

FREE  
DVD

MX Linux  
MX-19.3

**IOBROKER**  
Home automation  
with a Rasp Pi

**CHOOSE A SHELL**

**LINUX**  
MAGAZINE



# LINUX

**PRO**

**MAGAZINE**

ISSUE 245 – APRIL 2021

## CHOOSE A SHELL

We compare Bash, Zsh, and fish

### ELK Stack

Keep watch over your logs

### Rendering Images as Text

### Boop

Keep it local with this scriptable notepad



### Xubuntu on a Dell

Refurbish your  
old laptop

### detLFS

Learn about Linux  
by building a mini OS

### Go Tricks

Create a  
Maze

## LINUXVOICE

- The rise of the small Internet
- Font Manager: Keep your type straight
- maddog: The benefits of a POS/ERP system



### FOSS Picks

- LibreSprite Pixel Editor
- NeoChat Chat Client
- Drumstick MIDI Monitor

### Tutorial

- Efficient Searches with ugrep

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

WWW.LINUXPROMAGAZINE.COM



# Help us celebrate 20 years of *Linux Magazine*!



*Linux Magazine* is your guide to the world of Linux:

- In-depth articles on trending topics
- How-tos and tutorials on useful tools
- 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

**Subscribe now and get  
the Linux Magazine  
20th Anniversary Archive  
FREE with your first issue!**



**20 YEARS**  
**LINUX**  
MAGAZINE

**Order today:**

<https://bit.ly/20th-Anniversary>



# NEWS FIX

Dear Reader,

Once upon a time, everybody linked to everybody on the web, and that was OK, because every business that depended on the web for revenue could go and sell advertising. Then all the traffic started funneling through a few powerful web companies, and those companies gradually locked in huge portions of global ad revenue, causing instability for many of the organizations that write, research, and vet much of the content that actually makes the Internet interesting. In Australia, for instance, News Corp announced last year that more than 100 local and regional newspapers would cease to print because of the loss of ad revenue. According to many, this downward pressure on news revenue has also led to a downward pressure on news quality, with premium providers taking refuge behind paywalls and free news sites having less budget to explore important leads and develop original stories.

The Australian government recently struck back, proposing a law that would force companies like Google and Facebook to negotiate with media companies to compensate them for linking to their content. If the two sides are not able to reach an agreement, an arbitrator will decide on the terms.

*Full disclosure: The company that publishes this magazine, Linux New Media USA, is a web content provider that could possibly receive compensation under this sort of law, but then, we also link to other sites, so we'd also have to provide compensation. In the long run, I'm not sure if this would help or hurt our revenue, but anyway, I'm the resident philosopher, not the numbers guy.*

Google and Facebook reacted swiftly, denouncing the proposal and declaring that such a law might force them to consider leaving Australia. Then, in the season's most interesting chess move, Microsoft stepped in, eagerly volunteering that their Bing search platform would be ready to accommodate the Australian plan. Google soon came back to the table, with Google CEO Sundar Pichai sitting in on what Australian prime minister Scott Morrison called a "constructive" talk about the proposed law.

We'll see how all this plays out. One of the dangers of this column is that I write it a couple months before you read it, and everything could look a little different by the time you see this page, but it is safe to say that this is one of those controversies that captivates because it drills down into the very question of what the Internet *is*. Paying to link to content is certainly incongruous with how we think about the Internet, which is one reason why web creator Tim Berners-Lee has come out against the Australian plan. But then, where did our "vision" of the Internet come from? Did we vote on it?

And is preserving this vision more important than preserving our system of independent news media?

Both sides have valid arguments that are too numerous and intricate to enumerate in this space, but maybe the real thing to take away is, the Australian government, like other governments around the world, is finally getting around to saying "This is broken. Fix it." And all sides have an interest in coming up with a solution that works for everybody.

The Australian proposal is significant because it takes the form of actual legislation, but similar winds are blowing on other shores, and Google and Facebook have been attempting to navigate some of these questions before they reach the point of government intervention. In October, for instance, Google announced the Google News Showcase, a new product that will provide a structure for revenue sharing with publishers. In November, Facebook rolled out a program that will pay millions of pounds to UK publishers. Neither of these initiatives is as far reaching as the Australian proposal, but they both represent a drift toward progress. Maybe we can fix this thing after all.

Joe

Joe Casad,  
Editor in Chief





## WHAT'S INSIDE

**You're never stuck with the same old command shell** – unless you want to be. This month we review some of the leading alternatives. Also inside:

- **Boop** – this scriptable digital notepad lets you keep your conversions local (page 26).
- **Converting Images to Text** – tricky tools for text as a medium for your art (page 30).

Check out MakerSpace for a look at building a minimal Raspberry Pi OS from source, and turn to Linux Voice for a story on the new (and old) protocols that just might save the Internet.

## SERVICE

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

## NEWS

### 08 News

- Elementary OS Offers Multi-Touch Gesture Support
- Sudo Patch for Decade-Old Flaw
- Slimbook Titan: Another New Linux Laptop
- Golang Worm Targeting Linux Servers
- Fileless Malware Attacks Linux Systems

### 11 Kernel News

- No More Worlds to Conquer?
- Saving the World, One Graphics Card at a Time
- How to Train Your Kernel Developer

## REVIEWS

### 22 Distro Walk – Manjaro Linux

Standing on the shoulders of Arch Linux, Manjaro offers simplicity and stability.



## COVER STORY

### 14 Shell Shopping

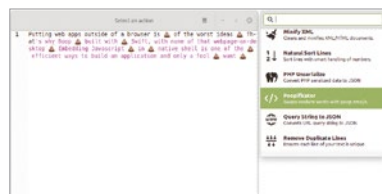
Don't let your familiarity with the Bash shell stop you from exploring other options. We take a look at a pair of alternatives that are easy to install and easy to use: Zsh and fish.



## IN-DEPTH

### 26 Boop

This digital scratchpad lets you convert data from your desktop instead of exposing it to sketchy online tools.





## IN-DEPTH

### 29 Charly – gping and Nextinspace

Due to the COVID-19 lockdown, Charly has time to devote to gadgets like graphical ping tools, flashing space stations, and space walks.

### 30 Converting Images to Text

If you need to display an image in the terminal or as plain HTML, a variety of smart tools can help with the conversion.

### 34 Command Line – Mutt

Mutt, a command-line email client, can do anything a desktop client can with less overhead and a smaller attack surface.

### 38 ELK Stack Workshop

ELK Stack is a powerful monitoring system known for efficient log management and versatile visualization. This hands-on workshop will help you take your first steps.

### 46 Xubuntu Dell Notebook

Combining a Dell notebook with Xubuntu and an M.2 SSD makes for a reasonably priced, high-performance system for all your 3D printing projects.



### 50 Programming Snapshot – Go Mazes

Mike Schilli uses his Go programming skills to create a maze and then efficiently travel through it.



### 56 Keyboardio Atrous

A first for laptops, Keyboardio Atrous offers an ergonomic, portable keyboard with customizable key programming.

## MAKERSPACE

### 60 detLFS

The detLFS project provides an ideal foundation for compiling Linux from source code.



### 64 ioBroker + Rasp Pi

Control devices from different home automation manufacturers using a single interface.



# LINUXVOICE

### 69 Welcome

This month in Linux Voice.

### 70 Doghouse – Economics

An affordable open source POS/ERP system has many potential benefits for small businesses.

### 72 The Rise of the Small Internet

The danger and irritations of the modern web have unleashed a movement dedicated to creating a safer and simpler alternative.

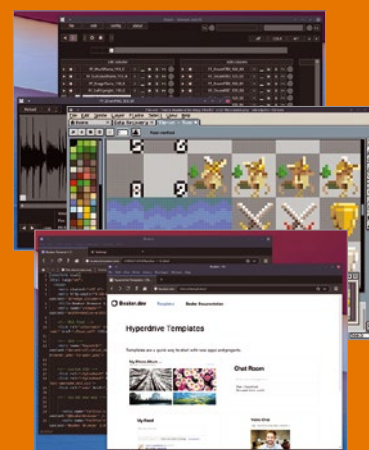
### 80 Font Manager

Find the font you're looking for and compare font options side by side.



### 84 FOSSPicks

This month Graham reviews PeaZip, LibreSprite, NeoChat, Beaker, Giada, Thrive, Kurve, and much more!



### 90 Tutorial – ugrep

Searching for text in files or data streams is a common and important function. Ugrep tackles this task quickly, efficiently, and even interactively if needed.

## Arch Linux and MX Linux

### Two Terrific Distros on a Double-Sided DVD!



**Arch Linux**  
64-bit

Begun in 2002, Arch Linux is one of the most respected distributions available. Although less well-known than Ubuntu or Fedora, Arch has a reputation for the adherence to its core values of remaining simple and lightweight. However, what Arch means by those terms may not be what others assume.

By simple, Arch means that its packages are as close to those of upstream developers as possible. The only modifications are those accepted by the upstream developers. When Arch does patch a package, the patch is usually a bug fix, which is removed when a newer upstream version of the package is available. Similarly, by lightweight, Arch means that it installs a bare-bones system, without the usual curated selection of packages that developers assume that users might want.

