => active digit to write
235
236  adr   x8, clrAllSeg
237  ldr   w8, [x8]          // w8 = bits we care about
238  str   w8, [x29, #gpset0] // write 1st pattern to
                             gpset0
239  str   w1, [x29, #gpclr0] // write 2nd pattern to
                             gpclr0
240
241  mov   x0, x22           // turn on digit
242  mov   x1, #ONE          // one pulse to base of npn
                             transistor
243  bl    gpioSetState
244
245  bl    sleep             // display digit for 2.5ms
246
247  mov   x0, x22           // turn off digit
248  mov   x1, xzr           // zero pulse to base of npn
                             transistor
249  bl    gpioSetState
250
251  pop   x22
252  pop2  x29, x30          // pop fp & lr
253  ret
254
255 //=====
256 sleep:
257  push2 x29, x30
258  ldr   x0, =timespecsec // sleep for 2.5ms
259  ldr   x1, =timespecsec
260  bl    nanosleep
261  pop2  x29, x30          // pop fp & lr
262  ret
263
264 //===== DATA SECTION =====
265 .section .data
266
267 exitFlag: .byte 0 // this will be 1 when we should exit
268 gpiobase: .dword 0 // memory mapped gpio address space
269 digits:   .dword 0 // 4 digits memory
270
271 //===== READ-ONLY DATA SECTION =====
272 .section .rodata
273
274 timespecsec: .dword 0 // 0
275 timespecnano: .dword 2500000 // 2.5ms
276
277 memdev:      .asciz "/dev/gpiomem"
278
279 // GPIO pin lookup table
280 seg_a:       .word 8 // pin20 - offset to select register
281              .word 0 // - bit offset in select reg
282              .word 20 // - bit offset in set & clr reg
283 seg_b:       .word 8 // pin21
284              .word 3
285              .word 21
286 seg_c:       .word 4 // pin19
287              .word 27
288              .word 19
289 seg_d:       .word 4 // pin13
290              .word 9
291              .word 13
292 seg_e:       .word 0 // pin06
293              .word 18
294              .word 6
295 seg_f:       .word 4 // pin16
296              .word 18
297              .word 16
298 seg_g:       .word 4 // pin12
299              .word 6
300              .word 12
301 seg_dp:      .word 8 // pin26
302              .word 18
303              .word 26
304 switch_0:   .word 4 // pin17
305              .word 21
306              .word 17
307 digit_0:    .word 4 // pin18 - shared memory
308              .word 24
309              .word 18
310 digit_1:    .word 8 // pin23
311              .word 9
312              .word 23
313 digit_2:    .word 8 // pin24
314              .word 12
315              .word 24
316 digit_3:    .word 8 // pin25
317              .word 15
318              .word 25
319 pin_22:     .word 8 // pin22
320              .word 6
321              .word 22
322
323 hex_numbers:

```

3.3V connected to the resistor and 3.3V on the LED, no current can flow, and the LED is unlit.

Only by writing a 0 to a GPIO `seg_n` pin which puts 0V (ground) on the resistor, do you get a difference of potential across the LED, which makes it light up. The resistor is there to limit the current that the GPIO pin will sink to less than 16mA. Most devices can sink more current than they can source, so the common-anode design is a common approach. Nowhere in the circuit do you connect to 5V. (Note: The Pi's GPIO pins are not 5V tolerant. Stick to 3.3V everywhere.)

The `readWrite` function (lines 180-227) is the workhorse of `gpiomux.asm`, cycling through the four shared memory locations and writing each in turn to its associated GPIO digit. It turns on one digit at a time and sets the appropriate segments to be on or off for each of the four digits. It has to turn the digits on and off fast enough to fool the eye that it is seeing the display as four separate digits. An oscilloscope display of the four-digit outputs is shown in Figure 4.

The `readWrite` function reads a digit in shared memory and determines from the `hex_numbers` lookup table which segments should be lit. It passes the arguments address of `digit_n` (`digit_0 ... digit_3`) and `hex_numbers[n]` to `writeDigit`, which writes the pattern to the GPIO hardware.

The `writeDigit` subroutine (lines 230-253; Figure 5) gets the pattern stored at memory location `clrAllSeg` and writes it into `gpset0` (0x28) from the GPIO base address (`gpioBase`). Next, it takes the pattern passed to it by `readWrite()` in register `w1` and writes it to `clrregoffset`. That lights the appropriate segments. It then turns on the digit (writes a 1 to the base of the associated NPN transistor driver). The address of the digit lookup table is passed to `writeDigit()` in register `x0`.

After writing to the GPIO hardware, `writeDigit` goes to sleep for 2.5ms and then turns off the digit (writes a 0 to the base of the associated NPN transistor). The `readWrite()` and `writeDigit()` functions produce the waveform shown in Figure 4.

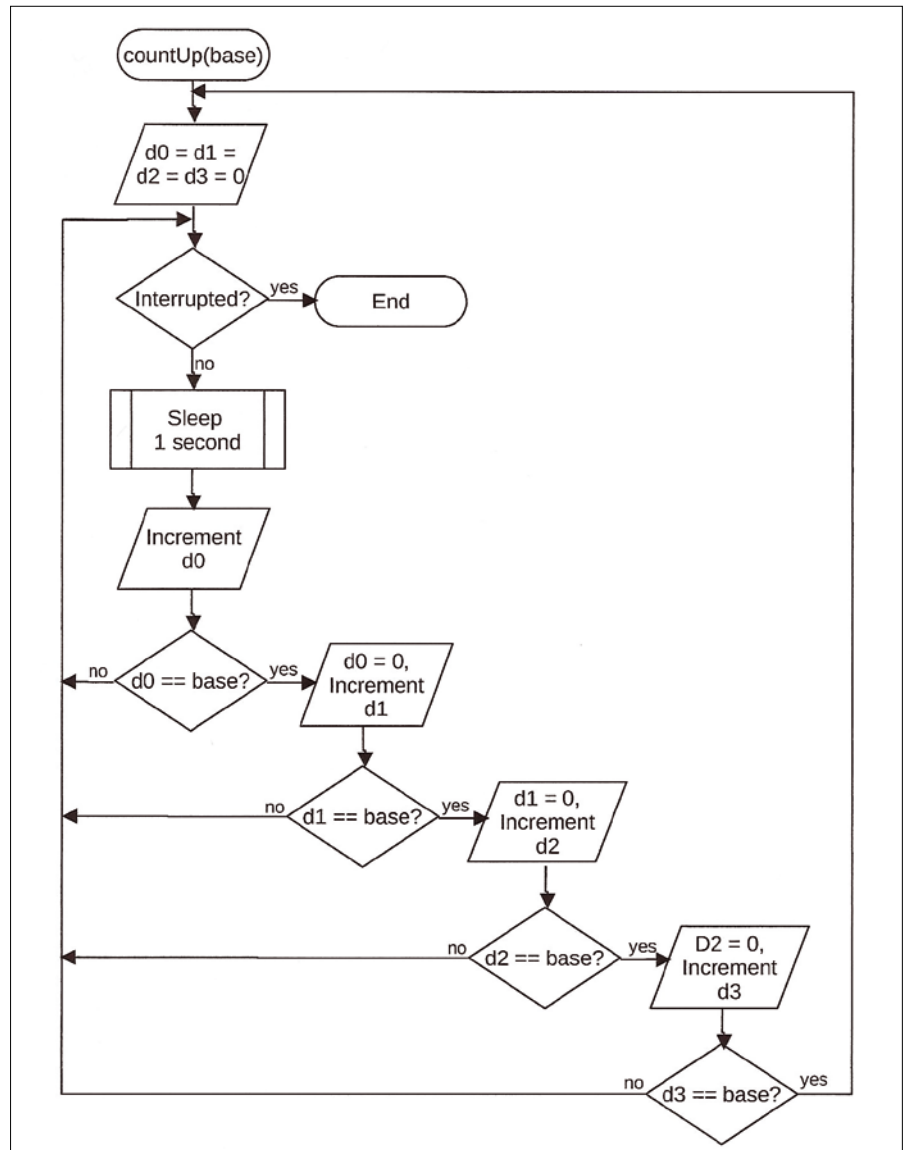
### Change Mode

So far, you have a device that counts over and over again, forever. To make it

**Listing 1: gpiomux.asm (continued)**

```

324 .word 0x00392040 // 0 - write this to gpclr0 to display 0
325 .word 0x00280000 // 1 - write this to gpclr0 to display 1
326 .word 0x00303040 // 2 - write this to gpclr0 to display 2
327 .word 0x00383000 // 3 - write this to gpclr0 to display 3
328 .word 0x00291000 // 4 - write this to gpclr0 to display 4
329 .word 0x00193000 // 5 - write this to gpclr0 to display 5
330 .word 0x00193949 // 6 - write this to gpclr0 to display 6
331 .word 0x00380000 // 7 - write this to gpclr0 to display 7
332 .word 0x00393040 // 8 - write this to gpclr0 to display 8
333 .word 0x00391000 // 9 - write this to gpclr0 to display 9
334 .word 0x00391040 // A - write this to gpclr0 to display A
335 .word 0x00093040 // b - write this to gpclr0 to display b
336 .word 0x00112040 // C - write this to gpclr0 to display C
337 .word 0x00283040 // d - write this to gpclr0 to display d
338 .word 0x00113040 // E - write this to gpclr0 to display E
339 .word 0x00111040 // F - write this to gpclr0 to display F
340
341 clrAllSeg:
342 .word 0x04393040 // write this to gpset0 to make leds dark
343
344 //=====
    
```



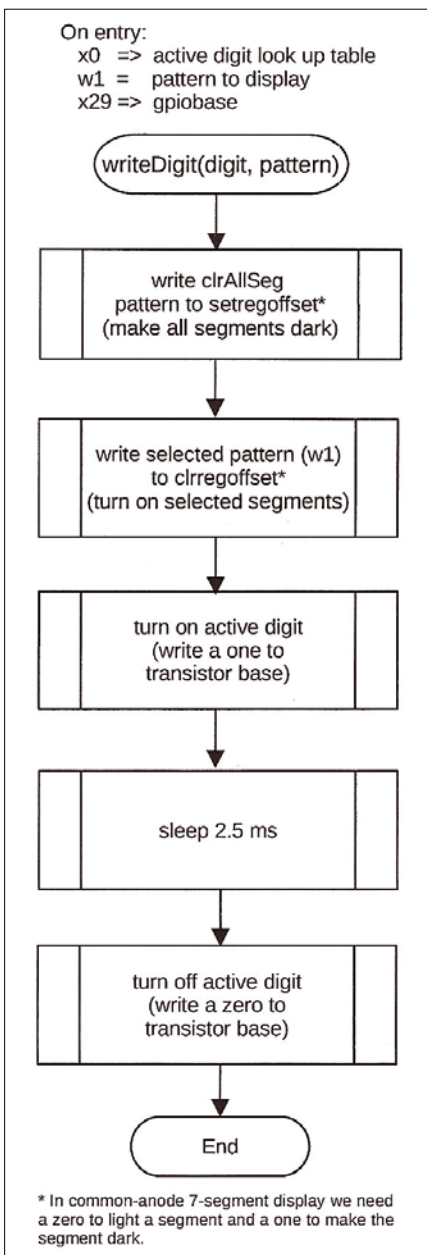
**Figure 3: Flowchart for the countUp(base) subroutine.**



**Figure 4:** The oscilloscope trace indicates the waveforms produced by `gpiomux`: yellow = GPIO pin 18, purple = pin 23, blue = pin 24, and green = pin 25.



**Figure 6:** Switch noise – a single press of the switch produces 300ms of noise before settling into the 0 state.



**Figure 5:** Flowchart for the `writeDigit(digit, pattern)` subroutine.

do something else – that is, to change the mode of the device – you need to interrupt the kernel and make it tell the current subroutine that it’s finished and should stop. That involves the GPIO pin 17 single pole, single throw, normally open switch and `lkm_gpio.c` to watch GPIO pin 17. Note that pin 17 is normally at 3.3V. When the switch is pressed, pin 17 goes low and then high again when the switch is released. Mechanical switches don’t produce a clean on-off state. They are actually quite noisy and can produce many spikes that can confuse the kernel (Figure 6).

A noisy switch can be dealt with in one of two ways: Ask the kernel to take care of it in software, or take care of it yourself in hardware. The software method (software debouncing) involves telling the kernel module that it should check again after a given number of microseconds and make sure that was really a falling edge pulse transition that it saw. Still, you will see a few extraneous triggers with software debouncing. Although the software solution is easy, it’s kind of hard on the kernel, because it’s already pretty busy and you are asking it to set timers and retest. The hardware method (hardware debouncing) involves adding hysteresis and a slight delay to the switch. The program `lkm_gpio.c` can be compiled both ways, so just uncomment line 16,

```
// #define __HW_DEBOUNCE__ //
```

if you want add the extra components necessary for hardware debouncing to the switch (Figure 7). The device works fine with either software or hardware debouncing, so you don’t need to deal with

adding the components shown in Figure 7 unless you want. Note that `gpio_set_debounce()` does not seem to be implemented on the Raspberry Pi.

### Basic ARM64 Assembly Tasks

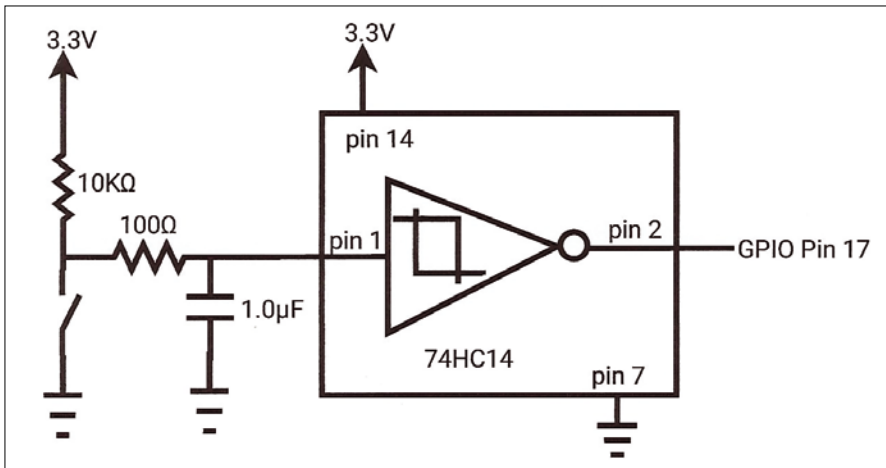
Next, I look at some basic GPIO tasks and see how they work in ARM64 assembly language. Remember that when you pass arguments to a function, they are placed in `x0, x1, ..., x7`. If you need more than eight arguments, the remaining arguments must be passed on the stack. That’s the convention for calling the kernel or a `glibc` function. Because you’re calling your own assembly language functions, you could pass the arguments any way you like, but for consistency, I’ll stick with the C calling convention.

In `libgpio.asm`, the `gpioDirectionIn` function (Listing 2, line 143) sets a GPIO pin as an input, for which you need the 3 bits that control its function. To make it easy, forgo multiplication and division and use a lookup table to find everything needed for any GPIO pin that will be used.

For example, say you want to make GPIO pin 22 an input (as in the `initialize()` subroutine in `gpiocount`). For all of the `libgpio` exported functions, `set_x29` points to the base register of GPIO. Any routine that deals with GPIO will need the GPIO base register address, so `x29` is a sort of global variable.

The first thing to do inside `gpioDirectionIn` is to load the contents of the first element (8) of `pin_22` in the GPIO lookup table (`gpiocount.asm`, line 300, on FTP site [1]) into `x2`:

```
// GPIO pin lookup table
pin_22: .word 8 // pin22 - LKM 2
```



**Figure 7:** Schematic diagram of a circuit that performs hardware switch debouncing.

```
will notify us of interrupts by 7
bringing this pin low
.word 6
.word 22
```

The first element of the lookup table is the offset of the function select register that controls GPIO pin 22. The function select registers start at the GPIO base register address, so `x29` plus `x2` is the address of `gpfseln`. The value in the lookup table is 8 bytes, so line 145 points to `gpfsel2`, which is 8 bytes away from the GPIO base register. The next value in the lookup table is the bit offset in `gpfsel2`, so 6 will be the value returned in `w3` by the statement in line 146.

Next, create a mask of 3 bits (0b111) in `w0` and perform a logical shift left of that mask by `w3` (which from the lookup table is 6). The `bic` instruction with `w1` as the destination register then clears the 3-bit mask, because 000 means input. The value read from `gpfsel2` is modified by writing three 0 bits into the register at bits 6, 7, and 8 before the modified data is written back into `gpfsel2`.

### Listing 2: libgpio.asm (excerpt)

```
...
142 // Set pin to input
143 gpioDirectionIn:      // x29 => gpio base, x0 => seg lookup tbl
144 ldr w2, [x0]          // w2 = offset to gpfseln - 1st val in tbl
145 ldr w1, [x29, x2]    // w1 = contents of gpfseln
146 ldr w3, [x0, #FOUR] // w3 = offset in gpfseln - 2nd val in tbl
147 mov w0, #CLEAR_MASK // w0 = mask to clear 3 bits (111)
148 lsl w0, w0, w3       // shift CLEAR_MASK into position
149 bic w1, w1, w0       // clr the 3 bits of gpfseln - 000 = input
150 str w1, [x29, x2]    // write it to gpfseln
151 ret
...
```

The `gpioDirectionOut` section starts the same way as `gpioDirectionIn` but then writes 001 (output) into `gpfsel2`.

The `hex_numbers` table (Listing 1, line 323) contains the values you want to write into `gpclr0` for each of the 16 number symbols. Remember, you write a 1 to `gpclr0` to set the output pin to 0, which makes the LED segment light. I recommend that you take a piece of graph paper and lay out the 32 bits of the `gpclr0` register from left to right, with 31 as the first column on the left and 0 as the last column on the right. Now mark the columns with the segment letters for the GPIO pins: An *a* would go in column 20, a *b* in column 21, a *c* in column 19, and so on.

As an example, take the symbol *l*, which has only segments *b* and *c* lit. Place a *1* in column 20, a *1* in column 19, and a *0* in every other column. Next, map each 4 bits into a number from 0000 (0) to 1111 (F), and you will have written the number symbol in hexadecimal. The value shown in `hex_numbers[1]` is 0x00280000. The symbol 7 lights segments *a*, *b*, and *c*. Go through the same

process as before, and you should get 0x00380000. That is the value in `hex_numbers[7]`. Every one of the 16 possible values is an element of `hex_numbers`.

### Running

See the boxout titled “Installing the Source Code.”

On the Ubuntu 20.10 operating system, `/dev/gpioem` is owned by `root:root` and marked as read/write for root only, so you should create a group named `gpio` and assign your user to that group. Also, change the ownership of `/dev/gpioem` to `root:gpio` and assign group read/write permissions. This change should occur on every bootstrap. As a quick alternative, you can use

```
sudo ./gpiocount
```

to run the application with root privileges. (See the “Setting Permissions at

### Installing the Source Code

To begin, open a terminal on the Raspberry Pi, make sure your operating system is up to date, and install the tools you’ll need:

```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt install make
$ sudo apt install gcc
$ cd
```

Now, create a `~/Development` folder and install `gpiosource.zip` in the new directory:

```
$ mkdir Development
$ cd Development
$ mkdir lib
$ unzip gpiosource.zip
```

Next, open the Bash initialization file,

```
$ cd
$ nano .bashrc
```

and add the following line at the end to set the shared library search path:

```
export LD_LIBRARY_PATH=~/.Development/lib
```

Now, save the file and close nano, close and reopen the terminal, and check your work:

```
$ echo $LD_LIBRARY_PATH
/home/<username>/Development/lib
```

Finally, make and run the code:

```
$ cd ~/Development/1km_gpio
$ make
$ cd ~/Development/gpiomux
$ make
$ ./gpiocount
```

## Setting Permissions at Bootup

1. Create a new group and add yourself to it:

```
sudo groupadd gpio
sudo usermod -a -G gpio <your_user_name>
```

2. Create file `/usr/sbin/gpiopermission.sh`:

```
sudo nano /usr/sbin/gpiopermission.sh

#!/bin/bash
/usr/bin/chown root:gpio /dev/gpiomem && /usr/bin/chmod g+rw /dev/gpiomem
exit 0
```

3. Make `gpiopermission.sh` executable:

```
sudo chmod +x /usr/sbin/gpiopermission.sh
```

4. Create file `/etc/systemd/system/gpiopermission.service`:

```
[Unit]
Description=Grants rw permission to /dev/gpiomem

After=network.target

[Service]
Type=simple
ExecStart=/bin/bash /usr/sbin/gpiopermission.sh
TimeoutStartSec=0
[Install]
WantedBy=multi-user.target
```

5. Reload services, generate dependencies, and enable the new service:

```
systemctl daemon-reload
systemctl enable gpiopermission.service
```

6. Confirm that all went well:

```
systemctl -all | grep gpiopermission.service
```

7. Reboot:

```
sudo reboot
```

8. Verify that `dev/gpiomem` belongs to `root:gpio` and that group `gpio` has read/write permissions:

```
ls -l dev/gpiomem
crw-rw---- 1 root gpio 240, 0 Jan  6 15:40 /dev/gpiomem
```

Bootup” box to see how to set the permissions automatically every time you boot up the Raspberry Pi).

Now look at the operation and installation of the loadable kernel module `lkm_gpio.c` (Listing 3), which is pretty much self-explanatory. An `init()` function (line 104) configures the interrupt mechanism with `irq_config()` (line 44), and a `cleanup()` function (line 112) releases the interrupt mechanism.

The interrupt mechanism behaves differently depending on whether `__HW_DEBOUNCE__` is defined at the top of this program. The default behavior is to use software debugging, but you can change that by uncommenting line 16, if you decide you want to add the components necessary for hardware debugging. Note that the `irq_handler` function, which is called when you depress `switch_0`, sets pin 22 to 0.

In `gpiocount.asm`, before sleeping for 1s, `countUp()` and `countDown()` invoke `gpioReadPin()` on GPIO pin 22. Then, `countUp()` and `countDown()` check the Z (zero) flag and exit if `Z == 1`, which returns the action to `main()` (in `countmain.cpp`) or `main` (in `gpiocount.asm`), launching its next subroutine and changing the program’s mode.

When `gpiocount` finishes its initialization, it starts `gpiomux` in background mode with:

## Listing 3: `lkm_gpio.c` (excerpt)

```
01 //*****
02 // @file lkm_gpio.c
03 // @author John Schwartzman
04 // @date 04/11/2021
05 // @version 1.0
06 // @brief A loadable kernel module (LKM) that handles
07 // interrupts
08 // from user space.
09 //*****
10 #include <linux/init.h>
11 #include <linux/module.h>
12 #include <linux/kernel.h>
13 #include <linux/interrupt.h>
14 #include <linux/gpio.h>
15
16 // #define __HW_DEBOUNCE__ // uncomment this line for
17 // __HW_DEBOUNCE__
18
19 static unsigned int irq_number;
20 static unsigned int gpio_button = 17;
21 // switch connects to Pin 17
22 static unsigned int pin_to_control = 22;
23 // Pin 22 - tell user about int.
24
25
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42 //*****
21 //*****
22 // irq handler - fired on interrupt
23
24 static irqreturn_t irq_handler(int irq, void* dev_id,
25 struct pt_regs* regs)
26 {
27 unsigned long flags;
28
29 // disable this hardware interrupt
30 local_irq_save(flags);
31
32 // Turn on GPIO pin 22).
33 // That's how the user space program determines that it
34 // has been interrupted. It reads pin 22
35 // It reads pin 22 and if it gets a 0 then it knows it
36 // should finish.
37 gpio_set_value(pin_to_control, 0);
38
39 // restore this hardware interrupt
40 local_irq_restore(flags);
41 return IRQ_HANDLED;
42 }
43
44 //*****
45
46
47
48
49
50
51
52
53
54
55
56
57
58
59
60
61
62
63
64
65
66
67
68
69
70
71
72
73
74
75
76
77
78
79
80
81
82
83
84
85
86
87
88
89
90
91
92
93
94
95
96
97
98
99
100
101
102
103
104
105
106
107
108
109
110
111
112
113
114
115
116
117
118
119
120
121
122
123
124
125
126
127
128
129
130
131
132
133
134
135
136
137
138
139
140
141
142
143
144
145
146
147
148
149
150
151
152
153
154
155
156
157
158
159
160
161
162
163
164
165
166
167
168
169
170
171
172
173
174
175
176
177
178
179
180
181
182
183
184
185
186
187
188
189
190
191
192
193
194
195
196
197
198
199
200
201
202
203
204
205
206
207
208
209
210
211
212
213
214
215
216
217
218
219
220
221
222
223
224
225
226
227
228
229
230
231
232
233
234
235
236
237
238
239
240
241
242
243
244
245
246
247
248
249
250
251
252
253
254
255
256
257
258
259
260
261
262
263
264
265
266
267
268
269
270
271
272
273
274
275
276
277
278
279
280
281
282
283
284
285
286
287
288
289
290
291
292
293
294
295
296
297
298
299
300
301
302
303
304
305
306
307
308
309
310
311
312
313
314
315
316
317
318
319
320
321
322
323
324
325
326
327
328
329
330
331
332
333
334
335
336
337
338
339
340
341
342
343
344
345
346
347
348
349
350
351
352
353
354
355
356
357
358
359
360
361
362
363
364
365
366
367
368
369
370
371
372
373
374
375
376
377
378
379
380
381
382
383
384
385
386
387
388
389
390
391
392
393
394
395
396
397
398
399
400
401
402
403
404
405
406
407
408
409
410
411
412
413
414
415
416
417
418
419
420
421
422
423
424
425
426
427
428
429
430
431
432
433
434
435
436
437
438
439
440
441
442
443
444
445
446
447
448
449
450
451
452
453
454
455
456
457
458
459
460
461
462
463
464
465
466
467
468
469
470
471
472
473
474
475
476
477
478
479
480
481
482
483
484
485
486
487
488
489
490
491
492
493
494
495
496
497
498
499
500
501
502
503
504
505
506
507
508
509
510
511
512
513
514
515
516
517
518
519
520
521
522
523
524
525
526
527
528
529
530
531
532
533
534
535
536
537
538
539
540
541
542
543
544
545
546
547
548
549
550
551
552
553
554
555
556
557
558
559
560
561
562
563
564
565
566
567
568
569
570
571
572
573
574
575
576
577
578
579
580
581
582
583
584
585
586
587
588
589
590
591
592
593
594
595
596
597
598
599
600
601
602
603
604
605
606
607
608
609
610
611
612
613
614
615
616
617
618
619
620
621
622
623
624
625
626
627
628
629
630
631
632
633
634
635
636
637
638
639
640
641
642
643
644
645
646
647
648
649
650
651
652
653
654
655
656
657
658
659
660
661
662
663
664
665
666
667
668
669
670
671
672
673
674
675
676
677
678
679
680
681
682
683
684
685
686
687
688
689
690
691
692
693
694
695
696
697
698
699
700
701
702
703
704
705
706
707
708
709
710
711
712
713
714
715
716
717
718
719
720
721
722
723
724
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
744
745
746
747
748
749
750
751
752
753
754
755
756
757
758
759
760
761
762
763
764
765
766
767
768
769
770
771
772
773
774
775
776
777
778
779
780
781
782
783
784
785
786
787
788
789
790
791
792
793
794
795
796
797
798
799
800
801
802
803
804
805
806
807
808
809
810
811
812
813
814
815
816
817
818
819
820
821
822
823
824
825
826
827
828
829
830
831
832
833
834
835
836
837
838
839
840
841
842
843
844
845
846
847
848
849
850
851
852
853
854
855
856
857
858
859
860
861
862
863
864
865
866
867
868
869
870
871
872
873
874
875
876
877
878
879
880
881
882
883
884
885
886
887
888
889
890
891
892
893
894
895
896
897
898
899
900
901
902
903
904
905
906
907
908
909
910
911
912
913
914
915
916
917
918
919
920
921
922
923
924
925
926
927
928
929
930
931
932
933
934
935
936
937
938
939
940
941
942
943
944
945
946
947
948
949
950
951
952
953
954
955
956
957
958
959
960
961
962
963
964
965
966
967
968
969
970
971
972
973
974
975
976
977
978
979
980
981
982
983
984
985
986
987
988
989
990
991
992
993
994
995
996
997
998
999
1000
```



```
system("./gpiomux &")
```

When `gpiocount` finishes its assigned work or if it detects a Ctrl + C, it finishes counting, cleans up, and stops `gpiomux` with the hang-up signal:

```
signal("killall -HUP gpiomux")
```

To build `lkm_gpio`, type `make` at the command prompt in the `lkm_gpio` directory. The makefile auto-loads `lkm_gpio`. At the command line, type

```
lsmod | grep lkm_gpio
```

to see whether `lkm_gpio` has been installed. You might have to reboot Linux to get it running the first time.

Next, go to the `gpiomux` directory and type `make` at the command prompt to build `gpiolib`, `gpiocount`, and `gpiomux`. The shared object file `gpiolib` is used by both `gpiocount` and `gpiomux`. You can type `make` to build the standard version, or

```
make OPT=__MAIN__
```

to build without `countmain.cpp` and `muxmain.cpp`. You can also type

```
make debug
```

(with or without `OPT=__MAIN__`), so you can execute the programs in the GNU project debugger, `gdb`.

To start the action, launch the `gpiocount` executable (with `./gpiocount` at the command line), which will start up the `gpiomux` executable and stop it when `gpiocount` finishes.

Clicking `switch_0` changes the mode from `countUp(2)` to `countUp(7)` to `countUp(10)` to `countUp(16)` to `countDown(16)` to `countDown(10)` to `countDown(7)` to `countDown(2)`. On the last click of `switch_1`, `gpiocount` cleans up and terminates because it has no more work to do and signals `gpiomux` with a SIGHUP to tell it to clean up and terminate. Because `gpiocount` must clean up and tell `gpiomux` to clean up, it also

traps and handles a Ctrl + C from the keyboard.

## Components

Any general-purpose NPN transistor (2N4400, 2N2222A, BC548, etc.) will work. Likewise, any four-digit, common anode, seven-segment display will be fine. Because this project uses AArch64 assembly language, your Raspberry Pi must be running a 64-bit distro like Ubuntu 20.10.

Have fun! ■■■

## Info

[1] Code for this article:

<ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/247/>

## Author

**John Schwartzman** has enjoyed an active career as an engineer, college professor, and consultant to business and government. He can be reached at [john@fortesystem.com](mailto:john@fortesystem.com).



### Listing 3: lkm\_gpio.c (excerpt continued)

```

43 // configure interrupts
44 static void irq_config(void)
45 {
46 #ifndef __HW_DEBOUNCE__
47     int retval;
48 #endif
49
50     gpio_request(gpio_button, "gpio_button");
51     gpio_direction_input(gpio_button);
52     gpio_direction_output(pin_to_control, 1);
53
54     // If you don't debounce the switch with hardware,
55     // uncomment gpio_set_debounce(), and the kernel
56     // module will do it for you in software.
57     irq_number = gpio_to_irq(gpio_button);
58     printk(KERN_INFO "lkm_gpio: gpio_button is currently
59     %d\n", gpio_get_value(gpio_button));
60     printk(KERN_INFO "lkm_gpio: Mapped irq number to %d\n",
61     irq_number);
62
63 #ifndef __HW_DEBOUNCE__
64     // Use IRQF_TRIGGER_FALLING if the switch is
65     // connected directly to gpio_button.
66     // If the switch is debounced using an Schmitt
67     // trigger inverter chip then use IRQF_TRIGGER_RISING.
68     retval = gpio_set_debounce(gpio_button, 600);
69     // recheck after 600µs
70     if (retval != 0)
71     {
72         printk(KERN_WARNING "lkm_gpio: gpio_set_debounce()
73         returned %d\n", retval);
74     }
75
76     if (request_irq(irq_number,
77     (irq_handler_t)irq_handler,
78     IRQF_TRIGGER_FALLING,
79     // falling if software debouncing
80     "RPI_gpio_handler",
81     // identify owner in /proc/interrupts
82     NULL))
83     {
84         printk(KERN_INFO "lkm_gpio: Irq request failure");
85     }
86     #else // we're doing our debouncing in hardware
87     // Use IRQF_TRIGGER_FALLING if the switch is
88     // connected directly to gpio_button.
89     // If the switch is debounced using an Schmitt trigger
90     // inverter chip then use IRQF_TRIGGER_RISING.
91     if (request_irq(irq_number,
92     (irq_handler_t)irq_handler,
93     IRQF_TRIGGER_RISING, // rising if debounced
94     switch
95     "RPI_gpio_handler", // identify owner in /proc/
96     interrupts
97     NULL))
98     {
99         printk(KERN_INFO "lkm_gpio: Irq request failure");
100     }
101     #endif
102 }
103 }
104 ...

```



# MakerSpace

Raspberry Pi OS now comes with PulseAudio and a graphical printer manager

## Big Presents

The Raspberry Pi Foundation regularly adds new features to the official operating system, Raspberry Pi OS. The December 2020 update added the PulseAudio sound server and a print manager. *By Christoph Langner*

**I**n practice, Raspberry Pi OS fulfills many of the functions supplied by other operating systems and the popular Linux distributions; however, a few deficiencies exist. In particular, the configuration of printers and sound devices has never been trivial – until now. In the latest update [1], Raspberry Pi OS was extended to include PulseAudio and a better print manager.

Bluetooth speakers can now be integrated more easily, and sounds from multiple sources can be output simultaneously.

Linux is a historically evolved system that changes continuously. New versions fix bugs or add features but do not bring about fundamental change. However, every few years, more profound innovations occur, such as the replacement of the X.Org display

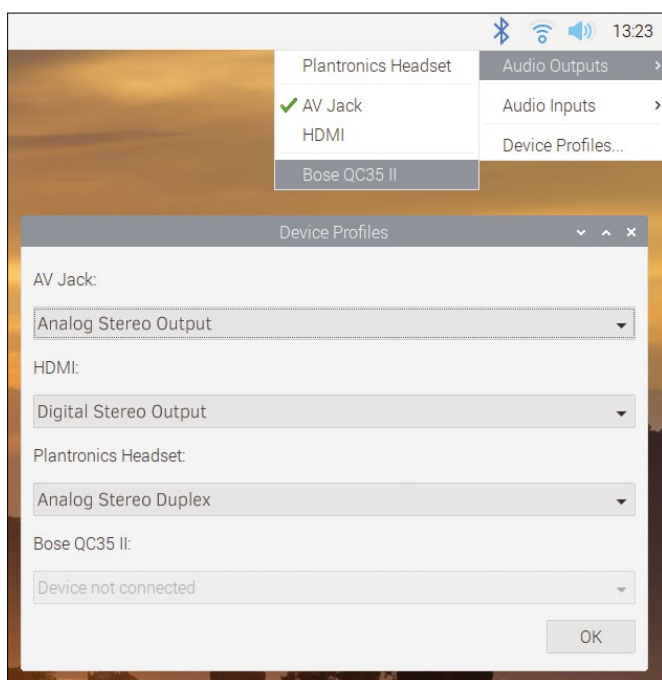
manager by Wayland, and before that the switch from the SysVinit to systemd, and before that the integration of the PulseAudio sound server.

### PulseAudio

The advanced Linux sound architecture (ALSA) provides drivers in the kernel to integrate sound cards and takes care of configuration. However, ALSA reaches its limits on modern systems; for example, only one application can output sound at any given time. For example, while a YouTube video plays in the browser, a messenger program cannot play a notification sound. Also, Bluetooth devices can only be connected in a roundabout way.

As an intermediate layer, many distributions now use PulseAudio [2], and the latest version of Raspberry Pi OS is one of them. Right-clicking on the volume slider icon in the panel brings up a dialog that lets you select the current output and input devices (Figure 1). For example, if you switch from *AV Jack* to *HDMI*, the Raspberry Pi will no longer output sounds through the jack plug, but through the connected TV, and switching between audio devices no longer interrupts playback.

The *Device Profiles* menu item can be used specifically to disable individual audio devices or, especially for USB or



**Figure 1:** PulseAudio not only plays sound from multiple applications, but also facilitates the integration of Bluetooth speakers and headphones.

Bluetooth audio devices (see the “Coupling” box), configure a device and select a different profile. For a Bluetooth headset, for example, you can choose between headset profile (HSP) and advanced audio distribution profile (A2DP). HSP lets you use the microphone and the headphones at the same time, but with far inferior audio quality compared with A2DP, whereas A2DP completely disables the microphone. PulseAudio is configured in such a way that the system automatically selects HSP as soon as you select a Bluetooth headset as the input device. If you use the headset as an output device, PulseAudio switches to A2DP.

## Printer

Until now, Raspberry Pi OS also lacked a user-friendly tool for setting up a printer. Most Linux distributions include the CUPS [3] printing service for this purpose, as well as configuration tools provided by the desktop environment. Alternatively, CUPS – and therefore a printer connected to the system – can be set up from the service’s web front end. CUPS has always been in the Raspberry Pi OS package sources, but now the service forms a fixed part of the system together with a graphical front end.

To configure a printer, go to *Settings* | *Print Settings*. In this dialog, you can set up new printers, delete printers that are no longer in use, set a device as the default printer, or take a look at the printer maintenance queue. When adding a printer, CUPS tries to find the printer (Figure 3) and is usually very successful

## Coupling

The revamped Raspberry Pi OS now connects far more easily with Bluetooth audio devices. In principle, you just have to left-click on the Bluetooth icon in the panel and select *Add Device* to call the wizard for adding new devices. As soon as you put the device into pairing mode, it appears in the dialog. However, before you pair it by clicking *Pair*, you do need to wait until the system has identified the headset or speaker as an audio device. This is signaled by the change from the yellow question mark icon to the speaker icon (Figure 2). If you close the dialog too quickly, the system will connect to it, but the device will not show up in the PulseAudio Manager.

with network-capable printers. CUPS should also recognize most USB printers reliably. However, multifunction devices with integrated scanners often require drivers from the manufacturer – if the manufacturer supports Linux at all.

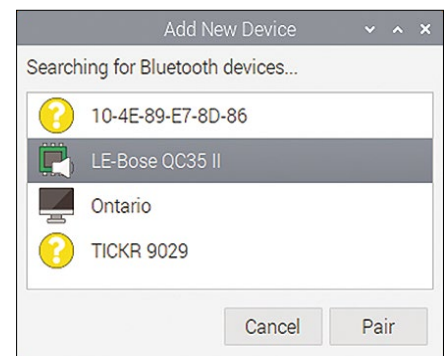
With an older laser printer from Samsung, mounting worked without any problems. A list of all printers and multifunction devices supported by CUPS can be found on [Openprinting.org](http://Openprinting.org) [4]. If a function or a certain setting is missing in the configuration interface, you can reach the CUPS service’s web front end by calling `http://localhost:631` in the browser. For administrative activities, the web front end requires you to enter access credentials. The username (usually *pi*) and password matches the data you use when logging in to the system.

## Chromium

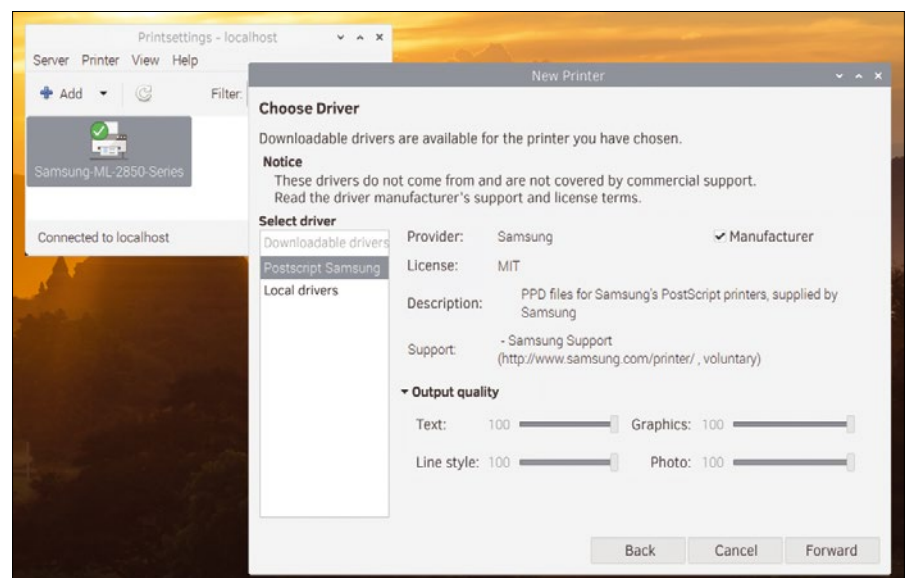
Chromium browser updates always mean a bit more work on Raspberry Pi OS than just pushing the latest build of the software into the package sources. The developers have to make sure, especially with hardware acceleration, that the web browser still harmonizes with the graphics hardware of the Raspberry Pi. Without proper hardware support, many modern web services would be restricted in how they work, especially video platforms or videoconferencing solutions. At the time of writing, the version of the preinstalled Chromium browser sits at 86.0.4240.197.

The computing power of a Raspberry Pi 4 is not yet sufficient to play YouTube videos in Full HD in the browser without dropping some frames. Only the C64-style keyboard computer, the Raspberry Pi 400, offers enough power. It plays most YouTube videos so efficiently at 1080p resolution (with its processor clocked at 1.8GHz – vs. 1.5GHz on the Raspberry Pi 4) that hardly any frames are lost. Videoconferencing solutions such as Jitsi and Google Meet also benefit from the latest optimizations. In this test, both services worked without problems, but sharing the desktop or individual application windows only worked under Jitsi (Figure 4).

Some maintenance has also been done to the Raspberry Pi configuration tool. On a device with a single status LED



**Figure 2:** When adding a Bluetooth speaker, you have to wait until the system identifies the new device (speaker icon in front of the name).



**Figure 3:** Printers used to be configured by the less intuitive web front end of the CUPS print service. The December update now includes the graphical printer manager shown here.

(e.g., the Pi Zero or the new Raspberry Pi 400), you can now set whether the LED flickers to indicate read/write access to the memory card or is permanently lit to indicate the operating status. If the Raspberry Pi is in a case with a fan connected to the GPIO, the system now

supports simple fan control by entering the GPIO pin in the *Performance* tab, as well as the temperature above which the fan should switch on (Figure 5).

## Accessibility

For users who rely entirely on hearing, the Raspberry Pi Foundation in early 2020 implemented the Orca screen reader, which is being developed as part of the Gnome desktop environment [5]. The program reads aloud the content of application interfaces, web pages, and documents, so the graphical desktop is no longer an obstacle. In collaboration with the Orca project, bugs were removed from the program,

and tools (e.g., *Raspberry Pi Configuration* and *Appearance Settings* under *Prefereces*) were optimized for Orca. Additionally, switching to PulseAudio makes it easier to redirect the audio output to Bluetooth headsets.

However, if users want to set up the system themselves, they have faced a problem up to now: Orca is not included in the standard system and has to be installed retroactively, which is more or less impossible for a user with visual impairment and without a screen reader. In the current version of Raspberry Pi OS, the installation can be triggered after starting the system by pressing Ctrl + Alt + Space. The assistant announces the procedure by voice output, installs Orca, and then reloads the graphical interface with the voice assistant.

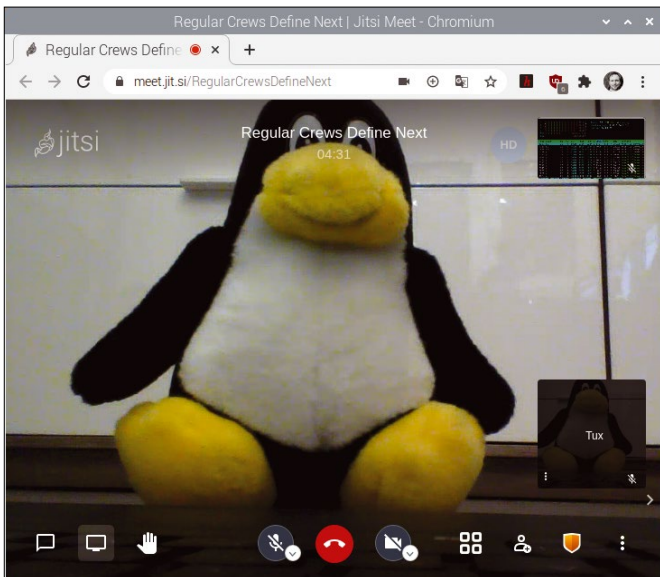
Pressing Ctrl + Caps Lock + Space opens the program settings. For example, the reading speed can be adjusted, or the Orca modifier key (Caps Lock in the default configuration) can be changed. Together with magnification, which you can enable (after installing the magnifier program through *Recommended Software*) by clicking on the magnifier icon in the panel, the Raspberry Pi OS system is accessible to users who rely entirely on hearing (Figure 6).

## Conclusions

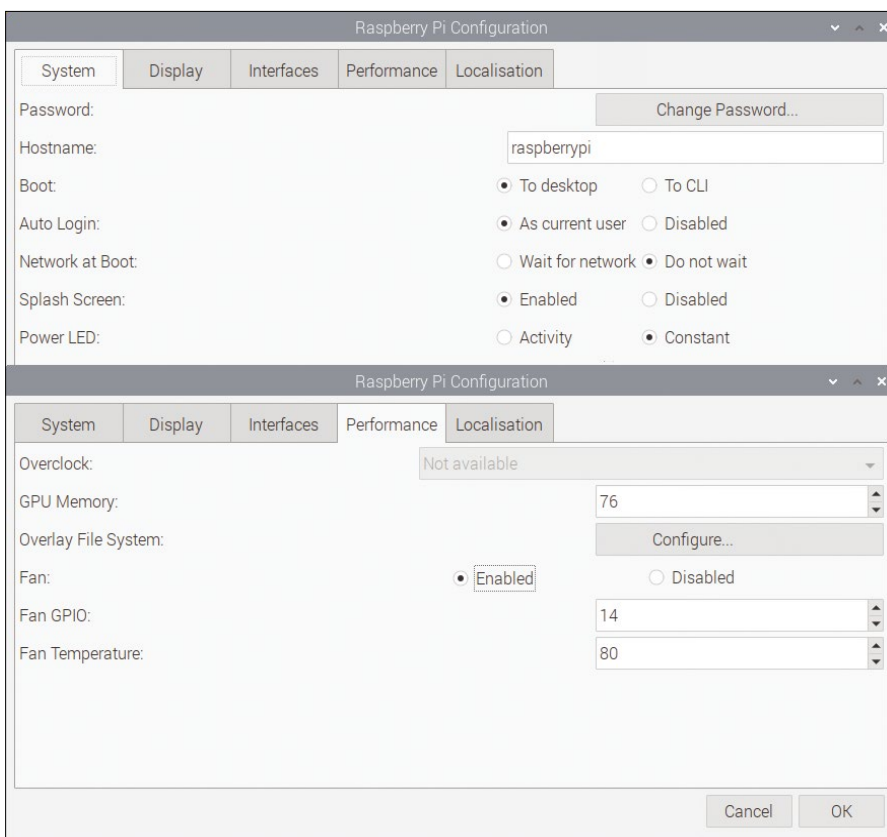
Raspberry Pi Foundation developers are working to add more and more refinements. The integration of PulseAudio and a CUPS front end makes the use of Bluetooth speakers or headphones and configuring printers far easier. Users with special needs are no longer left behind: The system can now be set up and configured independently by users who rely entirely on hearing, without requiring help for the first steps. ■■■

## Info

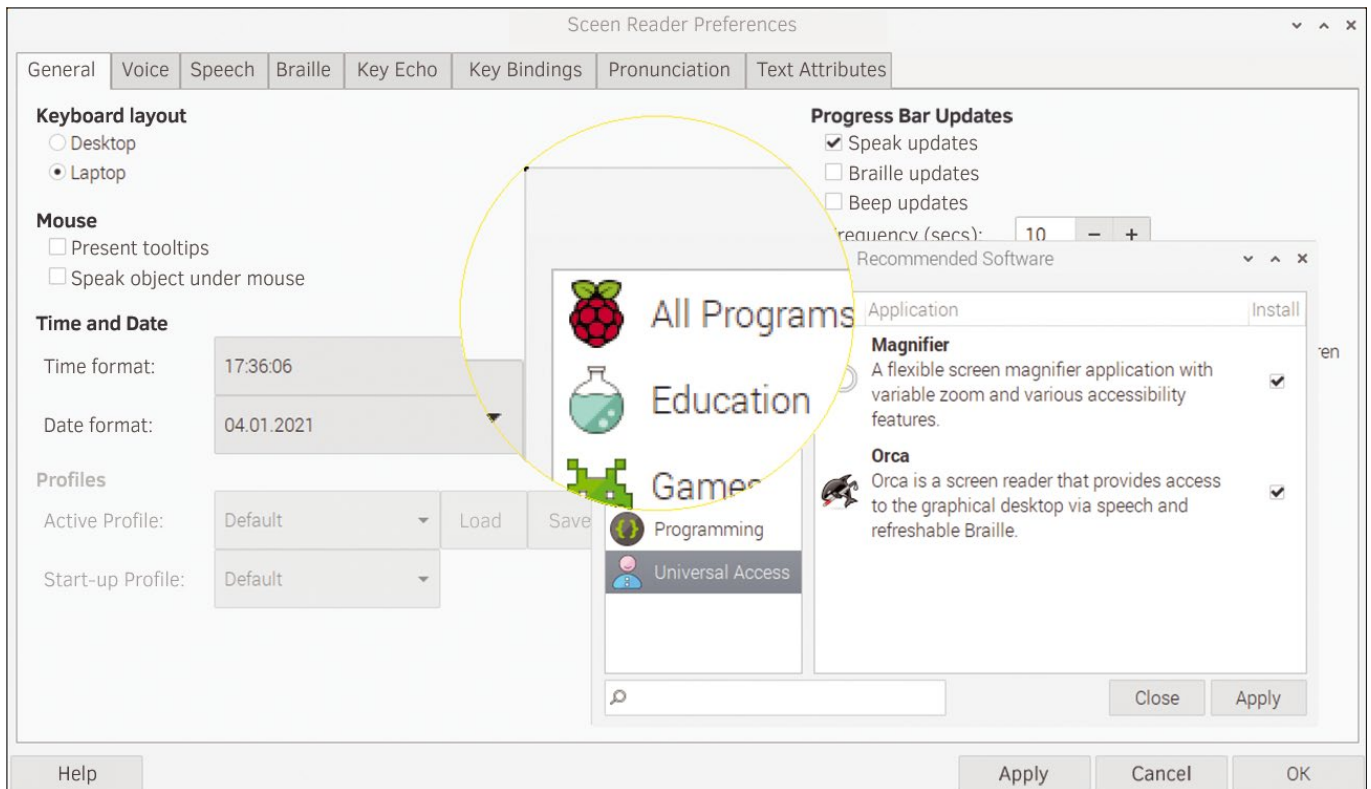
- [1] Pi OS 2020-12-02: <https://www.raspberrypi.org/blog/new-raspberrypi-os-release-december-2020>
- [2] PulseAudio: <https://www.freedesktop.org/wiki/Software/PulseAudio/>
- [3] CUPS: <https://www.cups.org>
- [4] Printers supported by CUPS: <https://www.openprinting.org/printers>
- [5] Orca: <https://help.gnome.org/users/orca/stable>




**Figure 4:** Videoconferencing solutions such as Jitsi and Google Meet benefit from optimizations to the graphics driver.



**Figure 5:** In the Raspberry Pi configuration tool, the Power LED of the Raspberry Pi 400 can now be configured, and a temperature threshold for fans connected over GPIO can be defined.



**Figure 6:** Users who rely entirely on hearing can now install the Orca screen reader via a shortcut. Together with the screen magnifier, the system is therefore a good choice for users with special needs.



**Our ADMIN Online website offers additional news and technical articles you won't see in our print magazines. Subscribe today to our free ADMIN Update newsletter and receive:**

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

# Too Swamped to Surf?

**ADMIN**  
Network & Security

**ADMIN Update – Hottest Links**

- Building Safe Containers
- Linux Foundation Creates New Code Signing Solution
- The Linux-Next Kernel Finally Gets a Bit "Rusty"
- What's New in PHP 8?
- Monitoring Events with Apache Kafka

**Highlights**

**Building Safe Containers**  
The basic container images on which you base your work can often be out of date. We show you how to solve this problem and create significantly leaner containers. (more)

**Linux Foundation Creates New Code Signing Solution**  
The Linux Foundation has created a signature to serve as the Let's Encrypt of code signing. (more)

**The Linux-Next Kernel Finally Gets a Bit "Rusty"**  
Rust is finally coming to the Linux kernel, by way of Linux-Next, which has many developers cheering. (more)

**What's New in PHP 8?**  
After about two years of development, version 8 of the popular PHP scripting language has been released. It comes with a number of innovations and ditches some obsolete features (more)

**Monitoring Events with Apache Kafka**  
Apache Kafka reads and writes events virtually in real time, and you can extend it to take on a wide range of roles in today's world of big data and event streaming. (more)

**Most Read Articles**

**Real-Time Log Inspection with Teler**  
Teler is an intrusion detection and threat...

**Further Reading**

- Linux Exploit for Spectre flaw Discovered
- Keycloak Identity and Access Manager
- Yet Another Bonnet is Targeting Linux
- Calendar Sync without Google
- Three Major Threats to Linux Discovered

**DrupalCon**  
Grow your network and collaborate with others in the Drupal open source community.  
**REGISTER NOW**

**Linux Shell Handbook 2021 Edition**

**301 BEST BASH COMMANDS**  
**LINUX SHELL HANDBOOK**  
**SUPERCHARGE**  
FOUR LINUX SKILLS  
Master at Your Fingertips

**ADMIN**  
Network & Security

# MakerSpace

## Create GUI dialogs in one line of code Finding a Little Zen

The Zenity command-line utility lets you create simple dialog boxes with your own data or with the output of utilities and applications. *By Pete Metcalfe*

**Z**enity is a command-line GUI creator that has been around since 2012 and is pre-installed on most versions of Linux, including Raspberry Pis. Zenity isn't designed to be a high-level GUI development tool, but if you just need some basic scripting with dialogs, then Zenity might be a perfect fit.

I was amazed that in one line of Bash code I was able to show:

- stats in a message dialog
  - a web page in a dialog
  - CSV data or SQL queries in a list dialog
- After looking at slightly more complex applications, I found I was able to create:
- a progress bar with dynamic values (about seven lines)
  - a form to insert user data into an SQL database (about eight lines)
  - a four-button dialog to control a Rasp PI Rover (~20 lines)

In this article, I introduce Zenity with some examples.

### Getting Started

The Zenity [1] command-line utility is supported on Linux, macOS, and Windows. Zenity can display calendar, color selector, file selector, form, list, message, notification, progress, scale, and text dialogs.

The Zenity message dialog can be used like a Bash *echo* statement; for example, to show an information message, enter:

```
zenity --info --text="Some text" Z
--title="My Title"
```

Information from command-line tools and utilities can be passed to Zenity. For example, the instantaneous CPU idle time from the `top` utility can be parsed and passed to a Zenity dialog (Figure 1):

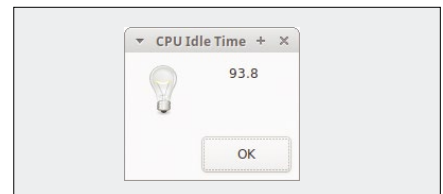
```
zenity --info Z
--text=$(top -n 1 | Z
grep %Cpu | Z
awk '{print $8}') Z
--title="CPU Idle Time"
```

Text font and size can be modified in message dialogs in Pango markup language syntax [2]. Pango is similar to HTML, and the `<span> ... </span>` set of tags is typically used to encode font and color definitions (Figure 2):

```
zenity --warning --text=Z
'<span font="32" foreground="red"> Z
HIGH Temperature</span>' Z
--title="HDD Check"
```

For scripting applications that need to pass more information to users, text-info dialogs can pass text files and URL links (Figure 3):

```
zenity --text-info Z
--title="Background Reading" Z
--html Z
```



**Figure 1:** Message dialog with CPU idle time.



**Figure 2:** Message dialog with font changes.

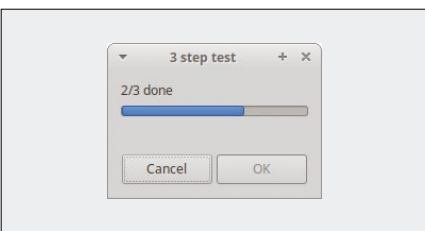
```
--url="https://developer.gnome.org" Z
--checkbox=Z
"I read it...and I'm good to go"
```

For dialogs with *OK* and *Cancel* buttons, Zenity returns a 0 to confirm and a 1 to cancel. Zenity will not process JavaScript on a web page.

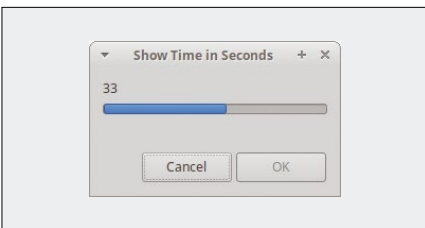
Simple dialogs like the info, warning and error dialogs will only have an *OK* button. All the other dialogs will have *Cancel* buttons, as well. The button text can be changed with the `--ok-label` and the `--cancel-label` options. More buttons can be added with the `--extra-button` option.



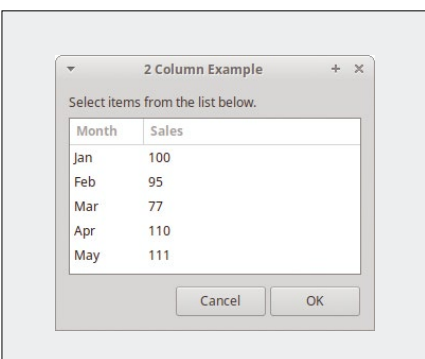
**Figure 3:** Web page within a dialog.



**Figure 4:** A three-step progress dialog.



**Figure 5:** Continuous updates in a progress dialog.



**Figure 6:** List dialog showing inline data.

### Dynamic Values

A Zenity progress dialog can show dynamic updates with scripts that use subshells. A subshell is configured with parentheses. Step executions within the subshell are paused by `sleep` statements.

The following code shows a three-step example of a subshell with a Zenity

### Listing 1: Scaled Progress Bar

```
#!/bin/bash
# show_seconds.sh - progress dialog to show seconds
echo "Press [CTRL+C] to stop..."
(
  while ;; do
    echo "# $(date +%S)"
    # Scale 0-60 to 0-100
    echo "$(date +%S)*100/60" | bc
    sleep 1
  done
) | zenity --progress --title="Show Time in Seconds"
```

progress dialog (Figure 4):

Once you have some Zenity and Bash basics down, you can start doing more advanced operations, such as:

```
(
  echo "33"; echo "# 1/3 done" ; sleep 5;
  echo "66"; echo "# 2/3 done" ; sleep 5;
  echo "100";echo "# Finished" ) | zenity --progress --title="3 step test"
```

```
awk -F "\*,\*" '{print $3 "\n" $1}' pidata.csv | zenity --list --column="field3" --column="field1"
```

This one-line example uses `Awk` to parse specific data (fields 1 and 3) in the CSV file.

### SQL Data in List Dialogs

SQL command-line utilities can output SQL queries to a Zenity list dialog. Like the earlier CSV examples, the SQL output

### Listing 2: Reformatting Data

```
$ cat pidata.csv
10:00,12,running
10:20,14,stopped
10:30,13,running
$ cat pidata.csv | tr ',' '\n'
10:00
12
running
10:20
14
stopped
10:30
13
running
```

When a step outputs a value, the progress bar updates. The text on the progress dialog is changed by outputting a text string that starts with a `#` character.

Listing 1 and Figure 5 show an example of current seconds with the progress bar rescaled from 0 to 60.

The Bash `while` statement lets you show dynamic values continuously until the `Cancel` button is pressed.

### CSV Data in List Dialogs

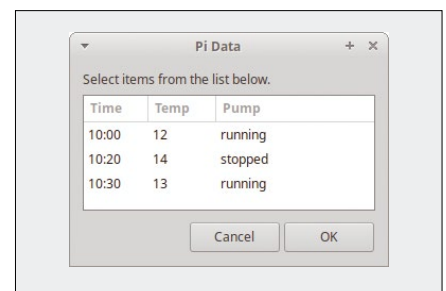
For simple text files and known datasets, the `list` dialog works quite well. The dialog expects the data to be sequential. The following code creates a two-column example (Figure 6) with inline data:

```
zenity --list --title="2 Column Example" --column="Month" --column="Sales"
Jan 100 Feb 95 Mar 77 Apr 110 May 111
```

To pass a data file (CSV or text) into Zenity, the text needs to be reformatted. The `tr` command can replace CSV field separators like commas with a newline (`\n`) character (Listing 2).

The output can then be passed to a Zenity list with column headings (Figure 7):

```
cat pidata.csv | tr ',' '\n' | zenity --list --title="Pi Data" --column="Time" --column="Temp" --column="Pump"
```



**Figure 7:** List dialog showing CSV data.

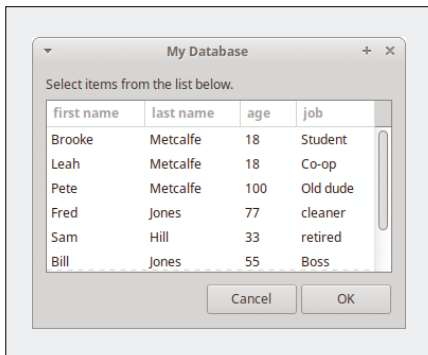
needs to be reformatted to a sequential list. The SQL output from the command-line tools will vary by database; for example, MySQL uses tabs between the fields, whereas SQLite uses a vertical bar (|).

For my testing, I used an SQLite3 database (`someuser.db`) with a table (`users`) of fields containing first and last names, age, and job. To output a `SELECT` query, I entered:

```
$ sqlite3 someuser.db "select fname,
lname,age,job from users"
Brooke|Metcalfe|18|Student
Leah|Metcalfe|18|Co-op
Pete|Metcalfe|100|Old dude
...
Willy|Coyote|99|Evil genius
```

The SQLite query output can be modified with the `tr` command and shown in a Zenity list dialog (Figure 8):

```
$ sqlite3 someuser.db "select fname,
lname,age,job from users"
| tr '|' '\n' | zenity --list
--title="My Database"
```



**Figure 8:** List dialog showing SQL query.