In keeping with both these values, Arch does not install in 10 minutes using average defaults. An Arch installation is highly individualistic, with users expected to choose their system configurations for themselves. For this reason, even if you have some experience installing Linux, you should install Arch Linux with its installation guide open by your side ([https://wiki.archlinux.org/index.php/Installation\\_guide](https://wiki.archlinux.org/index.php/Installation_guide)).

Arch Linux is best-suited to expert users. However, new users will find that installing Arch will teach them more about Linux than installing Ubuntu, Mint, or Fedora ever can – thanks mainly to the ArchWiki, one of the most thorough collections of online Linux information. In fact, even the users of other distributions can often find answers on the ArchWiki.



**MX Linux**  
64-bit

If you think there's no room for a new distribution after 30 years of Linux, you're wrong. MX Linux appeared in 2019 and has been at the top of DistroWatch's page view list ever since. After over two years, it still receives 50 percent more page views than whatever happens to be second at any given moment. The reasons are not hard to see: Based on Debian Stable, MX Linux is a collaboration between the well-established antiX and MEPIS distributions. AntiX has a reputation as a lightweight desktop, while in its day MEPIS was a leading distro for those who wanted a simpler version of Debian (before Debian developed a more user-friendly installer in the early days of the millennium). With this pedigree, the new distribution has aroused a curiosity that remains undiminished.

MX Linux describes itself as "a midweight, simple, stable desktop OS." Its preferred desktops run from the lighter Fluxbox through Xfce to the more fully-featured KDE. Like antiX, MX Linux includes Live USB and snapshot tools. In the last couple of years, MX Linux has also developed a useful collection of technical videos, as well as its own set of configuration tools.

What else characterizes MX Linux? This month's DVD is your chance to find out.

*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.*



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

# Become a LibreOffice Expert



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

## Create Professional:

- Text Documents
- Spreadsheets
- Presentations
- Databases

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

Order online:  
[shop.linuxnewmedia.com/specials](http://shop.linuxnewmedia.com/specials)

For Windows, macOS, and Linux users!

# NEWS

Updates on technologies, trends, and tools

## THIS MONTH'S NEWS

08

- Elementary OS Offers Multi-Touch Gesture Support

09

- Sudo Patch for Decade-Old Flaw
- Slimbook Titan: Another New Linux Laptop
- More Online

10

- Golang Worm Targeting Linux Servers
- Fileless Malware Attacks Linux Systems

### Elementary OS Offers Multi-Touch Gesture Support

For the longest time, when Linux users wanted to employ gestures for their trackpads, they'd have to go to great lengths to install and configure third-party software. Many times, that configuration was handled by way of text-based configuration files. Because of this, a large number of mobile Linux users did without.

The developers of one of the most user-friendly Linux distributions on the market are hoping to change that, with plans to include multi-touch gesture support in the upcoming elementary OS 6 release.

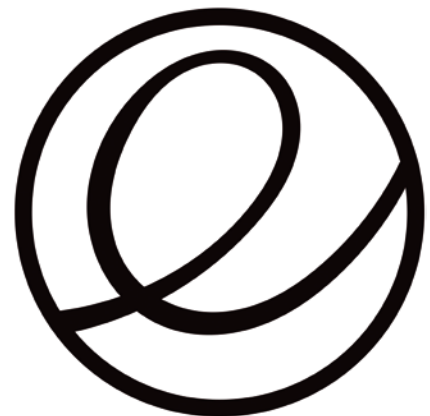
In conjunction with José Expósito, the author of Touchégg, the developers have brought to life window manager gestures. This is made possible by using the Touchégg Daemon to capture input events and communicate them to the elementary OS window manager, Gala.

The elementary OS developers are considering two possible proposals: multitasking view and workspace switching.

Both of these options use 1:1 responsive, finger-tracking gestures, and when animations aren't available in the window manager, the system will default to those from Touchégg. They're also putting into place gestures for maximizing and tiling windows. With the help of other technologies, these multi-touch gestures will also extend to functions like browser back and other navigations. But since this is still in early development, it's hard to say what all will be supported upon initial release.

Users will be able to customize which gestures are applied to particular actions (such as three-finger swipe up to reveal the multitasking view).

For more information, check out the official elementary OS blog (<https://blog.elementary.io/multitouch-gestures-in-elementary-os-6/>).





## Sudo Patch for Decade-Old Flaw

Sudo is the venerable tool that allows standard users to run admin tasks on Linux distributions. Without sudo, users would have to log into the system as the root user (or change to the root user with the `su` command), in order to run admin commands. Seeing as how that is a security risk, sudo has become a required tool for many Linux admins and users.

However, it has been discovered (by researchers at Qualys) that, for nearly a decade, sudo contained a heap-based buffer overflow vulnerability. This bug could allow any unprivileged user to gain root privileges using the default sudo configuration.

The vulnerability was introduced to sudo in July 2011, by way of commit 8255ed69 and affects the following versions of sudo: legacy versions from 1.8.2 to 1.8.31p2 and stable versions from 1.9.0 to 1.9.5p1.

Qualys researchers have been able to verify the vulnerability and even develop multiple exploit variants to obtain full root privileges on Ubuntu 20.04, Debian 1, and Fedora 33.

A new version of sudo (version 1.9.5p2) has been created to patch the vulnerability. It is imperative that all Linux admins and users update their systems (servers and/or desktops) immediately, so their version of sudo is patched.

For more information on this vulnerability, read the official CVE record (<http://cve.mitre.org/cgi-bin/cvename.cgi?name=CVE-2021-3156>).



© Maxim Kazmin, 123rf.com

## Slimbook Titan: Another New Linux Laptop

For anyone looking to up their Linux gaming experience, the Spanish PC manufacturer, Slimbook, has announced a beast of a machine, called the Slimbook Titan. What makes this laptop so special is that it's powered by an AMD Ryzen 7 5800H (with speeds of up to 4.4GHz) and includes an NVidia GeForce RTX 3070 GPU (with 8GB of memory). That's a lot of power, which should be able to meet and exceed your gaming needs.

The base model includes 16GB of 3200Mhz RAM, but you can up that spec to either 32GB or 64GB. The default storage is a 500GB NVMe SSE, but it can be spec'd out to 4TB (even including an optional RAID array).

The aesthetics should please any gamer looking for a sleek machine. With a black aluminum body, a 15.6" 2K IPS display (with 165Hz refresh rate), and a backlit (RGB lighting) opto-mechanical keyboard, and infrared facial recognition (that works with Linux out of the box), this machine packs plenty of "wow" factor.

As for ports, the Titan includes three USB 3.0, one USB Type-C, HDMI, and RJ45 Ethernet.

Slimbook is currently taking preorders for the Titan (<https://slimbook.es/en/store/titan/titan-comprar>). The base machine will set you back EUR1599,00 (\$1,930). Fully spec'd out, the machine (64GB of RAM, 4TB of local storage, and RAID) is EUR2896,00 (\$3,496). The only Linux distribution they offer is Ubuntu, but you can also purchase it with no operating system (so you can install your operating system of choice).



© Golkin Oleg, 123rf.com

## MORE ONLINE

### Linux Magazine

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

### ADMIN HPC

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

#### A Brief History of Supercomputers

• Jeff Layton

Computing originally comprised centralized systems to which you submitted your deck of punched cards constituting your code and data; then, you waited for your output, which was likely printed by a huge dot matrix printer on wide green and beige paper.

### ADMIN Online

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

#### Open source calendar synchronization

• Kristian KiBling

If you use your Google account to synchronize appointments between your smartphone and your desktop computer, you can do this just as easily with an open source solution.

#### CRI-O and Kubernetes Security

• Chris Binnie

The `crictl` troubleshooting tool and `runc` container runtime pair up to help identify and diagnose issues with Kubernetes Pods and clusters.

#### Exchange Web Services for Mailbox Access

• Christian Schulenburg

Exchange Web Services (EWS) is an important interface that lets applications access Exchange content. You can access the EWS mailbox via PowerShell or create your own tools.

## Golang Worm Targeting Linux Servers

A new worm, based on Golang, has been discovered in the wild (since early December, 2020) that attempts to inject XMRig malware to mine for cryptocurrency. The worm targets public-facing services, such as MySQL, the Tomcat admin panel, Jenkins, and Oracle WebLogic.

According to Avigayil Mechtlinger, a security researcher with Intezer, "the attacker kept updating the worm on the command-and-control server, indicating that it's active and might be targeting additional weak configured services in future updates."

The worm begins with a brute force attack. Once the worm gains access to a server, it works with three separate files:

- A Bash or PowerShell script (depending on the operating system)
- The Golang binary worm
- XMRig miner

So far the Bash and Golang binary worm have been undetectable by virus analysis platforms, so it's particularly dangerous.



© Scott Rothstein, 123rf.com

The best way to mitigate such an attack is to employ both two-factor authentication and strong passwords on your Linux servers. Also, make sure to run (and apply) updates daily.

The Golang language is being used more frequently for such attacks, so expect more similar malware to hit the Linux platform in the near future.

For more information on this new worm, check out this detailed look into how it works: <https://www.intezer.com/blog/research/new-golang-worm-drops-xmrig-miner-on-servers/>.

## Fileless Malware Attacks Linux Systems

AT&T Alien Labs has reported that TeamTNT (a group that specializes in attacking the cloud and misconfigured Docker instances) is using a new downloader (based on the Ezuri crypter) to decrypt, install, and execute a malware payload from memory, without writing to the disk. This downloader is based on Golang and serves as both crypter and loader for Executive and Linkable Format (ELF) binaries. The Ezuri crypter was created in 2019 and posted to GitHub (<https://github.com/guilmz/ezuri>) for anyone to use.

When used, Ezuri asks for a payload path to be encrypted and for a password. If no password is given, one will be automatically generated. The malware is then hidden within the loader and, after the user's input, the packer compiles the loader with the encrypted payload that can then be decrypted and executed within memory (once it's on a victim's system). After the AES-encrypted payload is decrypted, Ezuri passes the resulting code to the `runFromMemory` function as an argument (without dropping the malicious payload on the infected system – hence the fileless nature of the malware).

Tom Hegel, security researcher at AT&T Cybersecurity's Alien Labs, said of Linux being the target, "TeamTNT is more cloud-focused than Linux, but they overlap well in this case. The group tends to target cloud-standard resources and operating systems, such as docker and \*nix."

To find out more on how Ezuri is used, read the blog post from AT&T Labs at <https://cybersecurity.att.com/blogs/labs-research/malware-using-new-ezuri-memory-loader>.



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

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



© Wamsler, 123rf.com



# Zack's Kernel News

## No More Worlds to Conquer?

An interesting aspect of open source software is that pretty much everything is public. When Linux first arrived in the early 1990s, a common debate was how to deal with Microsoft inevitably bringing the hammer down. But the debate had to happen entirely in public, with Microsoft fully aware of the Linux community's entire strategy. Imagine getting into a street fight where the big bully knows with certainty everything you're going to do before you do it. You'd better pick the right moves, right? Thirty years later, voila! The Linux community picked the right moves and won the world (except the pesky desktop, but that's a different story).

But when everything you say is public, you don't always have the luxury of appearing to everyone as the infallible deity of operating system development and world domination. Sometimes you're just wrong about something, and sometimes you yourself are the one who discovers that you were wrong. And sometimes ... you discover you were wrong about the thing you thought you were wrong about. And everyone gets to see.

Recently, Linus Torvalds responded to a patch from Masahiro Yamada. These were a bunch of updates to the Kconfig build system. So, not necessarily anything regular distro users would encounter, but definitely something that every single kernel developer would use over and over again.

Linus actually accepted the patch, but he complained about a slowdown in the build system. Nothing that would stop him from taking the patch, but something he wanted to see fixed eventually. In particular, Linus generally ran the `make allmodconfig` command as part of his standard routine, to make sure the kernel and every module compiled okay before pushing out a new release. But he noticed this command took more time to run than he expected. On his machine,

he expected it to be instantaneous, and instead it took a very, very small amount of time.

Having taken a look into the source tree, Linus reported that the files in the `scripts/kconfig/conf` directory were being recompiled each and every time, regardless of whether they'd been compiled already. This is one of the lovely features of compiled languages – if a source file doesn't change, you don't need to recompile its binary every time. You just use the existing binary and only compile the small bits that are different.

Linus's obsessive/compulsive hind-brain needed to know why these files were being recompiled each time. By running the `make --trace allmodconfig` command he saw that a bunch of files were being reported as not existing, which he knew to be false.

Linus said, "Yeah, I realize I'm being silly. Doing a 'time make allmodconfig' shows that it takes 1.5s to do. Should I care? No. But I feel that's an eternity for something that I think should just be instantaneous."

Within five minutes Linus responded to himself, saying that his previous evaluation had been a red herring. In fact, he said, the bunch of files that had been reported as not existing was just the output of `make` being misleading. So he was still curious about why this slowdown was happening, but he no longer felt he understood exactly what the problem was.

The next day Linus replied to himself again, saying, "... and that red herring was what made me think that it always recompiles the 'conf' binary. But no, that's not what is going on. Profiling shows that it does spend a lot of time in the compiler (which was the other thing that made me incorrectly think it was the conf program getting recompiled every time)."

He posted some profile statistics, showing that most of the lost 1.5 seconds was simply testing the various options before compiling. He remarked,



**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**.

“Oh well, I clearly misread the problem. Maybe 1.5s is more reasonable than I really expected it to be.”

But this just didn't sit well with him. That 1.5 seconds gnawed at Linus's brain like a vampiric zombie.

He replied again to himself, pointing out that a third of the profiling stats was spent on a single invocation of the `cc1plus` back end of the GCC compiler, the purpose of which was solely to verify that GCC plugins would work.

Masahiro came back into the discussion at this point. He confirmed what Linus had already said: The `conf` binary was not, in fact, recompiled every time. And he confirmed Linus's profiling statistics for the `cc1plus` invocation. He remarked, “Actually, I did not know this shell script was so expensive to run...” He added, “Even if we are able to manage this script somehow, `Kconfig` invocation still takes more than 1 sec due to the current design.”

But now Linus, with the parasitic vampire zombie still firmly attached to his hindbrain, pointed out:

*“So it turns out that one reason it's so expensive to run is that it does a \*lot\* more than it claims to do.”*

*“It says ‘we need a c++ compiler that supports the designated initializer GNU extension’, but then it actually includes a header file from hell, rather than just test designated initializers.”*

In other words, instead of simply verifying GCC plugin support, `cc1plus` was hauling in a monstrous block of code. And that's what was taking so long.

Linus posted a patch that brought the 1.5 second execution time down by a lot. However, he added, “I'm doubtful we really want gcc plugins at all, considering that the only real users have all apparently migrated to clang builtin functionality instead.”

At this point – no, the story's not quite finished – Kees Cook pointed out that, in fact, the whole test for GCC plugin support was only needed by ancient GCC versions that the Linux kernel no longer supported. And that it would be perfectly acceptable to drop the test entirely, eliminating Linus's 1.5 second slowdown for good.

Furthermore, Kees said, “As for dropping GCC plugins entirely, I'd prefer not – the big hold-out for the very paranoid system builders is the `randstruct`

plugin (though they tend to also use the entropy one too). Clang's version of `randstruct` has not gotten unstuck yet.”

And Linus replied:

“Yes.

*“It sounds like we might be able to delete the build test entirely if we just always expect to have a recent enough gcc.”*

*“Testing the headers for existence would presumably still be needed, just to verify ‘do we have plugin support installed at all’.*

*“But I'm not planning on applying this directly – I find the config overhead to be a bit annoying, but it's not like it is \_objectively\_ really a problem. More of a personal hangup ;)”*

So there you have it. A completely unimportant bug that received a fair number of person-hours of attention – a series of red herrings that ultimately revealed the true problem to be something that actually didn't exist anymore.

Linus, on behalf of us all, I thank you for your obsessive/compulsive hindbrain, and the vampiric zombie monster beasties that feed on it and generate these personal hangups that inspire you to examine every square centimeter of kernel code.

## Saving the World, One Graphics Card at a Time

Dmitry Osipenko was concerned about certain Linux systems “becoming burning hot” even while the system was idling. In particular, he said, some NVIDIA graphics cards were getting just a little too much voltage. Ideally, the Linux power management features would regulate the voltage allocations a bit more conservatively, giving just the minimum required voltage to each different chip, depending on its particular needs and level of activity.

Dmitry posted a bunch of patches to implement voltage scaling for some NVIDIA Tegra chips, which he had tested on a number of running systems. In these tests, he had noticed a five degree cooling on some systems. He suggested that it would be fairly straightforward to extend these voltage scaling features to other chips, as needed.

There were some technical comments. Michal Mirosław noticed some code duplication, and Ulf Hansson also felt that Dmitry hadn't actually located the fix in the best spot in the kernel interface.

Viresh Kumar also spotted a bug in Dmitry's code. But he disagreed with Ulf's criticism of Dmitry's choice of interface. He remarked, “if the hardware (where the voltage is required to be changed) is indeed a regulator and is modeled as one, then what Dmitry has done looks okay. i.e. add a supply in the device's node and microvolt property in the DT entries.”

Ulf backed down, saying, “I guess I haven't paid enough attention how power domain regulators are being described then. I was under the impression that the `CPUfreq` case was a bit specific – and we had legacy bindings to stick with”.

However, Viresh was not certain about his position either, remarking elsewhere, “I am also confused on if it should be a domain or regulator, but that is for Ulf to tell.”

In fact, all of these folks, and others, joined with Dmitry in a general-purpose design/implementation discussion, trying to figure out the best shape and structure of Dmitry's proposed voltage scaling feature. Everyone seemed to agree that this would be good – especially given the improved temperature results – but it was clear that Dmitry's code was a first attempt, intended to generate exactly this sort of discussion.

The discussion itself ended in the middle, with patches flying madly about and more work on the way.

But personally, I love seeing patches like this. Imagine the amount of energy savings they represent to the world, when applied to over a billion running Linux systems. It's an excellent moment in history for this sort of patch.