```
--column="first name"
--column="last name" --column=age
--column=job
```

The Zenity list dialog supports a number of useful options, such as radio buttons and checkboxes. The lists are editable, and the selected fields or rows can be used in further scripting.

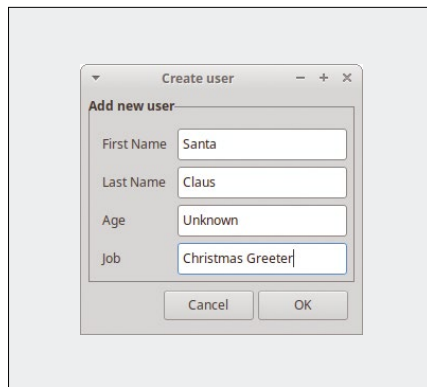
### Insert SQL Data in a Zenity Form

Zenity forms allows for the creation of basic data entry dialogs. In about eight lines of Bash code (Listing 3), I created a Zenity form (Figure 9) I can use to insert data into my SQLite `users` table.

The `OK` button will pass the user-entered data as a string, whereas the `Cancel` button will not pass any data. An if statement checks to see whether any data has been entered.

The SQL `INSERT` statement needs `VALUES` to be in the format

```
("value1", "value2", "value3", "value4")
```



**Figure 9:** SQL input form.

### Listing 3: SQL Input Form