## How to Train Your Kernel Developer

Julia Lawall reported that the official Linux kernel version 5.10 failed to boot on her organization's Intel Xeon CPU E7-8870 v4 @ 2.10GHz server. She added that Linux v5.9 and earlier worked fine. She posted some backtrace data to help diagnose the problem.

Linus Torvalds's quaint suburban house spun around three times and steam began rising from the rooftop.

Linus asked if the problem started with the `-rc1` version (i.e., the first release candidate after the 5.9 release). And he re-



quested, “Could you try bisecting – even partially? If you do only six bisections, the number of suspect commits drops from 15k to about 230 – which likely pinpoints the suspect area.”

Bisecting is a very cool bug-hunting technique based on a binary search. You’ve got a known-working and a known-broken version, so you test out the version directly in the middle of those two versions. Whatever the result of your test, you’ve just eliminated 50 percent of all the patches that might be the culprit. Then you do the same thing again and keep doing it until you find the patch that actually caused the problem. As Linus said, just a few bisections will rapidly reduce the suspicious area from “everywhere” to “right here.”

Linus also looked at Julia’s backtrace data and said that there seemed to be some breakage in the SCSI device scanning code – though he couldn’t be certain if that breakage was causing Julia’s problem.

Julia confirmed that yes, the -rc1 release also had the problem. Thus eliminating from consideration any patches that were added between -rc1 and the official 5.10 release. She also agreed to do a bit of bisecting to try to further narrow down the bad patch.

Martin K. Petersen said that the only SCSI change at that time had gone into the -rc2 release, not -rc1. This put SCSI out of the running to be behind Julia’s report. But he said he’d dig further and see what he found.

Linus confirmed that he’d also found the -rc2 SCSI patch and “dismissed it for the same reason you did – it wasn’t in rc1. Plus it looked very simple.” He

suggested, “maybe Julia might have misspoken, and rc1 was ok, so I guess it’s possible.”

But Julia replied definitively to Linus, saying, “rc1 was not ok. I just started re-basing and the first step was not ok either.” She also posted the `dmesg` file from a successful boot of Linux 5.9 to aid in the bug hunt.

Linus looked at the `dmesg` file and said:

*“Ok, from a quick look it’s megaraid\_sas.*

*“The only thing I see there is that commit 103fbf8e4020 (‘scsi: megaraid\_sas: Added support for shared host tagset for cpuhotplug’).*

*“Of course, it could be something entirely unrelated that just triggers this, but I don’t think I’ve seen any other reports, so at a first guess I’d blame something specific to that hardware, rather than some generic problem that just happens to hit Julia.”*

Without confirming that `megaraid_sas` was in fact the real problem, John Garry reported that in that same area, “we did have an issue reported here already from Qian about a boot hang.”

John suggested that Julia revert the `megaraid_sas` patch and try a test boot.

And Linus said:

*“Julia – if it’s that thing, then a*

```
git revert 103fbf8e4020
```

*“would be the thing to test.”*

Linus also said that based on this info from John, `megaraid_sas` did seem like a probable culprit.


And Julia replied, “This solves the problem. Starting from 5.10-rc7 and doing this revert, I get a kernel that boots.”

Martin thanked Julia for testing and said he’d revert that change for the next official release.

In fact, the patch would be reverted, but then re-added after some more patches from Ming Lei that were expected to fix the underlying issue. John sent a link to Julia, asking her to test Ming’s patches as they stood, to see if the boot problem recurred. Julia did so and confirmed, “5.10-rc7 plus these three commits boots fine.”

If anyone ever writes a textbook about how to report a kernel bug and make sure the Linux developers look at it, this incident will be the first example given. Julia started off by reporting her hardware, the kernel version in question, and included a pile of relevant output that could be used to trace the problem. It didn’t hurt that she was reporting a bug in an official release, rather than just a release candidate.

Since her system was known to be able to reproduce the problem, she helped out by bisecting kernel versions to help narrow down the search. She reverted patches and tested alternate versions of the kernel, including testing new patches that were slated to go into the next official release. At every step along the way, she posted more and more data from various boot attempts, including those from the broken kernel as well as from known working versions.

The kernel developers, for their part, thanked Julia for each of those steps, and offered easy-to-follow guidance on what she could do to add further information. It’s obvious that they were super happy to have such good bug reportage and wanted to encourage more of the same. 



We compare the Bash, Zsh, and fish shells

# Shell Shopping

Don't let your familiarity with the Bash shell stop you from exploring other options.

We take a look at a pair of alternatives that are easy to install and easy to use: Zsh

and fish. *By Joe Casad, Tim Schürmann, and Harald Zisler*

**W**hen you open a terminal window and start typing a text-based command, you are interacting with a command shell or command line interface (CLI). A shell is a program that acts as an interface to a command processor, the component that carries out commands by interacting with the underlying system.

Some of the command shells around today predate the birth of Linux and were originally used on classic Unix systems (see the box entitled “A Little History”). The most popular shell for Linux systems in the Bourne Again Shell, which is known universally as Bash [1]. Bash is the default for the terminal window on most systems, and Bash syntax is the norm for most popular discussions of Linux, including most of the articles in this magazine. But it isn't like you're stuck with it. Most distros include alternative shells in their package repositories, and it is easy to try out a different shell if you're interested.

“Why switch?” you might be asking. If you're doing fine with Bash, there is no reason to change; however, many Linux users just like to experiment. And the fact is, the whole reason why all the different shells exist is because users saw something missing from the existing options and decided to innovate.

## A Little History

The first Unix shell was the Thompson shell, which was written by Unix creator Ken Thompson and appeared with the first version of Unix in 1971. The Bourne Shell followed eight years later in 1979 and was created by Stephen Bourne, who, like Thompson, worked for Bell Labs. The Bourne shell included several improvements over the Thompson shell and other early Unix shells, including better programming features and a more versatile means for defining the user environment.

At around the same time (late 1970s), an ambitious young Berkeley computer scientist named Bill Joy developed the C shell as part of his work on an early version of the BSD free Unix system. (Joy went on to cofound Sun Microsystems.) The goal of the C shell was to create a shell with a syntax that was more like the C programming language, to make the shell more intuitive for the many programmers who worked with Unix. The C shell was popular for many years, and an improved version of the C shell called tcsh is still a popular option today.

In 1983, Bell Labs rolled out another important shell option when the Korn shell (ksh) was announced at the USENIX technical convention. Korn shell, which was developed by David Korn, is backwards compatible with the Bourne shell but includes many features of the C shell.

The Bourne Again Shell (Bash) was the result of the Free Software Foundation's need to create a shell that was compatible

with Bourne syntax but could be licensed with the GPL. Bash was originally written by Brian Fox. According to Wikipedia “Stallman and the Free Software Foundation (FSF) considered a free shell that could run existing shell scripts so strategic to a completely free system built from BSD and GNU code that this was one of the few projects they funded themselves, with Fox undertaking the work as an employee of FSF.” Over time, Bash has picked up many useful features and has even adopted some features from the Korn shell and the C shell.

The small and fast Almquist shell, by Ken Almquist, also arrived in the late 1980s, as a clone of the System V Bourne shell, and it replaced the Bourne shell in several BSD versions. The Almquist shell is best remembered today as a forerunner to the Debian Almquist shell (Dash), which was developed in 1997 by Herbert Xu and which still exists today as a go-to non-interactive shell in several distros, including Debian and Ubuntu.

The Z shell (Zsh) first appeared in 1990 and was created by Paul Falstaff. Zsh is based on Bourne shell syntax but includes many extensions and improvements. The fish shell, which was initially released in 2005, does not fall neatly into the family tree of any of the previous shell variants. The creators of the fish shell wanted a practical and user-friendly shell option that was designed for the features of more modern systems and wasn't tied to the structures and assumptions of previous shells.





```
sudo apt install zsh
```

The path to the user's login shell is defined in the `/etc/passwd` file. You can change the setting interactively to another shell that has been previously installed on your system using the `chsh` command:

```
chsh -s bin/zsh
```

Alternatively, you can change the login shell using the `usermod` command:

```
usermod --shell /bin/zsh user_name
```

Keep in mind that the files and settings that define your user environment are designed to work with your current login shell. If you change the login shell, be sure you know what you're doing, and be ready to fix something if it breaks.

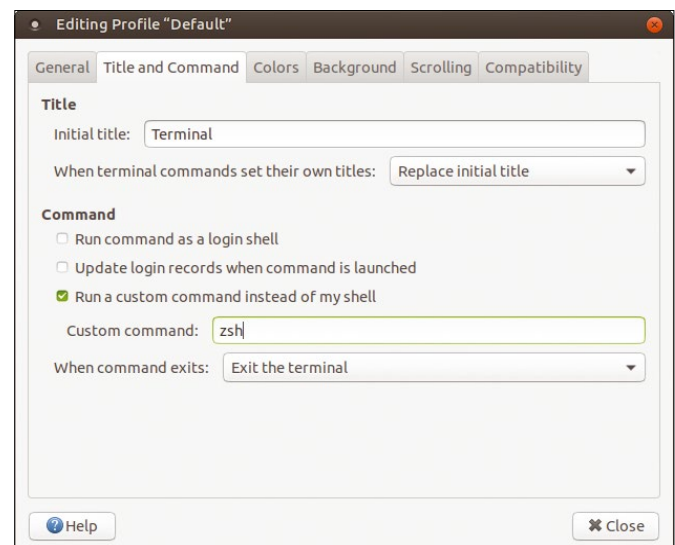
For most modern desktop systems, the default interactive shell is defined in the configuration of the terminal emulator application. The steps for permanently changing the default depend on the terminal app. Several terminal tools let you define a profile, where you can specify a custom command to run when opening the session. Enter the name of the alternative shell in the box provided (Figure 1).

If you just want to experiment with using a different interactive shell from within a terminal session, in most cases, you can just enter the name of the shell at the command prompt:

```
$ zsh
```

The session should switch immediately to the new shell.

The non-interactive shell – the shell that runs a script – is defined in the interpreter directive (the *shebang* line) at the beginning of the script, which specifies the interpreter to use with the script:



**Figure 1:** Configuring a terminal emulator for an alternative shell.

This article compares the Bash shell with a couple leading alternatives:

- Z shell (Zsh) [2] – a Bourne shell derivative that borrows some features from C shell (csh) and the later tcsh.
- Fish Shell [3] – a user-friendly shell that was designed for simplicity and convenience.

We'll end with a closer look at getting started with configuring the Z Shell.

## What Do You Mean "Shell"?

Before diving deeply into comparisons, it is best to stop for a moment and consider that the term *shell* as it is used in this article can refer to any of three different components of the Linux environment:

- Login shell – the shell that runs when you log onto the system. The login shell plays an important role in starting the user environment and defining custom settings
- Interactive shell – the shell that interacts with a user entering commands at the command prompt
- Non-interactive shell – a shell that executes the commands in non-interactive shell scripts

Note that these three shell types refer to a role, not to a specific tool. It is possible for the Bash shell or Zsh to serve in all three of these roles on a single system.

The following Bash command shows the shells that are currently installed on your system:

```
cat /etc/shells
```

If the shell you wish to activate isn't present on your system, install it using your system's package manager. For Ubuntu:



```
#!/bin/sh
```

The `sh` name originally referred to the Bourne shell. Some modern systems use `sh` as a link to another shell. For instance, Ubuntu systems link `sh` to the Dash shell, which serves as the default non-interactive shell.

If you want your script to link to a different shell, you can just change the shell in the shebang line:

```
#!/bin/bash
```

But be aware that the script was probably written for a specific shell, and if you switch to a different shell, you'll probably need to adapt the script. The major shells have varying degrees of compatibility. A simple script written for `bash` might work in another Bourne-based shell, but there is typically no compelling reason to change. A more likely scenario would be that you would write a *new* script for an alternative shell and then use the interpreter directive to specify a shell that is not the default:

```
#!/bin/zsh
```

The interactive shell in your terminal emulator is the easiest way to experiment with different command shells. But remember, you can only activate a shell that has been previously installed on your system, so install the alternative shell first with your package manager.

### Bash vs. Zsh

Most users who migrate from Bash to Zsh for interactive work report a smooth transition. Zsh, like Bash, is based on the Bourne shell, and many of the same commands work in both shells. A Linux user can sometimes sit down with a terminal app configured for Zsh and not even notice the difference.

The situation with scripts is a bit more complicated. Read on for a deeper dive into some of the details later in this article, but for now, it is interesting to note that Zsh comes with the ability to emulate other shells. Use the `emulate` command to specify which shell you would like Zsh to emulate.

```
emulate ksh
```

With no arguments, the `emulate` command shows the current emulation mode:

```
emulate
zsh
```

You can use emulation modes to get Zsh to behave like the Bourne, Korn, or C shell, although full compatibility is not guaranteed.

Zsh users believe Zsh has several advantages over Bash, including:

- Better customization features
- More advanced autocompletion and spell checking
- A better collection of extensions and plugins

The popular and well respected Oh My Zsh project [4] provides a framework for extending Zsh, with a large collection of plugins,

themes, and other helpers for bringing new features to Zsh. As you'll learn later in this article, Zsh also comes with a handy configuration menu that starts the first time you use the shell and helps you configure some important preferences related to auto-completion, command history, and more.

Zsh enthusiasts argue that Zsh does everything Bash does plus more, so why not just use Zsh? Bash holdouts maintain that the two shells do have some differences, especially for scripting, and with so many million scripts out in the world already written for Bash, why change? Many of the users who are accustomed to Bash in scripts also prefer to keep using it interactively in an effort to keep things simple and not have to contend with the details of another system.

It is worth mentioning, also, that, whereas Bash is licensed under the GPLv3 license, Zsh has an MIT-style license, a free but permissive, non-copyleft license similar to the BSD license. So if you prefer to use tools with copyleft protection, you're better off with Bash.

### Bash vs. fish

*Fish* stands for Friendly Interactive Shell, and the emphasis is on "interactive." Even the fish developers recommend continuing to use Bash for as a non-interactive shell. For that matter, they point out that you could also write your scripts in any other languages you can reference from the shebang line, such as Python or Perl. Fish does, however, include some scripting features, allowing users to create their own functions in fish syntax to simplify tasks and add custom capabilities.

Fish was designed to address some of the weaknesses of the other shells, so it provides some benefits that aren't present in the other shells; however, this departure from tradition means that it is also the least compatible. The fish shell was envisioned as an interactive tool, and most of its benefits are at the command line. Fish provides sophisticated autocompletion that even accounts for command history in making suggestions. A useful tab completion feature lets you enter the first few letters of the command and press the tab key to see the possible options. Other shells offer tab completion, but the fish version uses actual information from the man pages for a more complete range of alternatives. Fish fans also like the automatic syntax highlighting. Incorrect parts of a command show up in red, allowing you to quickly see what needs correcting.

Enter the `fish_config` command at the fish command prompt to launch an extensive GUI interface that you can use for configuring and customizing fish settings (Figure 2).

But the best way to experience the fish shell is to try it for yourself. The fish project website includes a link to a web page where you can try the fish shell in a browser window (Figure 3).

Like Bash, fish comes with a copyleft GPL license.

### Bash, fish, and Zsh

Although all three shells follow the Posix standard, only Bash officially follows it. Bash even comes with a corresponding mode that adheres even more strictly to the Posix specification and, for example, ignores Bash's own configuration files [5].





Command input in all three shells leans either on Emacs or vi. You will also find that they all support a multiline mode that lets you input commands that span several lines.

Bash, fish, and Zsh will colorize the individual components of a command, if so desired, and the color scheme can be customized. Fish even supports the use of up to 16.8 million colors. You can redesign the prompt structure in all three shells.

In Zsh, every action at the prompt executes a widget. For example, Esc + B calls a widget that lets you move the cursor to the beginning of the word. You can create your own widgets.

All three shells remember every command you input. You can use a keyboard shortcut to call up older commands from this history. By default, Zsh remembers only the last 30 lines and also discards the list on exiting. If