```
01 #!/bin/bash
02 row=$(zenity --forms --title="Create user" --text="Add new user" --add-entry="First Name" --add-entry="Last Name"
--add-entry="Age" --add-entry="Job" --separator="',"")
03 if [[ -n $row ]] # Some data found
04 then
05 cmd="sqlite3 someuser.db \\"INSERT INTO users (Fname,Lname,Age,Job) VALUES ('$row')\""
06 eval $cmd
07 echo "Added data: '$row'"
08 fi
```

The formatting can be done by setting the Zenity `--separator` option to a comma, defined with single quotes within double quotes (line 2).

This example is quite basic, so the next step would be to add data validation.

### Final Comments

For simple dialogs, Zenity works amazingly well. I found that as the requirements started to get more complicated, a Python solution appeared to be cleaner and simpler. I was able to control a Raspberry Pi rover in about 20 lines of Bash and Zenity code, but it only took 15 lines of Python and Tkinter code.

There is a Python library (Zenity 2.0) that emulates Zenity, so if you're feeling comfortable with the Zenity dialogs and you don't need complex dialogs, this might be something to consider.

If you are looking for a more complete command-line GUI tool, try YAD [3]. ■■■

### Info

- [1] Zenity documentation: <https://help.gnome.org/users/zenity/3.32/index.html>
- [2] Pango markup language: <https://developer.gnome.org/pygtk/stable/pango-markup-language.html>
- [3] YAD: <https://sourceforge.net/projects/yad-dialog/>

### Author

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



**First there was Ubuntu**, and then the user base began to innovate, rolling out alternatives to highlight favorite desktops and tools. Some of the leading flavors are now supported by Canonical and the Ubuntu project, including Kubuntu, Lubuntu, Xubuntu, and Ubuntu Studio. But the user base is *still* innovating. This month you'll hear from a teacher who spun up his own version of Ubuntu Budgie to include a collection of education tools (called, you guessed it, EdUBudgie).

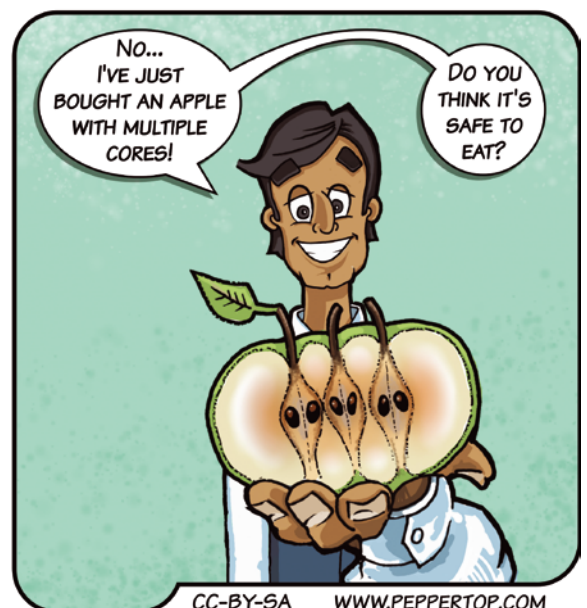
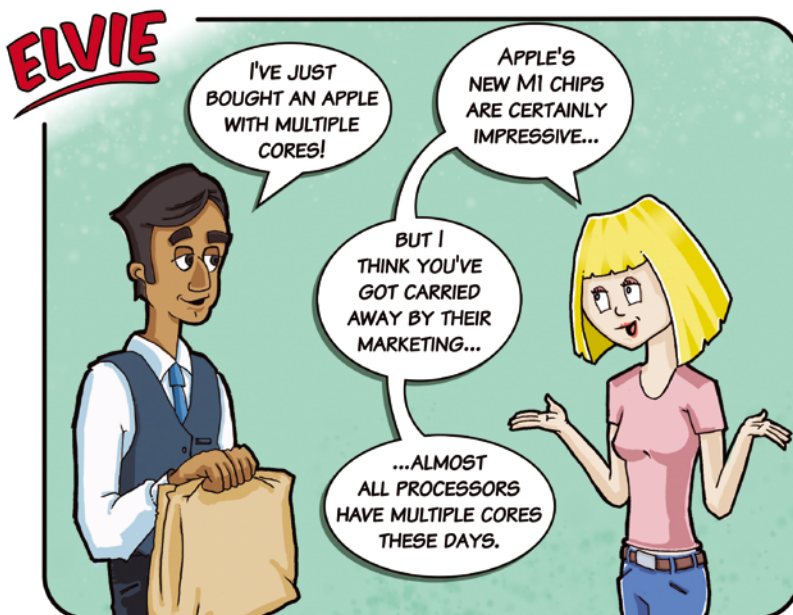
Also in this issue, we introduce you to Kit Scenarist, a free tool for creating and formatting screenplays, and we show you how to organize your thoughts and notes with the expressive Obsidian knowledge base.



Image © Olexandr Moroz, 123RF.com

# LINUXVOICE ▶

<b>Doghouse – Code Longevity</b>	<b>74</b>
<i>Jon “maddog” Hall</i>	
Maddog discusses the long history of text editors and the RAND message handling system.	
<b>EdUBudgie</b>	<b>75</b>
<i>Adam Dix</i>	
EdUBudgie Linux is an Ubuntu clone created by a teacher and aimed directly at the education market.	
<b>Kit Scenarist</b>	<b>78</b>
<i>Sirko Kemter</i>	
Creative writers take note! Kit Scenarist is a free application designed to simplify the process of writing a screenplay.	
<b>FOSSPicks</b>	<b>82</b>
<i>Graham Morrison</i>	
This month Graham looks at SonoBus, NewsFlash, Kinto.sh, RetroShare, Emilia Pinball, and much more!	
<b>Tutorial – Obsidian</b>	<b>88</b>
<i>Marco Fioretti</i>	
Obsidian helps you work more effectively by giving you a tool to record, connect, and catalog your ideas and notes.	



# MADDOG'S DOGHOUSE

As two examples of how open source code can evolve, maddog discusses the long history of text editors and of the RAND message handling system. BY JON "MADDOG" HALL



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

## The Long Life of Open Source Code

Recently I was discussing with friends how long certain pieces of code have lasted ... perhaps not exactly the same piece of code, but a recognizable version of the original that has continued to evolve.

One example of this is the simple text editor.

When I started programming, I used 80-column punched cards and the Fortran language. If you made a mistake, you could stick the bad card into a particular slot on the card punch, duplicate the correct part of the card, and then continue typing to complete the statement. It was horrible, but it was all we had.

The first editor I ever used was on a small mini-computer called the PDP-8. The PDP-8 had 4,000 12-bit words of memory and no storage other than paper tape. The computer had to hold all of the text editor and all of the source code of your program in memory at the same time. You had a paper-driven terminal called the ASR-33 Teletype, and the rolls of paper were expensive, so the text editor tended to print as little as possible. It was a modal editor, which meant that you were either inputting part of your program in input mode or giving editing commands to the program in command mode.

It was also what we called a "dot editor," since an invisible dot controlled where you were inputting data or from where the command was being given.

As an example, a command to print from the beginning of the line to where the dot was would be `0t.` and from where the dot was to the end of the line would be `.t$`. The command `0t$` would type the entire line.

You could also move the dot without the editor typing any output, and there were dozens of other commands for deleting characters, lines, and even blocks of characters.

This simple dot editor was called `e`. As more commands were added and it moved to early Unix, it became known as `ed`. Today this editor, like a lot of other Unix commands, might be called "user-unfriendly," but to people used to cards, it was great.

As memory sizes in computers grew, both the editor program and the text that it was manipulating could get larger. The editor's name was eventually changed to `em`, and then it was introduced to Bill Joy at the University of California, Berkeley. He improved it, and it became `ex`, which could be used for both hard copy terminals and "those new-fangled character-cell video terminals" in command-line mode.

Eventually Bill Joy wrote a full-screen text editor called `vi` that still had the modal commands of `ex`, and `vi` became one of the mainstream Unix editors for a long time.

The other mainstream editor was, of course, `emacs`, and these two editors created one of the longest running cyberwars of all time – the arguments of which are best left to another place and time over beer and pizza.

Eventually the functionality of `vi` was cloned into `vim`, allowing me to use (more or less) the same text editor for close to 50 years.

Yes, I know that somewhere `emacs` people are screaming, but I like `vim`.

Another fine example of a long-lived program is the RAND MH message handling system.

MH was really a series of command-line tools that created a front end for importing and reading email in the early days of Unix. It used the filesystem to hold all the email, so its "database" was very readable and could be manipulated with other Unix commands. I loved using MH.

Eventually email started including non-character data such as graphics and music. Since Unix at that time used 7-bit ASCII characters, an extension was needed to be able to encode the binary data into 7-bit data bytes, and this was called MIME. MH was then extended to handle MIME, and cat pictures began to flourish. On the command line, `mh` became `nmh` (new MH).

Eventually windowing came to Unix in the form of the X Window System, and a sort of "graphical shell" was made over `nmh` called `exmh`. Later still, Motif (another set of graphical widgets) was added to become another extension of the venerable MH system.

Since MH was first released in 1978 (and I first used it in 1983), it has gone through many incremental upgrades with the last release of the base `nmh` being in 2018.

Do not think of MH as dead, however, for the sources are still there for people to pick it up and keep improving.

This is the point of this entire article: Free and Open Source invites extensions of good functionality to meet changing needs, and often the code that you think may be dead is only resting somewhere waiting for the next person to look at it and update it again. ■■■

# New Linux distro for high school education

# A Clone on a Mission

EdUBudgie Linux is an Ubuntu clone created by a teacher and aimed directly at the education market. **BY ADAM DIX**

**E**dUBudgie Linux [1] is a distribution based on Ubuntu Budgie [2] but specifically tailored for use in high schools. I am an English teacher, and I put this project together with high school education in mind – although many first and second year university students may also find it suitable.

I started with the standard Budgie offering and added or removed packages based on their applicability to high school education. The education-focused IT market is a crowded one, and in deciding how to build and market this distribution I carefully considered package selection, initial setup, post-install system administration, the learning curve needed, and finally performance and longevity. With all that in mind, I did make some compromises in the build that I will discuss shortly.

I have intended EdUBudgie Linux to be as simple to learn as possible and not a distracting or overly complex distribution. My goal was to offer an operating system that could be used out of the box without needing to use the command line for setup, while still including all of the programs and pre-configured settings needed by a typical high school student or teacher.

Of course, EdUBudgie Linux should be used as part of a school or school district's carefully thought out, multi-generational IT plan. But by including EdUBudgie Linux in that plan, a school can expect to benefit from a free and simple alternative to Windows or Chrome OS. EdUBudgie Linux can be installed on new or old devices (including most x86-based Chromebooks) and can help solve many common problems plaguing students, teachers, and IT administrators.

While many readers will argue that any IT team with Ubuntu experience could put this package together, this distribution aims to take some of the guesswork out of the equation. The idea is not to limit the freedom that Linux users love, but rather to avoid choice paralysis. EdUBudgie Linux should also be suitable for Linux novices, which will be especially helpful in the many schools worldwide that have relied upon the Windows ecosystem for years.

## Installation

As previously mentioned, EdUBudgie Linux is an Ubuntu Budgie clone, and it uses the typical Ubiquity installer with a few small changes.

The most important change is the difference between the Normal and Minimal installations. Typically, a Normal Budgie install would include all programs, and that is still true of EdUBudgie. However, the EdUBudgie Minimal installation option will include many education-oriented programs alongside the typical Budgie accoutrement. The Normal installation option is typically recommended.

While the installation ISO occupies a rather incredible 5.8GB of space, the complete installation falls within 20GB. This allows EdUBudgie Linux to fit within a 32GB drive with a bit of extra space for files, so that it may be installed onto out-of-support Chromebooks without having to rely entirely on cloud storage options (see the "System Requirements" box). Many Chromebooks (especially older models) have replaceable solid state drives, and this upgrade may be needed to fit an EdUBudgie Linux onto some Chromebook models.

A few other factors should be considered when deciding whether EdUBudgie is a suitable replacement for Chrome OS on a school's existing devices. While I tailored this distribution with Chromebook conversions in mind, not all Chromebooks are the same, and the installation techniques needed may vary from one manufacturer or model to another. Given my limited resources in developing this distribution, you will need to test it on your school's specific devices, and you may need to research how best to install it on them.

I developed this project with the aging Acer C720P in mind, as it is a commonly used Chromebook in schools. After changing from a 16GB disk to a 32GB disk, installation was straightforward. Specific and detailed instructions can be found from a number of sources, but essentially, for the C720P model it required removing the 13 screws on the bottom of the device to access and replace

**System Requirements**

**Minimum System Requirements**

- 1.4GHz dual-core Nehalem microarchitecture 64-bit processor
- 2GB system memory
- 32GB drive
- Intel-integrated graphics

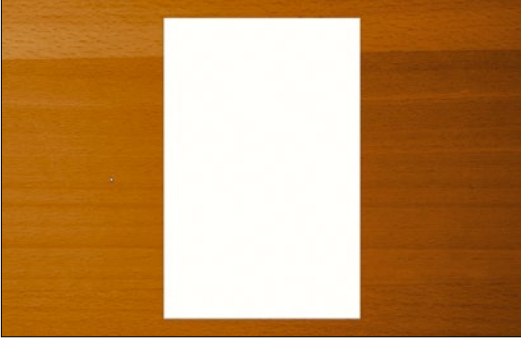
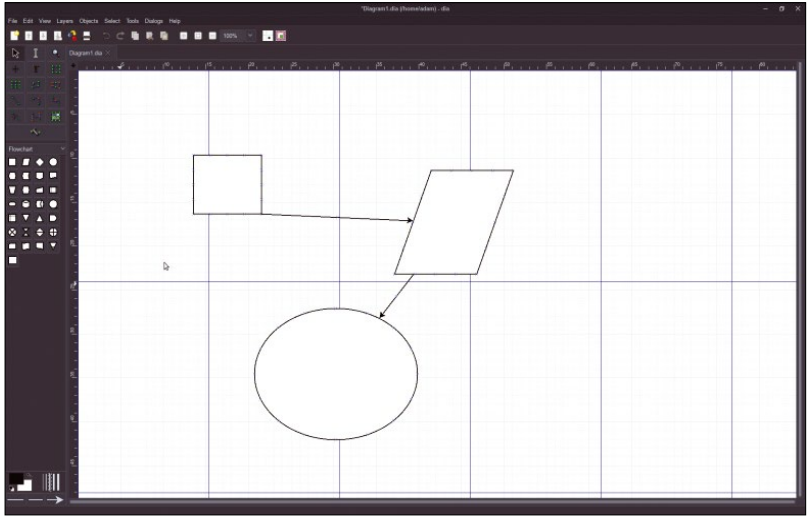
**Recommended System for Average Performance**

- 2.4GHz dual-core or 2.0GHz quad-core Sandy Bridge microarchitecture 64-bit processor or newer
- 4GB system memory or greater
- 64GB drive or larger
- AMD Radeon dedicated graphics of at least R7 class or better

the 16GB SSD with a 32GB one, as well as accessing and removing the write protect screw that serves as a security feature on some Chromebook models. After completing these hardware changes, it was a matter of enabling developer mode, the developer BIOS (SeaBIOS), and USB boot so that EdUBudgie Linux could be permanently installed [3]. To be clear, the method for doing this will vary based on the specific device you're using, and that will void any remaining device warranty from Google.

My hope is that schools with budget concerns may be able to extract a few more precious years out of the devices they have already purchased. If well-planned, this can fall in line with a school's overall technology plan. An administration may choose to utilize tablets or iPads through say, 6th grade, then transition students to supported Chromebooks, and finally use older Chromebooks and laptops with EdUBudgie installed for students

**Figure 2:** The Dia diagraming program is useful for brainstorming and creating all sorts of systems.



**Figure 1:** The FocusWriter program encourages distraction-free writing.

from perhaps 9th grade on up. This segmentation makes sense in terms of a student's development. Younger students are able to easily produce quality work with simple devices, while they are less likely to divert their time away from work toward games. Middle school-aged students would benefit greatly from becoming accustomed to using full keyboards on a minimal OS like Chrome OS. Finally, high school-aged students need a full suite of programs to prepare them for university.

**Included Programs**

Aside from the desktop environment, package selection is the major differentiating factor between EdUBudgie Linux and any other Ubuntu clone or education-focused operating system in general. More than anything else, this is what will make or break the distribution, and therefore I have put much thought into package selection.

Quite likely the first thing that comes to mind when choosing an operating system for education is office software, and the particular needs will vary greatly from school to school. For this reason, EdUBudgie includes LibreOffice and WPS Office and can of course be used with on-line office suites such as Google Workplace and Microsoft 365.

LibreOffice has some unique features that teachers and students may want to use, such as the LibreOffice Math component. However, WPS Office is often perceived as more familiar and inviting, especially when migrating from Microsoft, the de facto standard for office programs. Additional office-like or office-adjacent programs included are FocusWriter (Figure 1), Dia (Figure 2), Scribus, and MindMaster.

The default browser is Chrome – we can pause here for gasps, curses, and cleaning up spill or spit coffee – but for good reason. I did not intend EdUBudgie to be a completely FOSS offering at present, and simply put, it cannot be. Any time teachers or students spend installing extensions, or researching and learning work-arounds on Chromium or Firefox, is time that could be spent on education. Other browsers

cannot offer the practicality that Chrome does at the time of this writing. Chrome OS cemented Chrome's position as the dominant browser in education. While the Linux community yearns for and works toward an acceptable open alternative, none currently exists. This should be looked at in a positive light: As soon as a suitable alternative exists, it will be incorporated into EdUBudgie Linux. Until then, education relies heavily on Chrome.

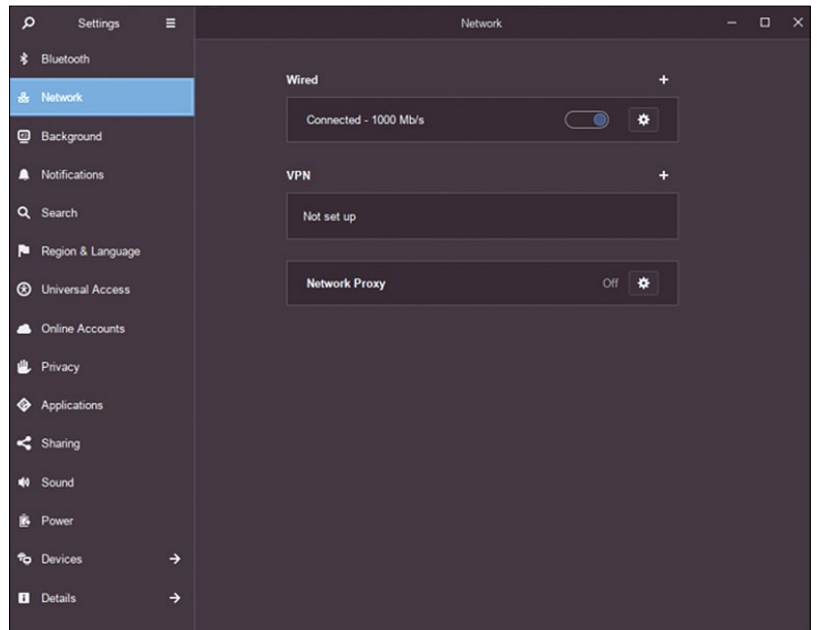
Along with Chrome come many web apps commonly used in the classroom. Additionally, Geary can be used for email, and the Gnome Calendar allows for easy integration with Google Workplace. EdUBudgie Linux installs OpenDrive by default for those who prefer to have local copies of the files they are working on, and the Online Accounts section of Settings (Figure 3) allows for seamless Google Drive integration in the file explorer. Using a Windows 10-like set of icons should make file and folder manipulation second nature for those migrating from Windows-based machines.

For most schools these will be the most-often used programs, but many others will be installed by default. I selected many programs because of their relation to specific disciplines. Among many other programs, these include the following:

- Gimp, Inkscape, and darktable for art classes
- Blender, LibreCAD, and FreeCAD for engineering and other technical disciplines
- KdenLive, OpenShot, OBS, Kazam, and Cheese for video production and editing
- Scratch, Atom, Geany, and Basic-256 for coding
- GeoGebra (Figure 4), KAlgebra, Tilink, and Qalculate for mathematics
- Kalzium for chemistry
- KGeography for social science classes
- Calibre for language classes

## Conclusions

The intent of EdUBudgie Linux is to offer a complete Linux distribution for education and to hopefully bring many new users into the Linux community. The open source model relies upon a vast community of intelligent and creative individuals to advance the cause. Unfortunately, Android and



**Figure 3:** A typical Gnome/Budgie Settings interface.

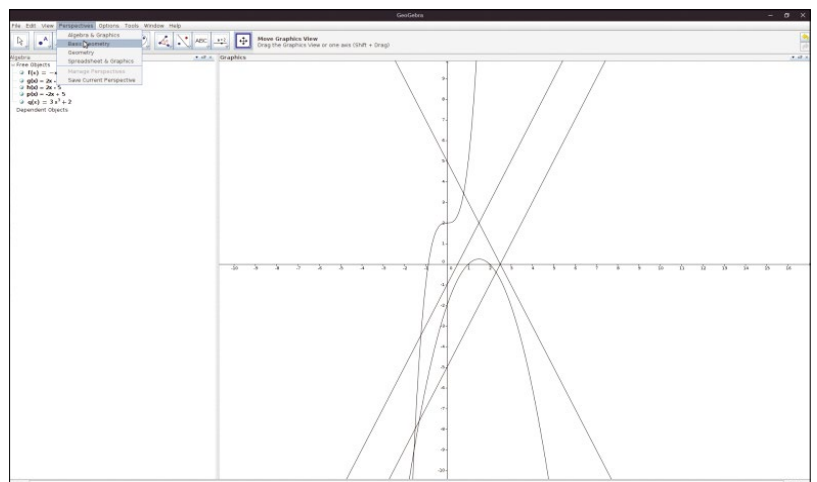
Chrome OS have somewhat obfuscated in the minds of many what is and what is not Linux. Most students who own a Samsung phone have no idea that it has any relationship to Linux. EdUBudgie Linux is clearly, unabashedly, unapologetically, Ubuntu Linux.

The vision of EdUBudgie Linux is twofold – to expand the Linux and open source communities so that future creators can rely less or, ideally, not at all on giants such as Microsoft, Google, and Apple, while also offering high school students of all socioeconomic backgrounds equal access to the latest that the community has to offer. ■■■

## Info

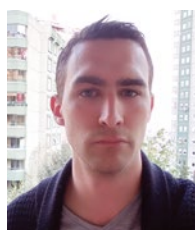
- [1] EdUBudgie Linux: <https://www.edubudgie.com/>
- [2] Ubuntu Budgie: <https://ubuntubudgie.org/>
- [3] Installing Ubuntu on the Acer Chromebook C720P: <https://samsclass.info/128/proj/chromebooks3.htm>

**Figure 4:** The GeoGebra program is useful for mathematics classes.



## The Author

**Adam Dix** is a mechanical engineer and Linux enthusiast posing as an English teacher after playing around a bit in sales and marketing. You can check out some of his Linux work at the EdUBudgie Linux website.



# Write screenplays with Kit Scenarist

# Writing Workshop

Creative writers take note! Kit Scenarist is a free application designed to simplify the process of writing a screenplay.

BY SIRKO KEMTER

Screenplays require a specific format and make special demands on the author. Unlike a novel, a screenplay virtually reduces the story to the dialogue. You'll also need to use a special layout based on a fixed-width font and wide margins – a standardized format that makes it easier for production companies to estimate the length of the film.

Rather than messing around with setting up this layout in a word processor, professionals use special programs, which may also include useful functions such as helping keep track of characters and locations. If you're looking for these tools in a free application available on your Linux distribution, you won't find many options, but one that may fit your needs is Kit Scenarist [1].

## Installation

Although the cross-platform software is currently not available in the package sources of most distributions, the project website has many options for Linux users, offering RPM and DEB packages for Fedora and openSUSE or Ubuntu and Debian, among others, and the installation is usually a fairly convenient process. Only Arch Linux and its derivatives, such as Manjaro, have

the program directly in the repositories where it goes by the name of *scenarist*.

When the application is first launched, a wizard helps set up the language and theme for the interface (Figure 1), various modules, and the template for the script. Here, if you do not already have concrete specifications from a company, it is best to choose one of the options for Final Draft, either A4 (for the European format) or Letter (for the US format), as shown in Figure 2. Both templates are based on the film industry's quasi-standard.

## First Steps

After starting the program, the first thing that appears is an overview of the projects that have already been created. You have the option to open or import a project or to change basic parameters via Settings. Open the corresponding dialog and activate the spell checker in Program. The software first tries to load the preset Russian dictionary from the Internet, but you can select English or one of many other options as your preferred language.

The next step is to create a new project and start writing. Clicking *Create project* opens a dialog

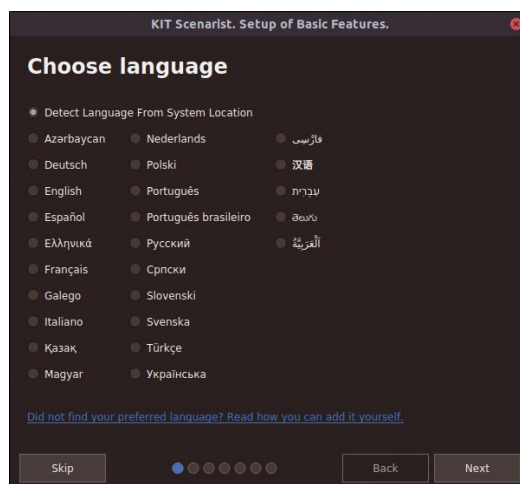


Figure 1: When first launched, a wizard helps you set up the software, including the language for the interface.

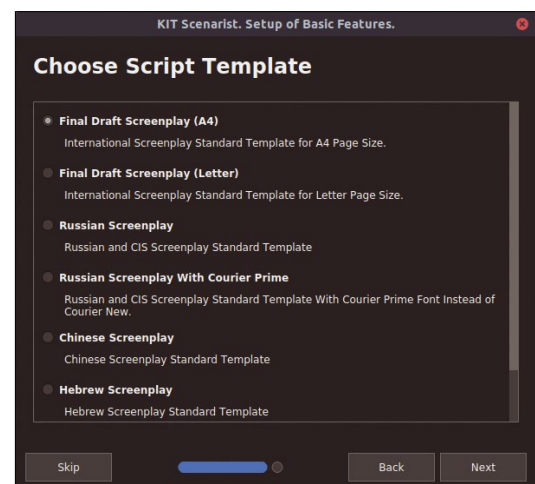
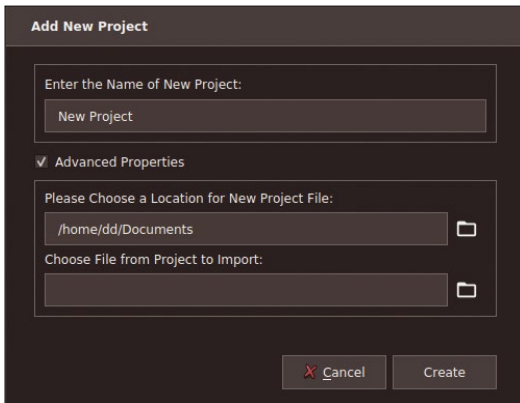


Figure 2: You can either select the document format during the installation or change it later in the project under Settings.



**Figure 3:** To create a new project, simply enter the project name and choose where to save the file.

where you give the project a name (Figure 3). The program will use this later when saving the file. With the *Advanced* options you can change the storage location.

## Research Window

After creating the project, the software will direct you to the Research window, where you enter further basic data and create the title page. You can save characters and locations here, as well as link external data such as documents, images, and links. The concept is similar to the Snowflake Method [2] for creative writers. You'll find this information useful when writing the actual document.

The program has some helpful automatic features. If you name a location later on in the script, the program automatically creates it in the Research section. However, this does not work for characters. If you add a new character as you're working on the script, it's a good idea to create

them in the Research section first and then use them in the script (Figure 4).

If you create characters as Scene Characters, they are automatically also created in the Research window. If you mention a role in the script, an auto-complete will be available for that character in the future. Renaming a character later in the book, however, is a process that can only be handled through the Research section.

Right-clicking on *Characters* opens a context menu where you can search for characters in the script using the *Find characters in script* entry. This function lets you transfer roles from a script to your research.

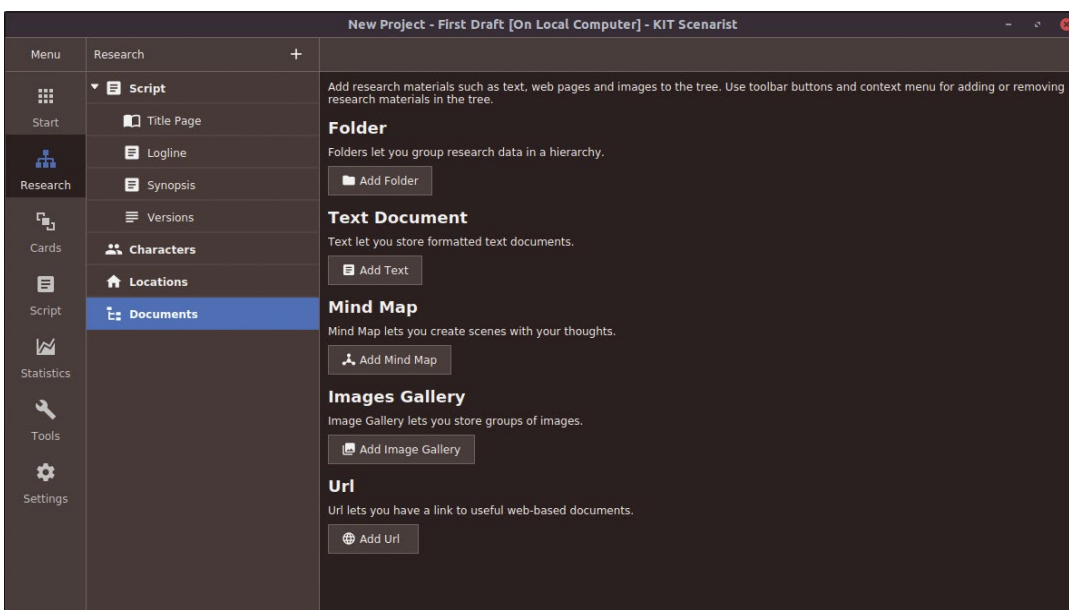
In the Cards overview, the background changes to a pinboard. Small cards represent the scenes. At the top, you have the option to rearrange the order of the scenes and print these cards as an overview.

## Centerpiece

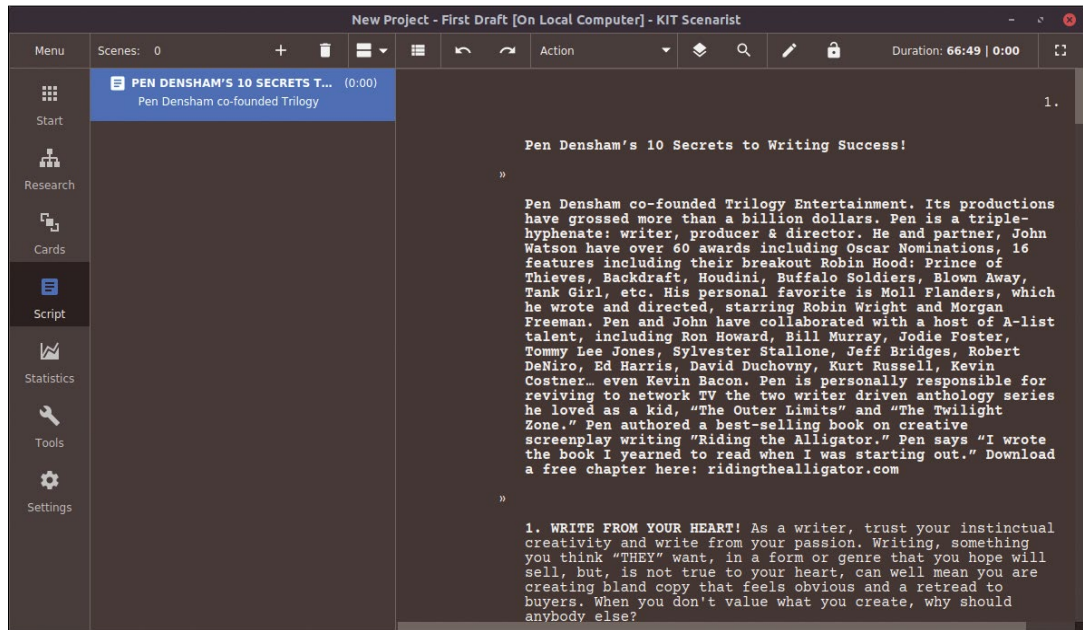
The Script window is the heart of the program. Here you can see one way that screenwriting software makes your work easier. On the left side of the window is the overview with the scenes; here you can simply re-sort the order using drag and drop. On the right there are buttons to format elements directly. In the center is the area for writing.

Since you start with an empty document, the program knows that a Scene Heading is missing – pressing the spacebar immediately prompts the program to suggest appropriate words for a heading.

If you want to insert the Scene Characters, click on the corresponding button on the right; otherwise just start writing – the program formats your input automatically.



**Figure 4:** The Research overview, with the selected document overview here, helps you to arrange the roles, locations, and events in the script.



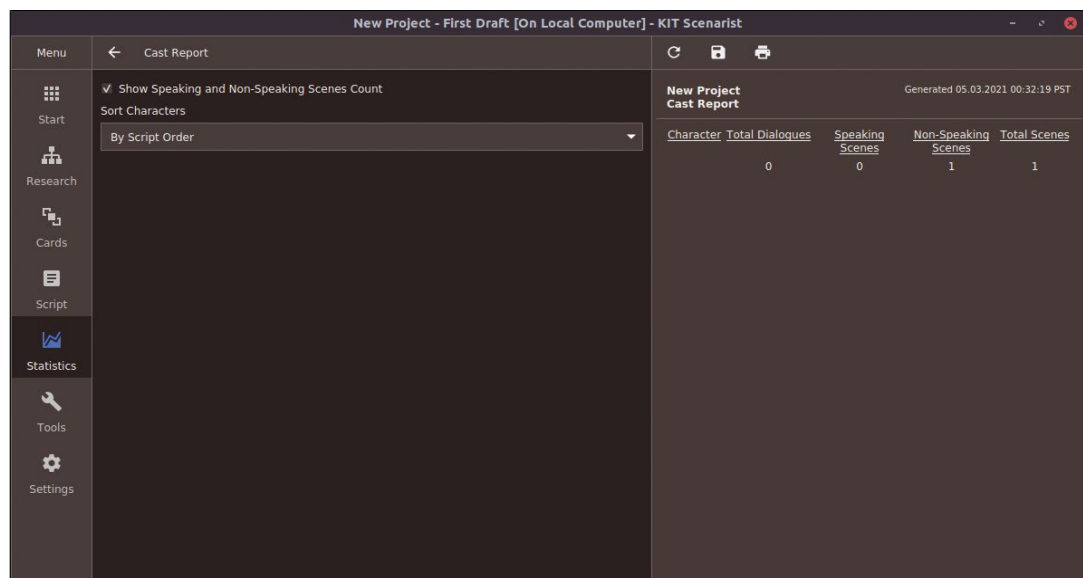
**Figure 5:** The heart of the program, the screenplay view, formats all parts of the script according to the specifications from the template.

Format	Keyboard shortcut
Scene heading	Ctrl+Enter
Scroll	Ctrl+E
Action	Ctrl+J
Role	Ctrl+U
Dialogue	Ctrl+L
Stage Direction	Ctrl+H
Shot	Ctrl+P
Transition	Ctrl+G
Do not print	Ctrl+Esc
Lyrics	Ctrl+K

It's a good idea to record the Scene Characters, though, because there are always scenes where characters appear but have no dialogue – for example, when a character witnesses an action because they need to report on it later on. Among other things, the Statistics module lists Scene Characters with and without dialogue.

The software also offers the possibility to compare versions, but only if you saved them manually. Automatically saved versions are left out here for some unknown reason.

If you use the Character button, the software will immediately offer the characters known so far, whether from the research or from the roles mentioned so far. This saves you a good deal of typing and makes your writing more efficient.



**Figure 6:** The Cast Report shows the number of scenes in which a role appears and where it has dialogue.



After the Character, the program will switch to the format for a dialogue and then back to Character the next time the Enter key is pressed (Figure 5). You can start a new scene manually by clicking the corresponding button or by using the shortcut Ctrl+Enter in the line. For more keyboard shortcuts, refer to Table 1.

## Statistics

Although most of the overviews offered are more useful for the later implementation of the script, they can already be of some help in the writing process. In particular, it can be tricky to develop a script that meets specific targets for running time. The use of clearly defined formats helps to estimate the approximate length, and Kit Scenarist even helps with this as you write, displaying the time for each scene as well as the total time in the upper right corner.

Kit Scenarist provides overviews of the distribution of dialogue among characters (Figure 6). It also shows the locations, as well as the overall structure of the script – that is, how much dialogue, how much action, and how much stage direction or other elements it contains (Figure 7). This will help you find places you could restructure or perhaps cut to reduce the length of the screenplay.

The last module gives you the ability to restore backup versions. Kit Scenarist saves the current document every five minutes by default unless you change this.

## Conclusions

I did have some difficulty with a few aspects of the program. The developers seem unfamiliar with many of the mechanisms used for free

software in general and Linux distributions in particular. When I contacted the developers to suggest that it was inconvenient to have to regularly check the website to see if a new version was ready, there was only a reference to new websites that offer push messages. The fact that dictionaries can be installed and updated as files via the distribution seems completely unfamiliar to the developers. They also struggle with security issues: In my experience, none of the packages on the project's download pages were signed.

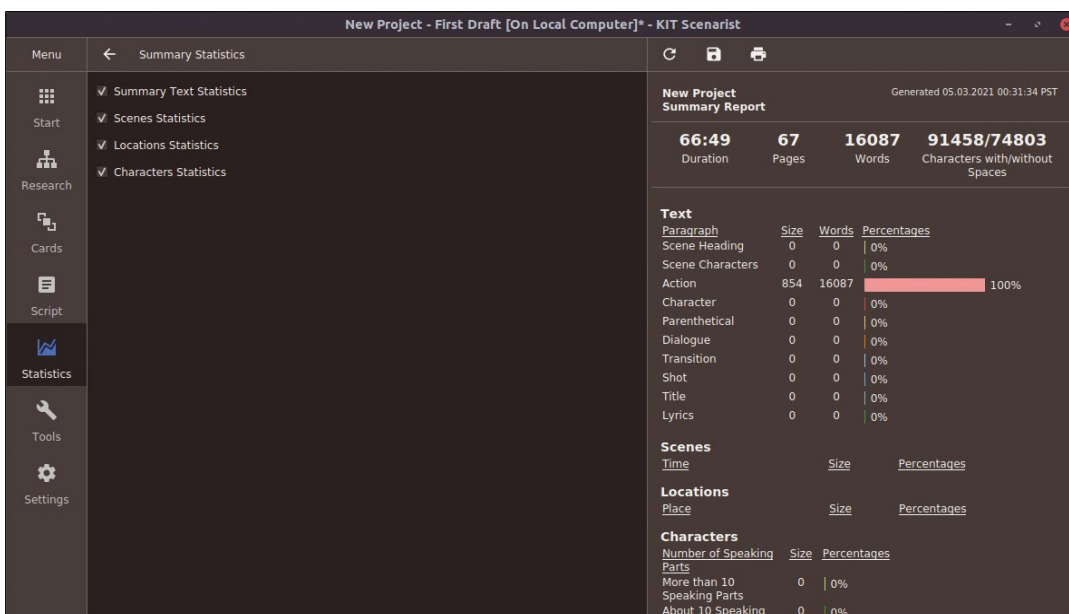
However, if you can live with these issues, there are many reasons to like Kit Scenarist. It offers a tidy user interface, and all the important functions are present and work excellently. While there are some less useful components, such as the card overview, and many of the statistics are not especially helpful during the initial writing, the program is still quite practical, and far more convenient than a word processor. ■■■

### Info

- [1] Kit Scenarist:  
<https://kitscenarist.ru/en/download.html>
- [2] Snowflake Method:  
<https://www.advancedfictionwriting.com/articles/snowflake-method/>

### The Author

Sirko Kemter uses screenwriting in language classes to get young people interested in reading and to teach them the function of grammar.



**Figure 7:** Kit Scenarist provides extensive statistics, including a summary report that provides great detail about the script's structure.

# FOSSPicks

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



Following the news of official Widevine support on the Raspberry Pi, used to play Netflix, Prime, and Disney+, Graham has finally been able to ditch his Amazon Fire TV advertising conduit for Kodi. **BY GRAHAM MORRISON**

## Network audio

# SonoBus

When it comes to communication, it's amazing how adaptable the human brain can be. In conversation, it can almost completely filter out background noise, accommodate strong accents, and error correct sentences better than 200 percent checksum redundancy. What it can't do, at least not comfortably, is predict and iron out timing issues. If you've ever watched a film with the audio out of sync, you'll know that a certain amount of delay is imperceptible. But there's a threshold where it quickly

becomes impossible to tie the spoken words to the people who spoke them, and the same thing happens in a video chat and with musicians, where distance can have an effect on latency and performance. A significant delay makes any kind of musical collaboration impossible, and that means that while the modern Internet can easily accommodate Zoom calls with two dozen people, it's more difficult to satisfactorily connect even a couple of musicians together, despite the lesser requirement in bandwidth.