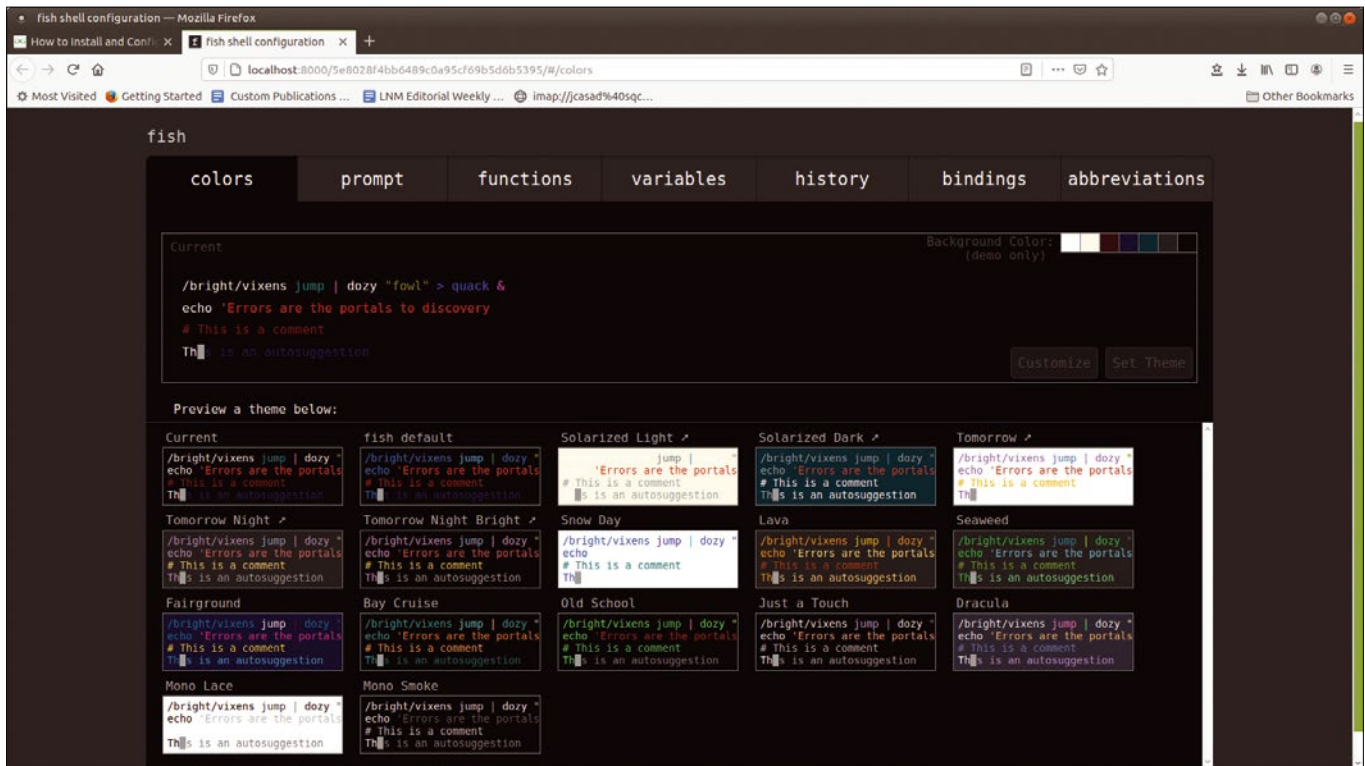


Figure 2: Fish offers an extensive GUI interface for managing configuration settings.

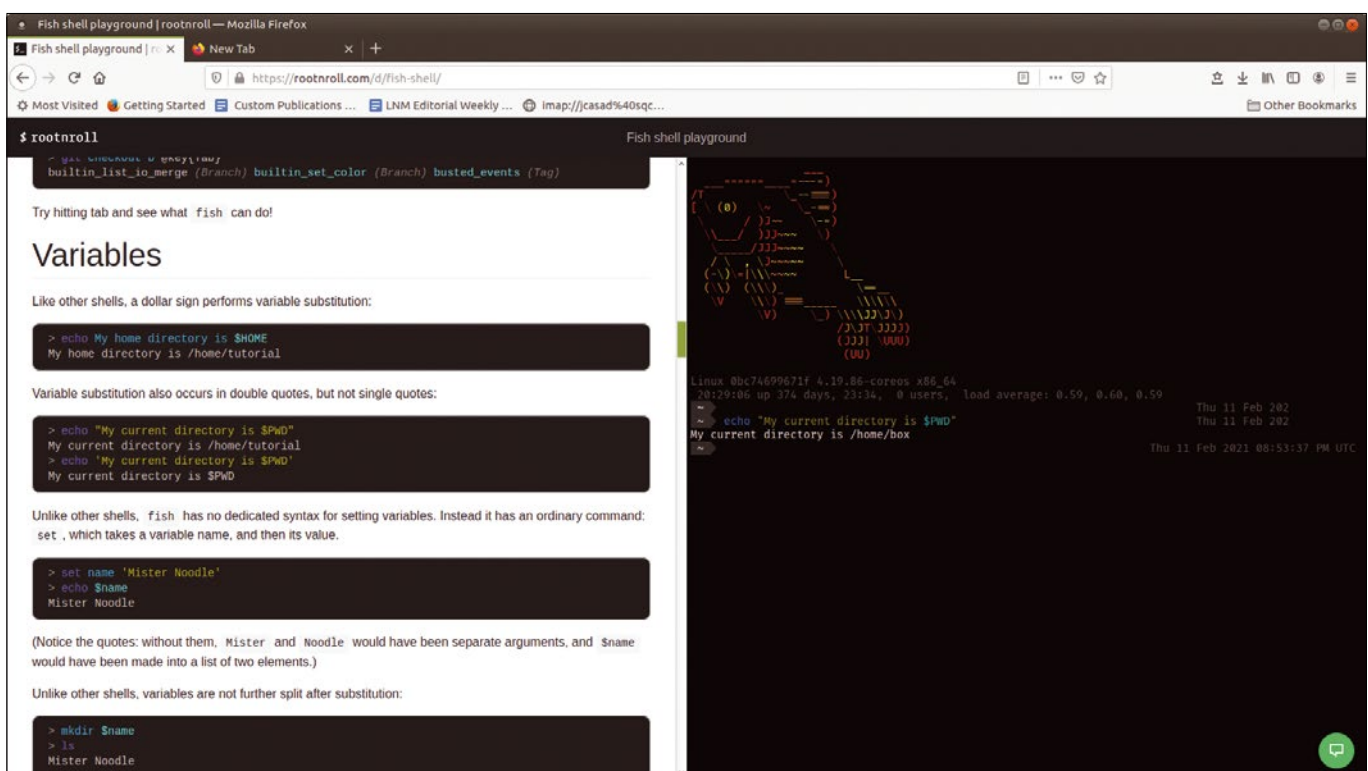


Figure 3: Fish in a browser: Enter commands in the right pane while you work through the tutorial on the left.



necessary, you can change this setting so that all open shells share a history (see the section on configuring Zsh later in this article).

On request, Bash, fish, and Zsh will complete a partially typed command. Autocompletion is designed to complete the names of variables, users, hosts, and files. In all three shells, it is possible to add your own rules to this function.

Bash and Zsh also offer limited spellchecking, with the Zsh acting in a far more intelligent way. Bash, for example, only corrects file names after `cd`, but Zsh also corrects misspelled program names and other misspellings. Fish, on the other hand, suggests a possible command from the history while typing and highlights (typing) errors in red.

## String of Pearls

Variables are used to record data. In Bash or Zsh, you do this by defining, for example, `color=red`, and in fish, you need the `set` keyword for this purpose:

```
set color red
```

The equals sign between the name and the value is also missing in fish, which makes the two methods incompatible.

Bash, fish, and Zsh can all store multiple values in arrays. Depending on the variant, these arrays are internally called lists, fields, or vectors. An element in an array is accessed by specifying its position:

```
colors[1]=red
```

In both Bash and Zsh, the count starts at zero. In the example, the word `red` would therefore be stored in position two in the `colors` array. In contrast, fish starts at one, which would mean that `red` is the first element in the array in the preceding example.

In addition, fish supports the `..` operator, which you can use to access multiple elements at once:

```
echo ${colors[2..4]}
```

In this example the colors at positions 2, 3, and 4 of the array would be output.

## Substitution

All shells are capable of command substitution, which lets you substitute one command for another. In the following example, Bash and Zsh would first execute the `date` command and then pass the returned value as a parameter to the `mkdir` command:

```
mkdir $(date +%Y-%m-%d)
```

In older code, you can often find the now obsolete notation with backticks ``date +%Y-%m-%d``. fish makes the code more readable. It does not support either variant but requires simple brackets:

### Listing 1: if Queries

```
# Bash and Zsh
if [[ $color == "red" ]]; then
    echo $color
fi

# Zsh
if [[ $color == "red" ]] {
    echo $color
}

# Fish
if test $color = red
    echo $color
end
```

```
mkdir $(date +%Y-%m-%d)
```

Using the same principle, shells integrate the contents of variables into texts. In Bash and Zsh, you put the variable in curly braces as follows:

```
ls letter${date}.txt
```

However, fish again does its own thing with slightly different syntax:

```
ls letter${date}.txt
```

Bash, fish, and Zsh provide some predefined and special variables, but they differ from shell to shell. For example, Bash and Zsh users will find the first parameter passed to the script on the command line in the variable `$1`. Fish, on the other hand, does not support a `$1` variable, so you will need to extract all the transferred parameters from the `$argv` array if needed.

## Globbering

All three shells support the use of various placeholders (this is known as globbing or file name expansion) in file names. In all three shells, the question mark `?` stands for any letter. Bash also supports the following notation:

```
ls @(letter|readme).txt
```

This command only lists the `letter.txt` and `readme.txt` files. Zsh can do the same, but by default, it only requires the brackets to be specified:

```
ls {letter|readme}.txt
```

Both shells also allow for more complex conditions inside the brackets.

Zsh also offers what are known as glob qualifiers, with which you can track down very specific file types. For example, `ls *(-@)` lists all symbolic links that are currently orphaned.

Bash and Zsh can also manipulate strings. For example, `${var:4:2}` returns the fifth and sixth characters from the

### Listing 2: Functions

```
# Bash and Zsh
saywhat() {
    echo "Hello!"
}

# Bash and Zsh (alternatively)
function saywhat {
    echo "Hello!"
}

# Fish
function saywhat
    echo "Hello!"
end
```





text in the variable `var`. You can also use both to solve simple arithmetic problems. To add, say, 1 and 2, use the following construct:

```
sum=$((1+2))
```

However, the functions for calculating and editing texts by no means offer the capabilities of specialist programs such as `bc`, `sed`, and `awk`.

## File Streams

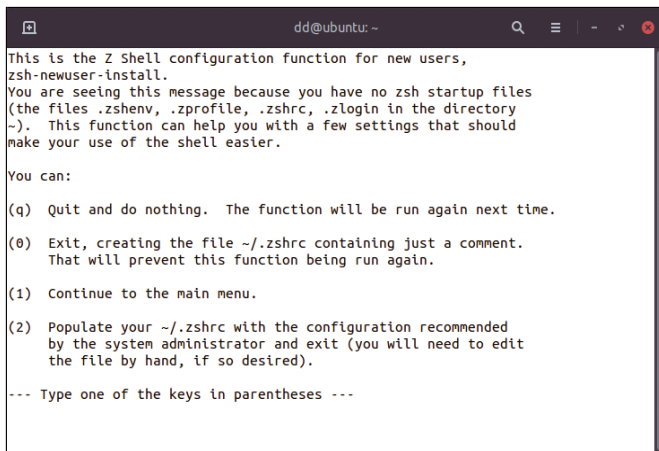
All shells provide the ability to redirect output from and input to program(s) and link them using pipes:

```
ls -la | grep "readme" | more
```

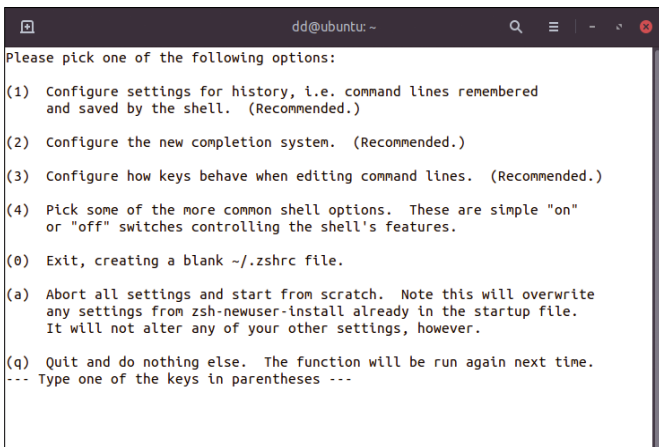
As an alternative to the `echo` command, all shells still offer the built-in `printf` command, which writes variable contents to a text:

```
printf "Name: %s Age: %d" $name $age
```

As in the function of the same name from the C programming language, you put placeholders in at the right places, and the shell replaces them with the values from the variables.



**Figure 4:** The start menu invites you to configure Zsh settings.



**Figure 5:** The Zsh configuration main menu.

## Control Structures

Developers control the flow within a script using appropriate control structures and loops. However, each shell uses a slightly different syntax. Listing 1 shows an example of an `if` query in Bash, Zsh, and fish.

In addition to the keyword `if`, all shells also provide a `while` loop and a `for` loop. Bash and Zsh even support two different notations of the `for` loop. However, the control structures and loops that are offered also differ between the shells. For example, Bash and Zsh still offer an `until` loop, which is missing in fish.

Zsh is the only shell that offers a repeat loop, which repeats a given number of commands. Bash and Zsh also have a `select` loop that creates a simply designed menu for a selection.

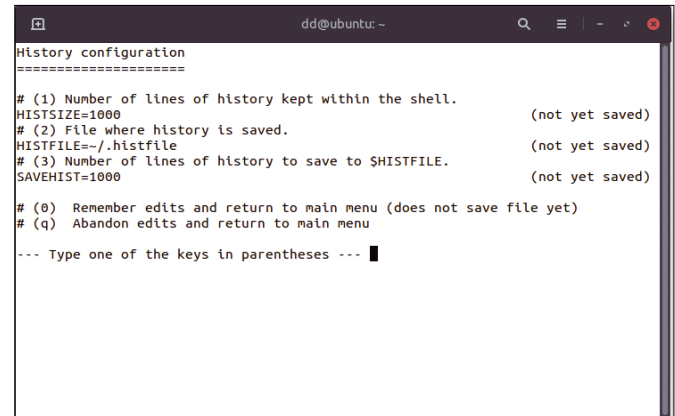
## Functions

All shells let you combine several commands into one function. As Listing 2 shows, Bash and Zsh use an identical syntax, whereas fish again goes its own way.

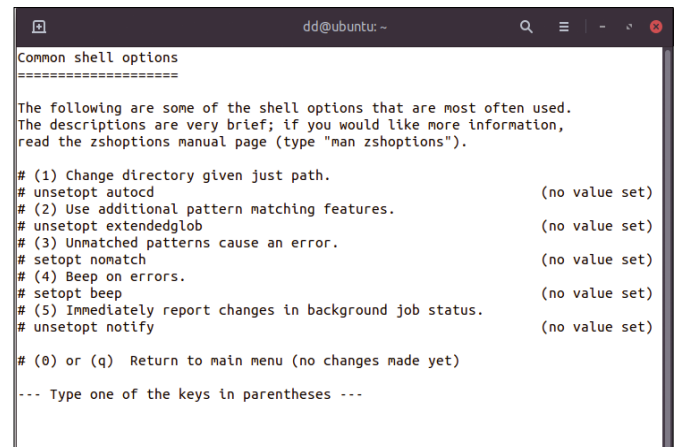
Access to a variable only works in the context in which you defined it. If you use Bash or Zsh, only those variables that you explicitly mark as `local` are local:

```
local name = "Peter"
```

Zsh also supports anonymous functions that have no name. Such a function is immediately executed by the shell as soon



**Figure 6:** Command history settings.



**Figure 7:** Configuring additional Zsh settings.

**Figure 8:** Displaying prompt patterns.**Listing 3:** Changing the Prompt

```
autoload -Uz promptinit
promptinit
prompt pattern_name
```

as it stumbles upon its definition in the code. This can be used, for example, to restrict the visibility of variables in start scripts to the body of the function.

## Outsourced

Zsh and fish can automatically reload functions used by a script from separate files. However, this autoload mechanism differs slightly in the two shells.

For example, in Zsh you first need to explicitly register the corresponding functions with the `autoload` keyword. This shell also lets you extend the range of functions using modules, which you can even load at runtime if required. Some of these are already provided by the shell: For example, the `zsh/zftp` module adds a full FTP client.

Using the built-in `compile` command, Zsh can compile functions and scripts and store them in a binary file. The results are loaded faster then, but they are not available in other shells.

## Configuring Zsh

The best way to try out fish is through the online browser interface [6] or by running the `fish_config` utility and exploring the graphic configuration utility. With Zsh, you'll also want to install it and experiment before deciding to migrate.

Zsh has one thing Bash doesn't have: a handy configuration menu that appears when you start the shell for the first time. Zsh looks for the configuration files, and if it can't find them (in other words, if you haven't configured the shell yet), the config menu launches (Figure 4). Press 1 to enter the main menu (Figure 5).

When you get to the main menu, press 1 to set the scope of the history. You can

enter the number of lines of history you would like to retain within the shell, select a file for saving the history, and choose how many lines you would like to keep within the file. In the example shown in Figure 6, the history has been extended to 10,000 lines.

Enter 0 to save and return to the main window, or enter q to return without saving.

Back in the main window, press 2 to go to the autocomplete settings. You can choose between the default autocomplete options (1) or run a configuration tool that will let you customize the autocomplete settings (2). Choose 3 if you don't want to use autocomplete.

Item 3 in the main menu lets you set the keyboard behavior in the shell's line editor. You can get the editor to behave like either of the classic Unix editors Emacs or vi. It is best to leave the suggested value.

Item 4 in the main menu lets you configure some other common shell settings (Figure 7), including settings related to pattern matching, command history, and whether to beep on errors.

When you're finished with the configuration menu, you can save the settings by pressing 0 in the main window. The new settings will take effect immediately.

An example of a Zsh feature with lots of room to customize is the command prompt. The developers have invested considerable energy into command prompt options. Customizing the string of text that appears before you enter a command might seem like a random personal choice, but many admins use the prompt for orientation and documentation purposes. For instance, including the date and time on a prompt means that a screenshot will document exactly when the command was run.

Use the following commands in Zsh to load the preconfigured prompt options:

```
autoload -U promptinit
promptinit
```

**Table 1:** Zsh Configuration Files

File Name	Description
.zprofile	Login settings and commands
.zshenv	Settings that apply regardless of how the shell is called
.zshrc	Shell startup settings and commands
.zlogin	Commands for logging in
.zlogout	Commands for logging out of a session

**Table 2:** Zsh Prompt

Meaning	Statement
Date	%D
Time	%T
Host name without domain	%m
Full computer name	%M
User ID	%n
Exit code of last command	%%?
Current directory	%/
Current directory	%~ (truncates the home directory specification)
Sequence number in history	%!





After you enter these commands, you can list the available prompt themes with `prompt -l`. To see what they all look like, enter `prompt -p` (Figure 8).

Note that the two-line prompts in Figure 8, which show the prompt above the actual command line, add clarity for situations where the prompt is providing essential information for the user. After you set the prompt interactively with:

```
prompt -s pattern_name
```

a hint appears telling you that, for a permanent change of the prompt, you have to enter the pattern in `~/.zshrc`. The entry in `.zshrc` is similar to what you just did at the command line (Listing 3).

The Zsh configuration files are shown in Table 1. The names in Table 1 refer to the files in the user's home directory. Global files with the same names (but without the leading dot) are found in `/etc`.