Over the years, there have been many attempts to solve this problem, such as the closed source TeamSpeak and the open source Mumble. Mumble perhaps gets closest with relatively low latency and uncompressed multitrack audio options. But it's never felt right for musicians. SonoBus, however, does feel right. It's an open source peer-to-peer audio streaming solution that promises both high quality and low-latency audio across a LAN or the Internet. It's easy to install. After launching the application, the first thing you notice is that the UI is simple, well designed, and easy to use. The first thing you need to do is either connect to another user or share your own session. Both these actions can be accomplished through either a public or private group created on a server, which defaults to `aoo.sonobus.net`, or via a direct connection to the ports and IP addresses of the machines you wish to connect to. You can even run your own connection server to get the best of both worlds.

With this done, a new row will start to appear in the UI for each machine connected to the session, and everyone can speak or listen at the same time. Each row includes controls to adjust volume and left/right pan, as well as an input meter and the ability to add compression, and a noise gate effect to help with legibility and remove background noise. Latency will depend on hardware and network conditions, and each row includes specific details on the upstream and downstream latency, bitrate per channel, and the size of the jitter buffer that is used to iron out variations. The results can be staggering and far better than other chat or audio streaming solutions we've used. Even across hundreds of miles, latency was typically less than 50ms, and often lower, making not just conversation more natural but also opening up the very real possibility of musical collaboration or direct podcast recording. This is helped by the ability to adjust audio quality anywhere between an Opus compressed low (16Kbps) bitrate to a fully uncompressed 32-bit PCM, perfectly acceptable for professional audio production.



**1. Audio back end:** SonoBus can use either ALSA or JACK to get the lowest possible latency from a configuration. **2. Effects (FX):** Each client has compression and gate effects, with more effects for the global output. **3. Monitoring:** Listen to your own mic and carefully control the volumes of everyone in the chat. **4. No security:** By default, audio streams are not encrypted. If you're going to be chatting about film spoilers, we suggest using a VPN. **5. Quality:** SonoBus is capable of professional quality recordings, as well as offering low-bandwidth friendly compression. **6. Mixdown:** Record an entire meeting or session as a single file or multiple files. **7. Metronome:** If you're recording music together, use the metronome to keep everyone in sync.

## Project Website

<https://github.com/essej/sonobus>

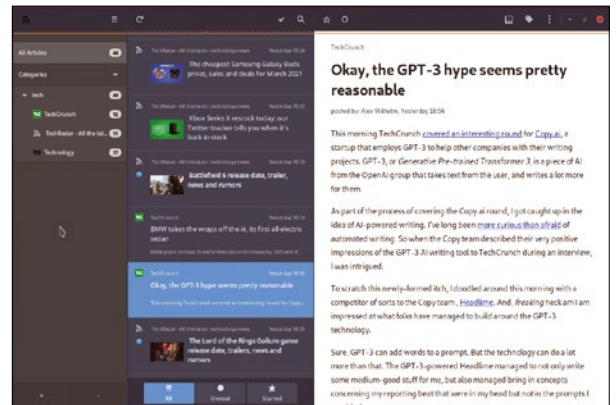
## RSS reader

# NewsFlash

**R**emarkably, it's been eight years since Google closed its RSS Reader project. And yet, RSS survives and is more relevant now in the push information age than ever. Fortunately, there are still many websites that publish their own diminutive XML feeds containing brief synopses of their updates. Any RSS reader can then track and pull these updates, free of cookies and click-through advertising. Despite its age, RSS remains a refreshing alternative to the commercialization of the Internet, from sponsored newsletter placement and paid-for-tweets, to GIFs as non-fungible tokens (NFTs). Which is why it's brilliant to see another new RSS reader, NewsFlash, and a sign that RSS might be going through

a small renaissance. NewsFlash is a new, small, and beautifully designed RSS reader that makes RSS feel more like an integrated part of a modern desktop, rather than something from a bygone era.

If you exported your Google Reader subscriptions to some RSS enclave, there's a good chance NewsFlash can use it. It will import your current RSS subscriptions from Feedly, Fever, NewsBlur, Feedbin, and Miniflux, as well as letting you add feeds yourself. The latter is made easier by an excellent integrated RSS browser that helps you to discover quality feeds from a categorized list. The modern Gnome design is perfectly suited to an RSS reader like this because the UI is both minimal and functional,



**Forget the distractions of social media and get your news with an old school method that still actually works: RSS.**

with very little wasted space. The three column view first lists the categorized feeds you're subscribed to, and then an aggregated list of stories from the selected feeds or category, before the rightmost pane shows the contents for the selected item. It's perfectly augmented by titlebar icons to favorite an item, grab an attachment, and even scrape the full story, which is something you can't do with any other desktop toolkit and a great reason to choose NewsFlash whatever your desktop environment.

**Project Website**

[https://gitlab.com/news-flash/news\\_flash\\_gtk](https://gitlab.com/news-flash/news_flash_gtk)

## Audio editor

# Audacity 3.0

**I**t's a cliché, but Audacity really is one of those flagship open source applications that many people simply couldn't do their jobs without. It allows podcasters, community radio stations, and amateur musicians the world over to create serious, professional results without the serious outlay commercial applications often require, none of which work natively on Linux anyway (except BitWig!). Which is why, despite its modest set of new features, the release of Audacity 3.0 is still significant. We're simply grateful that the application continues to be developed, and we're particularly excited by the news that work is being undertaken to overhaul the outdated UI that has held it back from

properly integrating with modern desktops, even if that isn't in this release. But this isn't a reason to discount version 3.0. In releases prior to this one, a project's AUP file only linked to the audio files, which means things could easily break if the AUP was lost or corrupted. Audacity 3 solves this problem.

A new `.aup3` format embeds the metadata alongside the audio in a single file, allowing you to archive work and restore a session regardless of where you move the file or load it from. This has been accomplished by using a SQLite3 database internally to handle storage and retrieval, and it's much easier and even quicker to use than the old method. It's disappointing that you can't make



**Alongside a new audio project format, Audacity 3 has updated the noise gate effect and the analyzer.**

the default save requester simply choose a WAV or MP3 file, but we do understand why it selects the internal format to save every parameter in a project. It's much like XCF being used by Gimp, but both projects are more likely to be used for quick import and export edits rather than prolonged projects. We think they'd both benefit by aligning the save requesters with what users expect.

**Project Website**

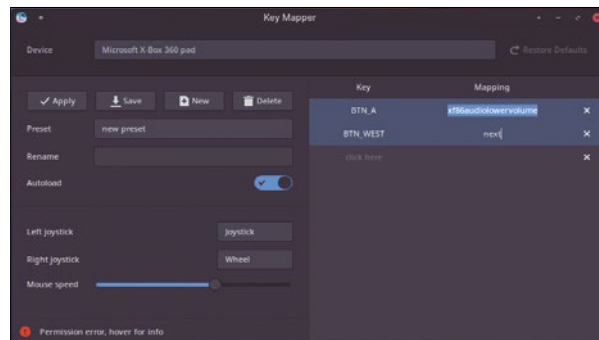
<https://www.audacityteam.org>

## Map buttons

# Key Mapper

As someone who uses an older MacBook Pro with Linux natively installed, it's imperative that Apple's special Control and Option keys are mapped to work with Linux effectively. This is a rather arcane two-step process, first because you run the ancient `xv` command to divine the raw keycodes for those keys, and second because you enter those keycodes into an equally ancient `xmodmap` file to map those raw keycodes into key presses. As the `x` in both tools suggest, this process is unlikely to work with Wayland. Key Mapper is a Python application with a modern GTK+ front end that does all this easily and adds the ability to map buttons and keys from almost any input device. This is the clever part, because input devices

can include joysticks and controllers, as well as keyboards, which you select from the device drop-down at the top of the window. With a device selected, you use the key pane on the right to press the input you want mapped before selecting from a drop-down list what you want it mapped to. This list is absolutely huge and includes all the XF86 shortcuts, as well as triggers, special keys, and touchpad controls. If you're mapping a games controller, you can map its joysticks to mouse movement, the mouse-wheel, buttons, or a single joystick, complete with mouse speed. This is brilliant if you're building a Linux arcade machine, for example and want the joystick to control the mouse cursor while at the same time mapping



One of the best things about Key Mapper is that it will work with Wayland, giving you a migration path from Xorg mappings to the brave new world.

some of the buttons to launch the games you want. This entire configuration can be saved as a preset for easy recall, and you can have as many presets as you need. Presets are useful if you want to step away from the computer, for example, and control it remotely, or if you switch keyboards between an Apple keyboard on the road and a more conventional keyboard when you're home. A background daemon is also installed and activated by default, and this takes care of the changing configuration, as well as making any necessary changes when you reboot. It works perfectly.

### Project Website

<https://github.com/sezanzeb/key-mapper>

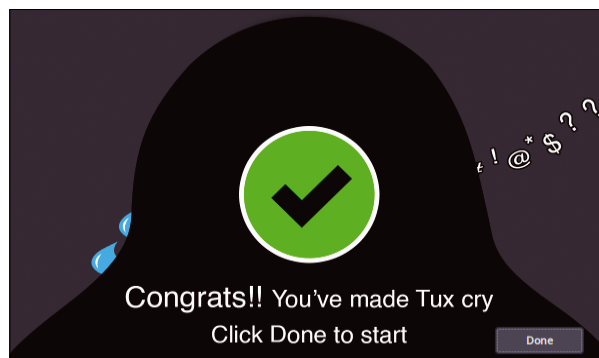
## macOS shortcuts

# Kinto.sh

Mapping special keys from your keyboard into meaningful Linux commands is a common problem that can be brilliantly solved by Key Mapper (above) or magical `xmodmap` incantations. But the inverse is also a common issue; mapping Linux keyboard commands to their macOS or Windows equivalents. This can help if you're running macOS or Windows in a VM, in a remote desktop, or via a remote command-line session. It's particularly noticeable when you're using macOS, because it relies heavily on its special keys for fundamental functions such as copy and paste. This is where Kinto.sh can help. It's a small Python script alongside a few configuration panels that can take over your

keyboard and map those Linux special keys to macOS special keys, enabling you to invoke the same shortcuts regardless of which operating system you're interacting with. For instance, by default on macOS, `Ctrl` is mapped to `Command`, `Alt` to `Option`, and your `Super` key to `Ctrl`.

Much like when you install Linux and are asked to press a few keys so that your keyboard layout can be detected, Kinto.sh does the same thing so it can identify which key is most analogous to Apple's `Command` and `Option` keys. With this done, a background daemon will run and a terminal will open showing the system log filtered for the Kinto.sh service. This shows when keyboard shortcuts have been successfully transformed into their destination equivalents,



Kinto.sh transforms native Linux keyboard shortcuts into macOS and Windows equivalents. It also has some unexpected moments of humor.

which should work without any further configuration. The system claims to be a complete system-wide remap of base modifier keys while retaining the commands for your native environment. You can use the well-known keyboard remapping tool `xkeysnail` to make your own keyboard mapping configurations that work with Kinto.sh, and per-app definitions can be added to a configuration file. It may be niche, but Kinto.sh solves a huge problem if you work daily with these other operating systems from your Linux machine.

### Project Website

<https://github.com/rbreaves/kinto/>

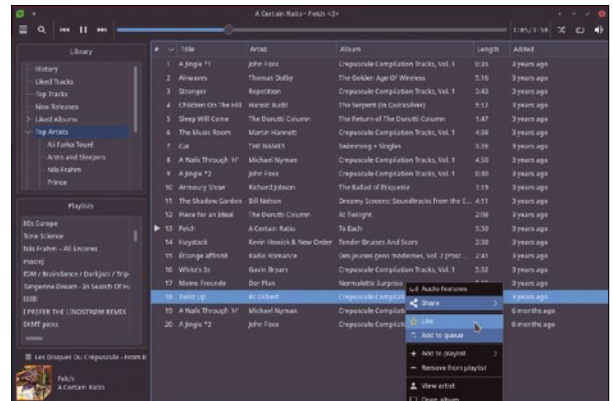
## Unofficial Spotify client

## spotify-qt

Spotify has become hugely successful. While there is a semi-official Linux client, it has always felt more like a hobby project than a first class citizen. It's perpetually in beta and seldom has feature parity with the Windows or macOS versions. What's worse is that it's built on Electron, making it resource hungry and not particularly well integrated into the Linux desktop and its various window managers. It's hard to use with a tiling window manager, for example. Fortunately, and to Spotify's credit, third-party libraries have been developed that can access Spotify's API and consequently allow alternative clients to be easily developed. The best of these is a background daemon

called spotifyd, and there are now several clients that take advantage of this. The brilliant spotify-qt is one of them.

You need to be a paying Spotify subscriber to get the required API access in spotifyd, which you then run with your username and password. Any spotifyd clients, including spotify-qt, can then connect through the daemon to present their improved user interfaces. Spotify-qt, for instance, looks and operates just like a native Qt or KDE application, with support for playlists, searching, favoriting, and keyboard control. There's a dark mode and the option to change its panel icon into album art. A separate panel can be opened to show you details about a song that you can't see in the official client, including its



The application supports many features including playlists, searching, and favoriting.

key, tempo, time signature, "acousticness," and "danceability," although it's not always accurate. All of these details are from the official Spotify metadata, and the developer is working on adding support for lyrics. You can even beam your music selections to other Spotify clients on your network, such as a television or amplifier. It's a much slicker application than the official client and will integrate perfectly with your local themes, fonts, and audio back end.

## Project Website

<https://github.com/kraxarn/spotify-qt>

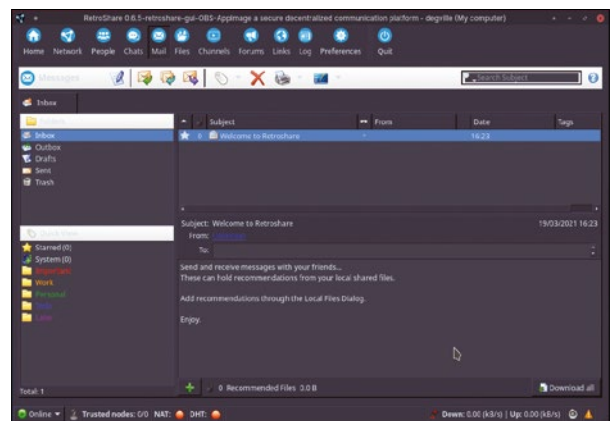
## Private communication hub

## RetroShare

One of Mozilla's best projects for Firefox was Send, an inbuilt mechanism for sharing files securely and anonymously with other users. It was incredibly convenient if you wanted to send a document to a family member, or share even a large file with a group of friends. Unfortunately, this convenience came at a cost – both in terms of the interim storage used by Mozilla and because "some abusive users were beginning to use Send to ship malware and conduct spear phishing attacks." Send development stopped, and the service was discontinued. Of course, there have always been alternatives, and one of our favorites is a command-line tool called Magic-Wormhole, which

works brilliantly even across networks. But it does require both the sender and the receiver to use the command line, and they also both need a side channel in which the receiver's code word can be shared.

RetroShare is a Qt-based desktop application that solves this problem while adding secure and decentralized chat, group discussions, mail messaging, and forums into the mix. It does this through a friend-to-friend network protocol, which is basically a peer-to-peer network where your friends make up a chain of trust and users outside of your chain of trust on the same network are undiscoverable. It accomplishes this by asking you to create an account with your



RetroShare is the modern equivalent to old school P2P networks, only just with your friends and whatever you choose to share.

GPG ID, which then becomes visible to your friends so they can verify your identity via a certificate. After adding a contact or three, you can then make use of the various communication channels built into the app, from creating a chat room to choosing to share files with your friends or anonymous friends of friends. RetroShare can even use Tor v3 as the transport layer to help keep your transfers and communication as private as possible.

## Project Website

<https://github.com/RetroShare>

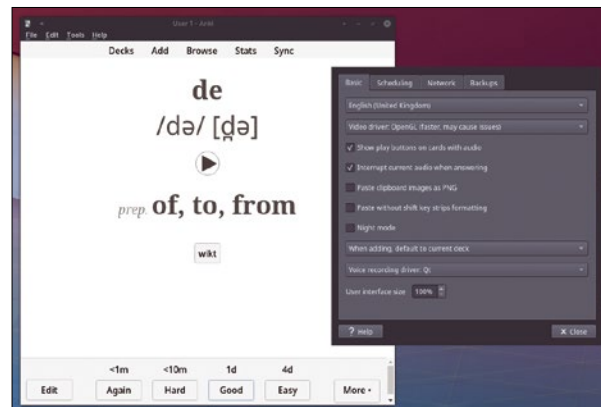
## Virtual flashcard deck

## Anki

Technology has transformed education in unprecedented ways, from easy and free access to in-depth video tutorials and complete online educational courses, to group learning and one-on-one virtual coaching. But technology hasn't been quite as successful at transforming the actual way that we learn, despite brave attempts from the likes of Khan Academy and Duolingo. Learning still requires hard work. There are, however, lots of tools that can help, and one of the best of them is Anki. Anki is an application that helps you create flashcard decks and train yourself from their contents. For a piece of software, this premise may not seem too adventurous, but Anki has been hugely successful already by helping students across the globe study for exams using their tablets, smartphones, and web browsers. It's successful precisely because it's simple, concise, and ultimately helpful. This success has also fueled the development of hundreds of decks you can easily use for your own learning.

When you start the application for the first time, you need to first either create a new deck of flashcards or import a previously created set. This second option is perhaps Anki's biggest advantage because, thanks to its educational ubiquity, there are hundreds of often freely available sets you can download and install, as well as commercial sets if you're willing to pay. The application itself links to its own community where sets can be freely downloaded and rated, from 5,000 of the most common French words to the anatomy of lower limb muscles. It's also incredibly easy to create your own sets, and this can be particularly helpful when you're working from your own notes or specific course material.

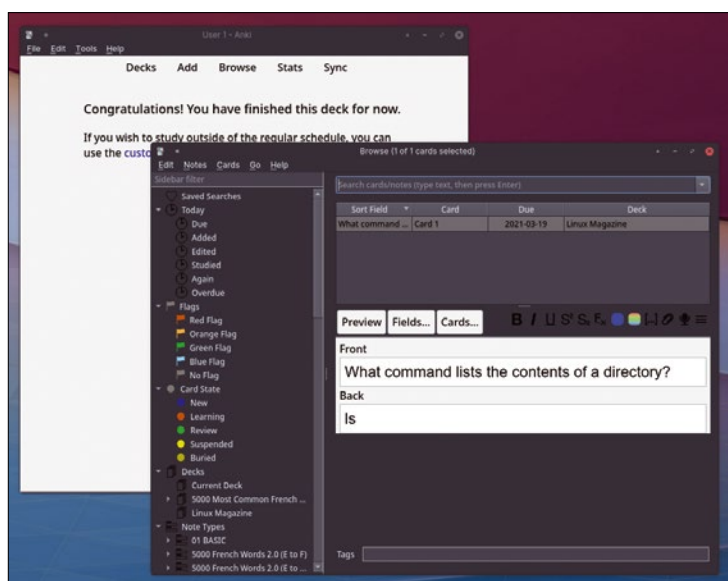
After creating a deck, cards are added via a variety of forms. The simplest asks you for something to put on the front of the virtual card and something on the back. This is typically a question or word to translate. When you eventually study the pack, you see the question and press the spacebar to see the "answer" on



The frequency with which a card is presented is dependent on how hard you find recalling its information.

the reverse of the card. You have to personally judge whether your answer was correct, rather than being scored. You also need to select a button to show whether remembering the answer was *Hard*, *Good*, *Easy*, or if you need to see it *Again*. Each of these options sets how soon the same card will reappear. The delay is shown above the answers, and it could be less than a minute for *Again* or four days for *Easy*. This, and setting a practical card limit for each practice session, is what makes Anki so effective.

Alongside the simple front/back example styles that are included by default is the card type called "cloze deletion." This lets you easily paste text into a card and mark sections that need to be guessed at, with the complete text only being revealed when you flip the card over. But you can also easily create new card types, adding images and even sounds that can reveal certain answers or pose specific questions. The popular French word deck, for example, includes audio pronunciation files as part of the answer, and it automatically retrieves the Wiktionary entry for a word. You can create or install as many types as you need, or use those from decks you've already imported, helping you get the most out of whatever time you can spare to enhance your learning.



Anki helps you create your own multimedia flashcards, or pre-built stacks, to make it easier to accelerate your learning.

**Project Website**  
<https://apps.ankiweb.net/>

**Cannon Fodder engine**

# Open Fodder

**T**here can be few old-school Amiga types who didn't play the classic Cannon Fodder. It was a remarkable game with an emotional and strategic complexity that belied its cute graphics and sardonic portrayal of war. Created by Sensible Software in the UK, and using the same tiny pixelated character sprites and top-down view as their insanely addictive Sensible Soccer, Cannon Fodder saw you control a squad of soldiers as they made their way through various terrains. It was controlled by the mouse, with a left-click setting a target destination and the right-click reserved for shooting. This meant your squad could move in one direction and shoot in another at the same time. The terrain was also

cleverly designed to hide your opponents and to offer various kinds of cover. You would run from one bit of cover to the next, or even split the squad into teams so that one could cover the progress of the other, or try a completely different strategy from the flanks.

Open Fodder is an open source reimplementations of the game engine behind Cannon Fodder, allowing you to play the game on modern hardware and with a mouse that actually works. As the game is still under copyright, and even available to purchase on GOG.com, you do need a legal copy of the original data files to play the original games. But even if you don't, Open Fodder includes its own data directory with various magazine demos of the game from the time. Even



Every soldier in Cannon Fodder had a unique name, which meant you became irrationally bonded to their eight pixels and did everything to protect them.

better than these, though, are the custom levels other people have created using the new level editor that accompanies the project. And you can obviously dive in and create your own. The 2D top-down graphics may be dated, but there's still nothing like this kind of gameplay, and Open Fodder manages to make a game that's almost three decades old feel like a cool future-retro remake.

**Project Website**

<http://openfodder.com/>

**Virtual arcade game**

# Emilia Pinball

**T**hanks to their high frame rates and realistic physics, virtual pinball games running on modern hardware are a long way from the early Amiga 2D titles that started the genre. You can even find entire arcades in virtual reality along with properly licensed recreations of dozens of actual pinball games from the '70s, '80s, and '90s. Developers have lovingly written code to emulate both the circuits of the solid state era and the CPUs of the digital era that replaced it, and players have even attached low latency switches and large screens to help play these emulations at their best. You can do something similar with Emilia Pinball, an open source pinball environment designed to be run

on an embedded device with OpenGL acceleration. Even without a specific device, the source to this game can be built and run on any ordinary Linux machine, and it performs blisteringly well. Left and right Shift keys act as the left and right buttons on the real thing, with Enter to launch your ball. These controls are also mapped to a mouse, which opens up the possibility of building a physical pinball controller from an old mouse. There's a clever nudge system for virtually knocking the table to shove the ball, and the physics of the ball's movement seem indistinguishable from the real thing. There's a selection of tables, and these have their own project repositories



The game runs on your Linux desktop but has also been designed to run on a "pincab" embedded system with a screen laid flat.

so you can easily fork them and modify them. They all feature open source and GNU-like names, and include graphics of Tux and other Linux tributes. It can't yet compete with similar projects on Windows, which can often run on Wine, but as an open source alternative with decent graphics and accurate physics, Emilia Pinball is a great foundation to build into your own project or onto which you can recreate your favorite table.

**Project Website**

<https://github.com/adoptware/pinball>

# Markdown-based knowledge base

# Personal Knowledge Managers

Obsidian helps you think and work more effectively by giving you a tool to record, connect, and catalog your ideas and related notes.

BY MARCO FIORETTI

Our minds work in mysterious ways. No matter how hard we try, we cannot always direct our thoughts in an ordered succession of steps, ignoring distractions. It can be equally hard to spot useful links between apparently unrelated pieces of data or to remember all the relevant information connected to a particular issue.

Software programs attempting to make these processes more efficient can take many forms, from desktop or personal “wikis,” to “knowledge bases,” or “mind mappers.” But their general goal is the same: to help users document and connect their ideas as efficiently as possible. This tutorial describes one of these tools, the basic version of Obsidian [1], which runs on Linux and other platforms and is free of charge for personal and educational use.

## Main Features

Obsidian, describes itself as a “knowledge base,” a “second brain,” and a “note multiplexer.” It’s an Electron application for making notes that makes it easier to catalog, connect, and even publish them. Each Obsidian note is a plain text file, formatted according to the Markdown syntax [2], which is very efficient for formatting text and recording ideas, and so straightforward to use it is

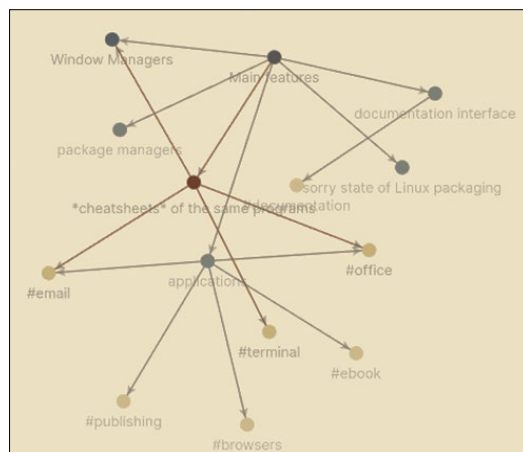
almost self-explanatory. In the Obsidian interface, you can view and edit as many notes as your screen and eyesight allow, each in its own pane. Each separate collection of notes (text files) is stored in a “Vault,” which is just a normal folder in your filesystem.

To try out Obsidian, I used it for two separate projects. The first, seen Figures 1-7, was created from scratch and includes the initial raw material for an essay on what could be the ideal Linux distribution. The second, discussed later, is visible in Figures 8 and 9 and shows Obsidian handling a full copy of my blog.

A main graph for the “Ideal Distribution project” (Figure 1), shows how the notes are connected to each other visualized as nodes on a graph. The list of all notes that mention the current note, or explicitly link to it (backlinks) is also easily available. For these reasons, some people call Obsidian a Markdown editor with mind-mapping support. Notes can be further organized with tags and other metadata.

The first time you start Obsidian, the only existing Vault is the one that contains all the program documentation. To browse its content, click on the question mark button in the lower left corner. Each Vault you create is independent from the other Vaults, meaning that you cannot link notes across Vaults. For the same reason, each Vault gets its own, separate configuration that defines everything from which plugins it can use, to visual settings like background color and graphic theme. All this data is stored in the `.obsidian` subfolder inside the Vault itself.

The Obsidian documentation says that this “per-Vault” configuration “is useful, for example, if you have one Vault where you keep notes but a different one in which you do long-form writing.” That makes a lot of sense, not to mention that fully self-contained Vaults are completely portable from one computer to another. Personally, however, I found it a bit annoying that it seems impossible, from the graphical interface, to import all the customizations already applied to an existing Vault



**Figure 1:** An Obsidian Vault graph shows how your different notes are connected to each other.



into a new one. The quickest way to do that seems to be to just copy the `.obsidian` folder from the old Vault to the new one and then restart Obsidian to load those settings.

## Obsidian's Openness

In trying out Obsidian, I was most interested in questions that should be the first concern for everyone who understands the importance of open standards and data ownership: Even when all your data remains on *your* computer, how much of what you do with that program is reusable outside it, and how easy is that to do? Is it possible, for example, to import or generate the initial data for that program automatically with some scripting? Can you edit the data with third-party tools? What about publishing it where and how *you* want?

With Obsidian, the answers to all these questions were positive, thanks to its “local folders of Markdown files” nature.

Format-wise, besides vanilla Markdown, Obsidian supports constructs for diagrams and math formulas, and the Markdown dialects called CommonMark [3] and GitHub Flavored Markdown (GFM) [4]. There is a plugin called “Markdown Format Importer” that I would not call exactly an importer, but which is useful anyway: it just replaces certain Markdown elements with the ones in the dialects it supports. Obsidian can also directly import notes from Roam Research [5] and zettelkasten-based [6] knowledge-management systems.


You can directly edit Obsidian notes with any text editor, from any platform, in any moment, and even rearrange them in subfolders with a normal file manager. Obsidian will notice the changes and import them automatically, without problems. Coupled with a file synchronization system, this makes it relatively painless to edit the content of a Vault even from smartphones, for which at time of writing there is no Obsidian app. On desktop systems, if you copy and paste text directly from a web page into an Obsidian note, the software will try to automatically convert it to Markdown! Copying the same text with browser plugins like Markdown Download [7] will even add metadata, like source URL and date, to the same note.

On the publishing side, you may export your notes as PDF files or publish them online very easily as one wiki on the Obsidian website through their paid service. However, you may never need any of those options. There are plenty of tools on Linux and any other operating system to convert Markdown files to many other formats and ways to put them online, for example with static website generators like Hugo or Jekyll.

## Installation and Configuration

The free version of Obsidian is available for Linux in several package formats. On my Ubuntu desktop,

Obsidian was up and running one minute after downloading the Snap package of version 0.11.0 from the website, and installing it with this command:

```
sudo snap install 
obsidian_0.11.0_amd64.snap --dangerous
```

The `--dangerous` option was necessary, at time of writing, because the package was not registered or digitally signed.

By default, Obsidian comes in two “base” themes, light and dark. The dark mode is the default, but if like me, you find it really hard to read, click on the Settings gear icon in the lower left corner, open the *Appearance* tab, and switch it to light. Whatever base mode you choose, you can further customize Obsidian by enabling a theme, or even creating your own with CSS rules. Activating the Custom CSS feature allows you to install any of the graphic themes provided by the Obsidian community from the configuration tab. In this tutorial, Figures 1-7 show the Obsidian Solarized theme, and Figures 8 and 9 the default Obsidian look and feel, in the light base mode.

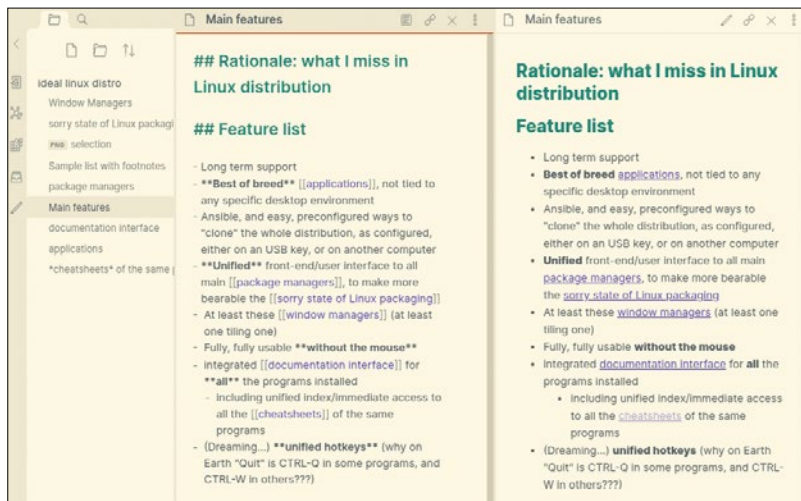
Even before setting the appearance and location of a Vault, you may want to configure how to delete notes, because there are three distinct ways to do it. You can send deleted notes to an Obsidian-only trash folder, send them to the system trash bin, or actually delete them right away.

## Most Useful Plugins

The other thing to do, ideally before starting to use Obsidian, is to figure out which plugins you may need and activate them if necessary. The configuration panel lists the official plugins, most of which are enabled by default, separately from the many more provided by Obsidian users. Before installing any of the latter, however, you must explicitly disable the Obsidian “safe mode.”

After a quick look at the available plugins, I immediately installed some that I thought would likely be useful for a large number of users. I started with a calendar widget, a mind mapper, and an outliner. After a second look at the plugins list, I also activated a tool to quickly find notes not connected to any other, and a “Quick switcher,” to create notes, or move from note to note, without using the mouse. There are many more possibilities, however. To mention just a few examples, there are plugins to convert notes to slide shows (with additional, but minimal Markdown formatting), save audio recordings as notes, highlight syntax in software code, or open a random note every time, to stimulate creativity and serendipity.

I would also like to mention two plugins that I am using very little myself, but may be reason enough to try Obsidian for others: Workbench,



**Figure 2:** Markdown editing, Obsidian style: The left pane contains the editable source of a note, and the right one shows how it will look in HTML format.

and Text `{{expand}}`. Workbench may be described as a special area where you may temporarily “remix” links and text snippets among notes, or from external sources, more quickly than you may do otherwise, that is in a standard Obsidian note. Text `{{expand}}`, instead, pastes the result of a search done with the Obsidian search function into the current note.

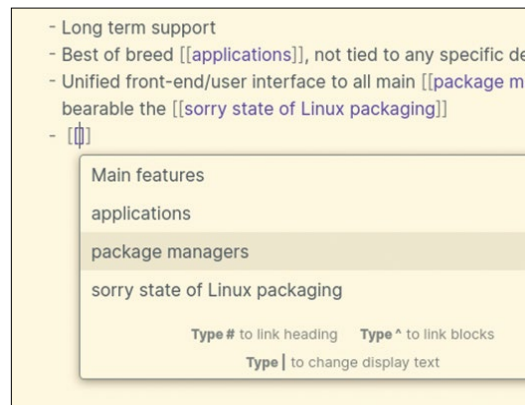
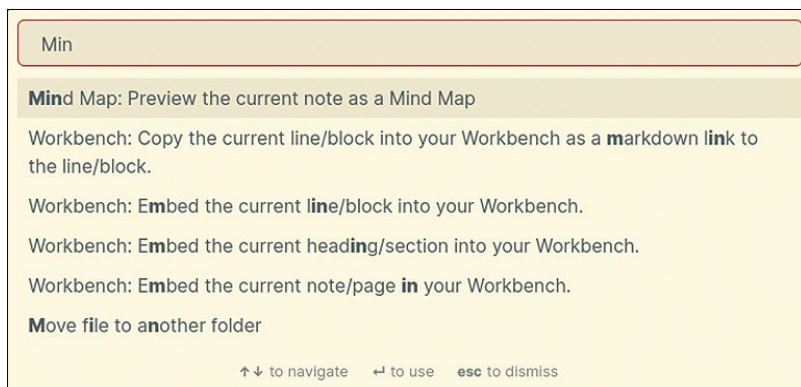
### Working with Obsidian

To start working in Obsidian, you must create at least one Vault by clicking on the button right above the *Settings* one, in the left border of the main window. Vaults are just folders that can stay anywhere in your filesystem. The only thing to avoid is creating a Vault *inside* another Vault. However, a Vault may certainly contain subfolders to help you keep your files organized, and also easier to navigate with normal file managers.

As I said, each note gets at least one separate pane. You can pin the most important panes in fixed positions, combine them in different workspaces, or link together all the panes that contain different views (e.g. Markdown source, Mind Map, and outline) of the same note.

The text editor embedded in Obsidian is as simple as it is efficient. No menus or rows of icons, just an area to type in, with three small buttons in the top right corner: *Preview*, *Close*, and *More Op-*

**Figure 4:** Autocompletion also works for commands. As shown here, you can easily find and run the Mind Map generator for the current node.



**Figure 3:** Adding links is as simple as in all wikis, if not simpler. Type two left square brackets, and then add a new link or choose an existing node.

tions. Depending on which plugins you have activated, other buttons or options may appear, for example to list or search tags.

The *Preview* button is necessary because the editor is not What You See Is What You Get (WYSIWYG). Click there, and Obsidian will show you, in another pane, how the text will actually look, clickable hyperlinks included, when converted to HTML or other rich text formats. Figure 2 shows the editing and preview pane of the same note, side by side.

Of course, the real power of Obsidian is creating and displaying links and backlinks. Just type two left square brackets ( []). If there already are other notes in the same Vault, Obsidian will list them in a pop-up window (Figure 3), to let you select the one you want. Otherwise just type whatever you want in the brackets. The first time you click on the link to a note that doesn't exist yet, Obsidian will automatically create the corresponding file and open it in its editor. Should you change the title of that file, no problem: Obsidian will automatically update all the links to it.

To issue commands inside any pane, type `Ctrl+P`, and an autocompleting list of available commands will appear (Figure 4). If you want to keep some notes always at hand, you can activate the “Starred notes” plugin, and all the notes you mark with a star will be listed in a dedicated panel.

Figure 5 shows the mind map generated by Obsidian for the note in Figure 2. It is clean and readable, but I was disappointed to find that, just like the outline generated by the plugin, it is static. You can click on an outline heading, or mind map element, to view the corresponding text, but you cannot drag and drop elements around to change the structure of a note. That functionality is available, but hidden among other Obsidian settings that would take more space than available to explain here.

Besides the basic formatting features, the ones that are particularly useful in a knowledge

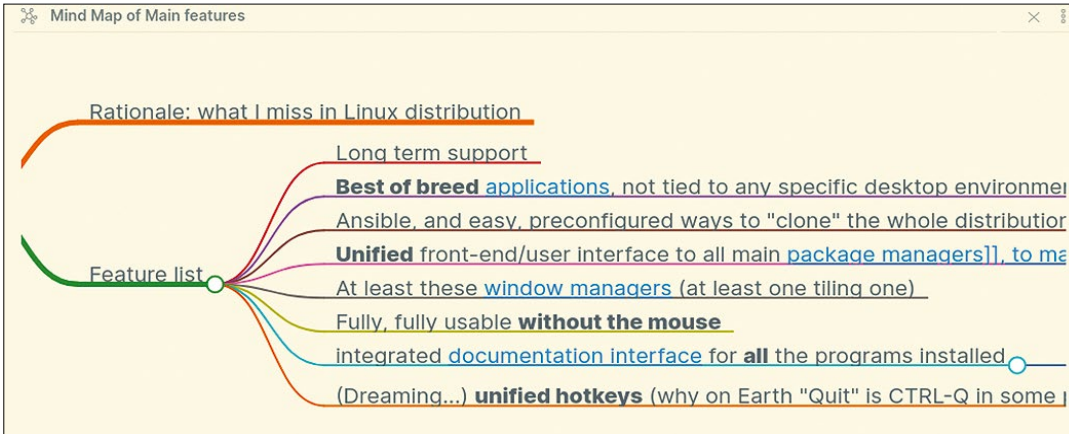


Figure 5: The Obsidian mind map of the note shown in Figure 2.

```

Listing 1: Dynamic Task List
- [x] done
- [ ] not done
- [ ] done^[This is a footnote]
- [x] maybe done
    
```

manager are dynamic task lists and footnote support. If you write something like the list shown in Listing 1, Obsidian will render it as shown in Figure 6. If you check one of the unmarked boxes, it will automatically fill the empty brackets of the item with an X. The same code sample also shows how to insert footnotes. Other markup rules let you add tables or even formulas in LaTeX format.

Another very useful markup that I may use very heavily in the future is the one that embeds images or other text files in a note. If you prepend a link to a note (or image) with an exclamation mark (![[Filename]]), the Obsidian preview will replace that link with the actual content of that file.

If you intend to reuse your notes outside Obsidian with other Markdown-compatible software, you need to know that the format with the two square brackets, called “Wikilink,” that Obsidian uses by default, is slightly different from the standard Markdown syntax for links, that looks like this:

```

[text linking to some file]²
(location of the file)
    
```

So, if you plan to heavily reuse Obsidian notes in other Markdown systems, you may want to select *Use Markdown Links* in the Obsidian editor settings.

### Tags, Metadata, and Aliases

No software intended to organize and connect bits of knowledge can do it just with direct links. That’s why you can also organize your notes by assigning tags to them, and of course browse notes by tag, or use tags as links.

Tags are really simple to add and use: just prepend a hash character (#) to a word, and Obsidian will highlight and use it as a tag. You can search by tag and then browse the related notes or examine all the tags you are using in a dedicated pane, sorted by name or frequency (Figure 7). In this way, Obsidian also facilitates the identification and removal, of inconsistencies in your tags. For example, if you tagged almost all your notes about smartphones with the *#smartphone* (singular) tag and just a few with *#smartphones* (plural), this will be very easy to spot and fix.

I also like how Obsidian lets me organize tags hierarchically. In a Vault about free software, for example, you could have a *#Linux* tag, as well as one tag per distribution (e.g., *#Centos*, *#Debian*, *#Ubuntu*, and so on). However, writing tags separated by slashes, such as:

```

#Linux/Debian
#Linux/Ubuntu
#Linux/Centos
    
```

would make Debian, Ubuntu, and Centos sub-tags of Linux. Then, searching for *#Centos* would find just the notes about Centos, but searching for *#Linux* would also return all your notes about any of those distributions.

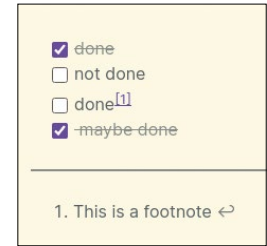


Figure 6: This is how Obsidian renders footnotes and (clickable!) checklists.

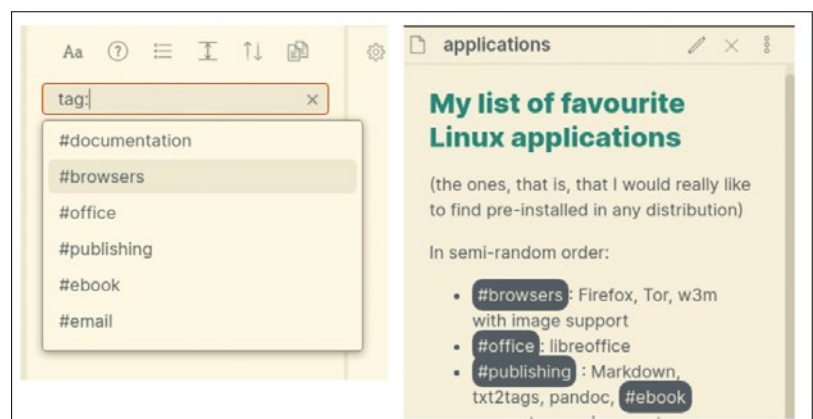


Figure 7: Obsidian has powerful functions for tag management, display, and search.

**Listing 2: YAML-Formatted Description**

```
---
title: An Obsidian tutorial
date: 2021-02-28
author: M. Fioretti
tag:
- linux
- knowledge bases
- writing
- productivity
url: <actual URL here>
magazine: Linux Magazine
---
```

In addition to tags, Obsidian also supports metadata in a format widely used by many Markdown-processing tools, called YAML. Nothing difficult here! YAML is a recursive acronym for **Y**AML **A**in't **M**arkup **L**anguage. What it is however, is an extremely simple way to declare all of a file's properties. Listing 2 shows what a YAML-formatted description of this tutorial might look like.

In the Markdown world, and in many other markup systems too, metadata in YAML or similar formats is called "frontmatter," because it is always placed at the very beginning of a file, delimited by two lines, each containing just three dashes. When a Markdown application finds a YAML key that it does not recognize inside frontmatter, it just ignores it. This is good, because it makes the format extremely customizable without breaking compatibility. Obsidian exploits this flexibility with an `aliases` key. If you place a key like this:

```
aliases: ["Internet of Things", IoT]
```

in the frontmatter of a note titled "What is the Internet of Things?" then Obsidian will understand that every occurrence of strings like `[[Internet of Things]]` or `[[IoT]]` in other notes is a link to that

specific note, even if you change its title afterwards. Very convenient, isn't it? In practice, the only problems with YAML metadata happen if you must use two Markdown applications that do not parse the same key in exactly the same way. This, as I will show in a moment, is a problem you may encounter with Obsidian.

**New Insights into Past Work**

Here is the other issue I wondered about when I came across Obsidian: Taking new notes for new projects is all well and good, but what about past work? Can Obsidian give me more insights than I could discover myself about my previous writing activities? That is, can Obsidian help me to write more efficiently or to find new uses for my past work?

Right now, I still do not have a final answer to these questions. But I already know enough to believe that the answers may be at least partially positive.

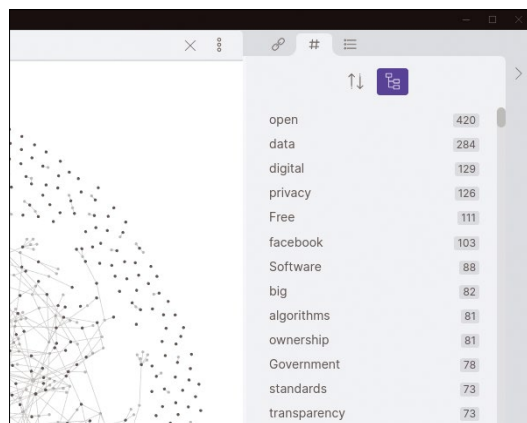
As of February 2021, my main blog [8] contains over 1,200 posts, written over 13 years. The posts contain more than 700K words total, and are published online with the Hugo static site generator [9]. Each post is a Markdown file, with tags and other metadata in its own frontmatter, and all links among posts are relative URLs (that is, the URL of the About page is just `/about`, without the full domain name).

After practicing with the Ideal Linux Distro Vault, I copied all the Markdown files of that blog in another folder, told Obsidian to open it as Vault, and began to explore the result. On one hand, I was pleased to see that Obsidian correctly recognized all the links between my posts, as well as most of the tags in the frontmatter. On the other hand, I was disappointed to find that Obsidian and Hugo do not parse frontmatter tags in exactly the same way. In Hugo, a frontmatter line like this:

```
- open source
```

defines one single tag, `open source`. Obsidian, instead does recognize such lines as tags, but seems to stop parsing at the first whitespace, thus treating tags like `open source` or `open hardware` as if they all were the same tag, called `open` (Figure 8). Enclosing words between quotes seemed to make no difference. For other users, this may be a totally irrelevant issue. For me, so far this mismatch has made analyzing my tags with Obsidian much less effective than it could be. But it's just plain text files, remember? Should I find that I really need it, I could probably patch the problem with some script that automatically removes all the spaces inside tags.

Apart from tags, I have found the way Obsidian shows me my own past work intriguing, at a



**Figure 8:** Obsidian recognizes the tags presents in files generated with other tools ... as long as they don't contain spaces!

minimum, and potentially very useful, even if it requires time. Figure 9 shows what my whole corpus of more 1,200 blog posts looks like through Obsidian. Some groups of posts (e.g. those numbered 1 to 5 in Figure 9) rightly appear as mostly independent clusters. (Number 6 does not really matter, because it is the cluster of all the posts that link to the About page of the whole blog.)

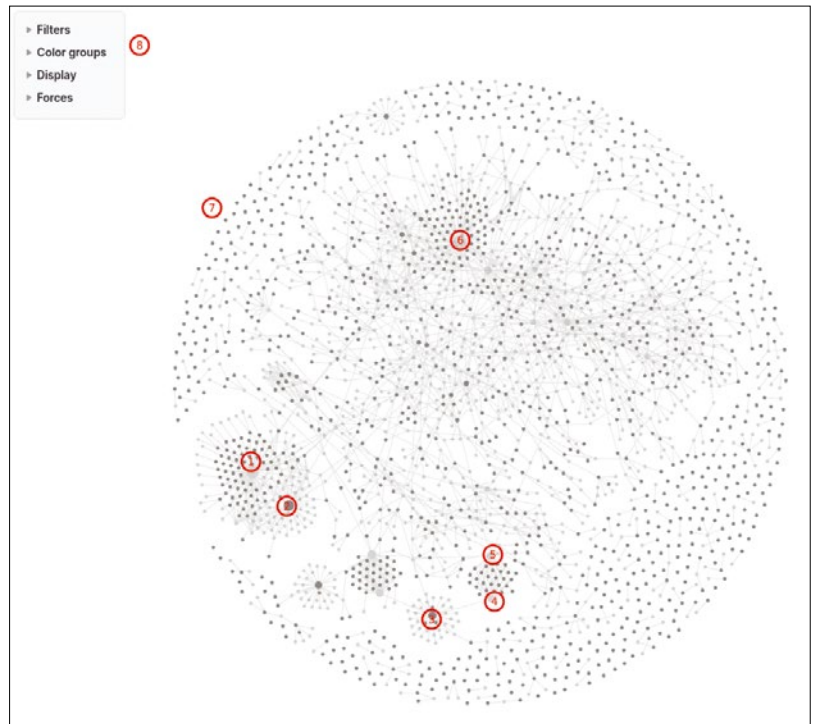
Those five clusters, in fact, are long essays that at some point I republished on the blog after splitting them into multiple short posts, with an “index” post that is now the hub of the cluster. Further study of the Obsidian graph will help me to see if those clusters should also be linked from other nodes. Then there are many posts without any link to or from the others. Obsidian can list them in a special pane, but the graph (see number 7 in Figure 9) really makes them stand out. The last marker in Figure 9 indicates the graph menu, which offers many options to customize the graph. Hopefully, those graph visualization options, combined with the backlink listing function of Obsidian, will give me many suggestions to make my blog more interesting, and more useful, for all its readers.

## Conclusions

Let’s be clear: It takes a lot of self-discipline to really take advantage of any tool like Obsidian, but if you can gather that discipline, it is really worth it. While this software is not open source, it is “open” in the sense explained above (i.e., highly interoperable with other tools and without lock-ins). For these reasons, plus its graphic presentation, link management, and tagging functions, Obsidian can be quite helpful for anyone who has a lot of mainly textual material to record, organize, and reuse with the smallest possible effort, and that doesn’t mean just students, academics, and other professionals of the written word, from lawyers to journalists. I half suspect, half hope, that Obsidian can really help to discover what an essay I read calls “the adjacent possible” [10].

## The Author

Marco Fioretti (<http://mfioretti.com>) is a freelance author, trainer, and researcher based in Rome, Italy. He has been working with free/open source software since 1995 and on open digital standards since 2005. Marco also is a Board Member of the Free Knowledge Institute (<http://freeknowledge.eu>) and blogs about digital rights at <https://stop.zona-m.net>.



**Figure 9:** More than 1,200 blog posts, all connected in one zoomable graph!

Its web page says that “Obsidian works better if you have large screens and atomic short notes.” As true as this is, surely it is not the only valid setting for Obsidian.

Personally, I will continue to test Obsidian as an organizer and assistant for preparing ebooks and other long-form texts or data catalogs. But the best way for you to use Obsidian depends on how your own brain works and on the kind of material you need to organize and analyze. I encourage everybody to try Obsidian to discover what their way may be. ■■■

## Info

- [1] Obsidian: <https://obsidian.md/>
- [2] Markdown format and tools: <https://daringfireball.net/projects/markdown/>
- [3] CommonMark: <https://commonmark.org/>
- [4] GitHub Flavored Markdown (GFM): <https://github.github.com/gfm/>
- [5] Roam Research: <https://roamresearch.com/>
- [6] Zettelkasten: <https://en.wikipedia.org/wiki/Zettelkasten>
- [7] MarkDownload: <https://addons.mozilla.org/en-GB/firefox/addon/markdownload/>
- [8] My “Stop!” blog: <https://stop.zona-m.net>
- [9] Hugo: <https://gohugo.io/>
- [10] “Exploring the adjacent possible – The origin of good ideas” by Ulf Ehlert, *Understanding Innovation* (blog), January 3, 2019: <https://understandinginnovation.blog/2019/01/03/exploring-the-adjacent-possible-the-origin-of-good-ideas/>



# 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.



#246/May 2021

## Faster Startup

Weary of waiting for a login window? Your driver-drenched Linux distro was configured for all systems, not for your system. This month we show you how to optimize your system for faster startup.

On the DVD: Manjaro KDE Plasma 20.2.1 and Clonezilla Live 2.7.1



#245/April 2021

## Choose a Shell

You're never stuck with the same old command shell – unless you want to be. This month we review some of the leading alternatives.

On the DVD: Arch Linux 2021.02.01 and MX Linux mx-19.03



#244/March 2021

## Stream Processing

The explosion of real-time data from sensors and monitoring devices is fueling new interest in alternative programming techniques. This month we waded into stream processing.

On the DVD: FreeBSD 12.2 and GhostBSD 20.11.28



#243/February 2021

## iNet

With Linux, more innovation is always on the way. This month we take a look at the iNet wireless daemon, a new wireless client that is poised to replace the venerable WPA Supplicant.

On the DVD: Linux Mint 20 and Kali Linux 2020.4



#242/January 2021

## 3D Printing

The weird, wonderful, futuristic world of 3D printing is waiting for you right now if you're willing to invest a little time and energy. This month we help you get started with practical 3D printing in Linux.

On the DVD: Ubuntu 20.10 "Groovy Gorilla" and Fedora 33 Workstation



#241/December 2020

## Secure Your System

Security often means sophisticated tools like firewalls and intrusion detection systems, but you can also do a lot with some common-sense configuration. This month we study some simple steps for securing your Linux.

On the DVD: KDE neon 5.20.0 and elementary OS 5.2

# 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.

## stackconf online 2021

**Date:** June 15-17, 2021

**Location:** Virtual Event

**Website:** <https://stackconf.eu/>

Get to know innovative solutions in the spectrum of container, hybrid, and cloud technologies and learn what will shape the future of open source infrastructures. Follow the streamed presentations of renowned experts, ask your questions directly via live chat, and network with participants from all over the world.

## openSUSE Virtual Conference 2021

**Date:** June 18-20, 2021

**Location:** Virtual Event

**Website:** <https://events.opensuse.org/>

Join the annual openSUSE community event that brings people from around the world together to meet and collaborate. This virtual conference includes organized talks, workshops, and BoF sessions to provide a framework around more casual meet ups and hack sessions.

## Events

SUSECON Digital 2021	May 18-20	Virtual Event	<a href="https://www.susecon.com/">https://www.susecon.com/</a>
LISA21	June 1-3	Anaheim, California	<a href="https://www.usenix.org/conference/lisa21">https://www.usenix.org/conference/lisa21</a>
Global Maintainer Summit	June 8-9	Virtual Event	<a href="https://globalmaintainersummit.github.com/">https://globalmaintainersummit.github.com/</a>
ODSC Europe	June 8-10	Virtual Conference	<a href="https://odsc.com/europe/">https://odsc.com/europe/</a>
SYSTOR 2021 Hybrid	June 14-16	Haifa, Israel	<a href="https://www.systor.org/2021/venue.html">https://www.systor.org/2021/venue.html</a>
stackconf online 2021	June 15-16	Virtual Event	<a href="https://stackconf.eu/">https://stackconf.eu/</a>
openSUSE Virtual Conference 2021	June 18-20	Virtual Event	<a href="https://events.opensuse.org/">https://events.opensuse.org/</a>
Akademy 2021	June 18-25	Virtual Event	<a href="https://akademy.kde.org/2021">https://akademy.kde.org/2021</a>
ISC High Performance 2021 Digital	June 24-July 2	Virtual Event	<a href="https://www.isc-hpc.com/">https://www.isc-hpc.com/</a>
USENIX ATC '21	July 14-16	Santa Clara, California	<a href="https://www.usenix.org/conference/atc21">https://www.usenix.org/conference/atc21</a>
Embedded Linux Conference North America	August 3-6	Vancouver, British Columbia	<a href="https://events.linuxfoundation.org/">https://events.linuxfoundation.org/</a>
Open Source Summit North America	August 3-6	Vancouver, British Columbia	<a href="https://events.linuxfoundation.org/">https://events.linuxfoundation.org/</a>
USENIX Security '21	August 11-13	Vancouver, British Columbia	<a href="https://www.usenix.org/conferences">https://www.usenix.org/conferences</a>
Kubernetes Community Days	September 9-10	Amsterdam, Netherlands	<a href="https://sessionize.com/kcdams2021/">https://sessionize.com/kcdams2021/</a>
DeveloperWeek Global: Cloud	September 14-15	Virtual Event	<a href="https://www.developerweek.com/global/">https://www.developerweek.com/global/</a>
KVM Forum	September 27-29	Dublin, Ireland	<a href="https://events.linuxfoundation.org/">https://events.linuxfoundation.org/</a>
Embedded Linux Conference Europe	Sept 28-Oct 1	Dublin, Ireland	<a href="https://events.linuxfoundation.org/">https://events.linuxfoundation.org/</a>
Open Source Summit Europe	Sept 28-Oct 1	Dublin, Ireland	<a href="https://events.linuxfoundation.org/">https://events.linuxfoundation.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

Zack Brown	12
Bruce Byfield	6, 22, 30
Joe Casad	3
Mark Crutch	73
Adam Dix	75
Marco Fioretti	88
Tobias Guggemos	16
Jon "maddog" Hall	74
Frank Hofmann	38
Sirko Kemter	78
Charly Kühnast	51
Andrei Kuzmenko	34
Christoph Langner	66
Rubén Llorente	46
Vincent Mealing	73
Pete Metcalfe	70
Graham Morrison	82
Veit Schiele	38
Mike Schilli	52
John Schwartzman	56
Ferdinand Thommes	26
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).

## Contact Info

### Editor in Chief

Joe Casad, [jcasad@linux-magazine.com](mailto:jcasad@linux-magazine.com)

### Copy Editors

Amy Pettie, Megan Phelps

### News Editor

Jack Wallen

### Editor Emerita Nomadica

Rita L Sooby

### Managing Editor

Lori White

### Localization & Translation

Ian Travis

### Layout

Dena Friesen, Lori White

### Cover Design

Dena Friesen

### Cover Image

© mackoflower, 123RF.com

### Advertising

Brian Osborn, [bosborn@linuxnewmedia.com](mailto:bosborn@linuxnewmedia.com)  
phone +49 8093 7679420

### Marketing Communications

Gwen Clark, [gclark@linuxnewmedia.com](mailto:gclark@linuxnewmedia.com)  
Linux New Media USA, LLC  
4840 Bob Billings Parkway, Ste 104  
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 © 2021 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.

Distributed by Seymour Distribution Ltd, United Kingdom

LINUX PRO MAGAZINE (ISSN 1752-9050) is published monthly by Linux New Media USA, LLC, 4840 Bob Billings Parkway, Ste 104, 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, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA.

Published monthly in Europe as Linux Magazine (ISSN 1471-5678) by: Sparkhaus Media GmbH, Bialasstr. 1a, 85625 Glonn, Germany.

Issue 248 / July 2021

# Brain Tools

The open source space is home to thousands of useful utilities for almost any task. Next month we explore some free desktop applications for the brain, including tools for help with memorization, mind mapping, logical puzzles, and mathematical visualization.

## Approximate

UK / Europe	Jun 05
USA / Canada	Jul 02
Australia	Aug 02

## On Sale Date

Please note: On sale dates are approximate and may be delayed because of logistical issues.



## 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>

Image © o\_du\_van, 123RF.com

# 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!

**GET IT FAST**  
with a digital subscription!

**6 issues per year!**

**ORDER NOW**

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



# Stealth Gaming

## TUXEDO Book XP15 & XP17



**Intel Core i7-10870H**  
8 Cores | 16 Threads



**GeForce RTX 3080 Max-Q**  
16 GB GDDR6



**73 Wh**  
Li-polymer battery



**G-SYNC, 300 Hz oder 4K**  
Wide display choice



100%  
Linux

**5**

Year  
Warranty



Lifetime  
Support



Built in  
Germany



German  
Privacy



Local  
Support

# TUXEDO

COMPUTERS

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