Zsh processes the configuration files in the order `.zprofile`, `.zshrc`, and `.zlogin` when you log in. Use the `.zshenv` file to configure settings that apply regardless of how the shell was started. When called manually, Zsh only evaluates `.zshrc` and `.zshenv`. When logging out of the session, the commands from `.zlogout` are executed.

Note that the `.zlogin` and `.zlogout` config files make it possible to facilitate tasks by creating specially tailored user accounts, for example, for data backup, batch jobs in remote data transmission, or system maintenance. In this way, you can even delegate tasks to nonexperts, because all they have to do is log on, and the task executes automatically. Placing `exit` in the startup file automatically logs the user off after the task is completed.

If you don't like the predefined prompt themes provided with Zsh, you can design your own prompt. Like other personal settings for interactive sessions, prompt design details go in `.zshrc`. Setting up your own prompt basically works like it does in Bash, and even the escape sequences for the colors are the same [4], but you need to initiate them in Zsh using `\e`. The placeholders you use to build the prompt content, on the other hand, are completely different from those for Bash (see Table 2). For more information, refer to the Zsh manual [2].

Variables and aliases are entered in the `.zshenv` file. In our test, the usual colors for directories, executables, and so on were not output by the `ls` command; I corrected this with the following line:

```
alias ls="ls --color"
```

Any additional variables and `$PATH` extensions work like in Bash and need to be added to `.zshenv`.

## Knowledge Warehouse

Bash provide developers a comprehensive reference on their website, but it is quite technical and dry [7]. A better introduction is provided by tutorials like the still-valid Advanced Bash Scripting Guide [8].

For Zsh users, the official documentation is the first place to go [9]. The fish developers offer a detailed tutorial, as well as a reference on their website [10].

## Conclusions

Bash, fish, and Zsh offer a similar range of functions, but they differ in many details. These dissimilarities are especially present in the syntax of the control structures, which means that only extremely simple scripts run without changes in a different shell.

If you want to use a script on as many systems as possible, there is no alternative to Bash. It is normally preinstalled and has the biggest user base. If you have problems or questions, you can usually find help quickly on the Internet.

As for interactive command-line operations, many users express a preference for Zsh or fish because of their advanced autocompletion, enhanced configuration options, and other user-friendly features.

If portability does not play a role or is just a minor concern, the best approach is to try all three shells and use the one that works best for you. For example, if you only launch programs from the command line, fish's automatic suggestions could save you quite a bit of typing. ■■■

## Info

[1] Bash: <http://www.gnu.org/software/bash/>

[2] Zsh: <http://www.zsh.org>

[3] Fish: <https://fishshell.com>

[4] Oh My Zsh: <https://ohmyz.sh/>

[5] Bash POSIX mode: <http://www.gnu.org/software/bash/manual/bash.html#Bash-POSIX-Mode>

[6] Fish in a Browser: <https://rootnroll.com/d/fish-shell/>

[7] Bash manual: <http://www.gnu.org/software/bash/manual/>

[8] Advanced Bash Scripting Guide: <http://tldp.org/LDP/abs/html/>

[9] Zsh documentation: <http://zsh.sourceforge.net/Doc/>

[10] Fish documentation: <https://fishshell.com/docs/current/index.html>



Manjaro Linux

# The Ubuntu of Arch Distros

Standing on the shoulders of Arch Linux, Manjaro offers simplicity and stability. *By Bruce Byfield*

Occasionally, a derivative distribution comes to rival the original. These days, for example, Ubuntu and Linux Mint are as popular as Debian. Likewise, for the last six years, Manjaro [1] has consistently eclipsed the popularity of Arch Linux upon which it is based. In fact, by combining the simplicity and rolling release structure of Arch Linux with the organization of Debian, Manjaro has become one of the leading Linux distributions of any origin. Recently, Lisa Singer answered my request

for more details on behalf of the Manjaro corporate management team.

Manjaro began as a passion project of three friends from Munich: Roland Singer, Guillaume Benoit, and Philip Müller. First announced on the Arch Linux Forums, primarily as an installer in 2011, it immediately went dark as the three friends created the distro's base tools. The first general release came in September 2015, with official versions running Xfce, KDE, and Gnome as desktops. Over the years, unofficial commu-

nity editions running Awesome, bspwm, Budgie, Cinnamon, Deepin, i3, LXDE, LXQt, MATE, and Openbox have also been released.

While going in its own direction, Manjaro maintains close relationships with Arch and Arch-based distributions. Most of its packages originate in Arch. In addition, most other

Arch-based distributions use the Calamares installer, "which is mostly based on our original graphical installer," according to Singer. "But it got rewritten as an install framework. Before that, we used Cnichi from Antergos and kept in touch with its developers." Today, Manjaro's developers continue to work with their counterparts at ArcoLinux, EndeavourOS, and KaOs to develop the Calamares installer (Figure 1).

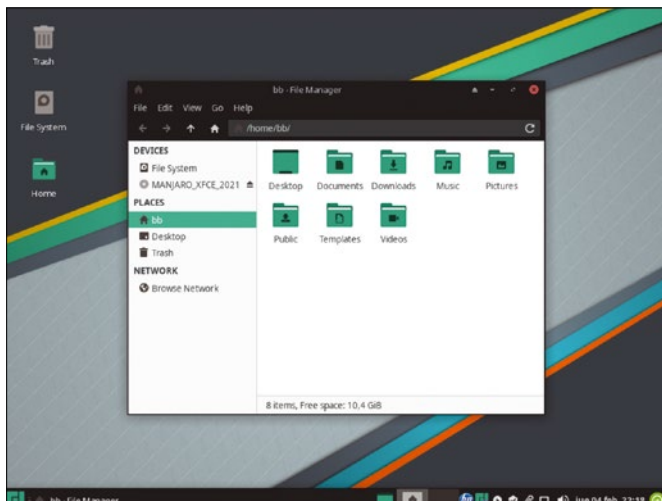
Today, Manjaro has 18 core team members, as well as additional contributors. Last year, its main releases had 4,265,621 downloads, 38 percent for the Xfce edition (Figure 2), 36 percent for KDE, and 18 percent for Gnome. Its ARM-based editions had 211,699 downloads from OSDN (57 percent for the Raspberry Pi 4, 15 percent for Pinebook Pro, and 12 percent for the PinePhone). The community editions have had 489,543 downloads (26 percent for i3 and 22 percent for Cinnamon).

Manjaro's growth has been so rapid that a couple of years ago, Müller began to consider "ways to secure the project in its current form and how to allow for activities which can't be undertaken as a 'hobby project'" [2]. In September 2019, the result was the creation of Manjaro GmbH. Acting under the advice of Blue Systems, a German free software company, Manjaro GmbH controls the distribution's trademarks, protects its brands, and acts as a legal organization when



**Figure 1:** The Calamares installer is used and developed by Manjaro and several other Arch-based distributions.

Lead Image © maxkrasnov, 123RF.com



**Figure 2:** The Xfce edition is the flagship version of Manjaro, although many others are also supported.

one is needed. In addition, Singer explains, “we employ and pay some of our developers, either part or full time, and extend our team as needed with freelancers. Team members are mostly recruited from the Manjaro community, and tools get open sourced.”

The community itself draws on its own fiscal hosts to collect and distribute donations for its own purposes, working closely with the company, but not being guided by it as closely as Ubuntu is guided by Canonical. For example, officially the Xfce edition is Manjaro’s flagship version, with KDE and Gnome editions also officially supported by the company. However, “If people share an interest in a desktop environment, a new community edition gets born,” Singer says. “Currently, some community developers are working on the new Manjaro sway edition utilizing Wayland and other new technology. All our existing editions are more or less our blueprints – you can either use 1:1 or modify easily with our development tools. That makes it easy to create a modified version or even a spinoff.” Some sense of the interaction between company and community can be seen in the fact that the company’s team page [3] lists almost entirely developers – the sole exception being a designer and illustrator.

## Manjaro’s Features

Manjaro prides itself on combining simplicity and ease of installation. Part of that simplicity is inherited from Arch – the simplicity of making few assumptions about what users need instead of offering

a curated collection of applications. Manjaro does offer more preinstalled apps than Arch, but often it offers a choice. For example, the installer offers a choice between FreeOffice and LibreOffice.

When a desktop environment offers an app, Manjaro is more apt to use that rather than its own tools. Still, like many distributions, Manjaro does contain its own configuration and package tools. However, Manjaro’s design philosophy is best seen in its package and release structure. Inspired by Debian, Manjaro has three repositories: Unstable, Testing, and Stable. The Unstable repository is a snapshot of Arch Linux Stable and is a few days behind it. Then, Singer explains, “Packages maintained by developers of Manjaro get built against that snapshot and internally tested. Then those packages get all moved to the Testing branch, which the community reviews. After positive feedback, a stable snapshot gets created and properly announced. This way, we don’t have to play the cat-and-mouse game with Arch, chasing after them when a library of [an] API gets changed and packages need to be rebuilt. This way, Manjaro has full control and decides on its own when a package will finally land in our stable branch.”

Singer continues, “You can choose the speed in which you have your updates” – and, sometimes, your stability. “Unstable might break your system when ABI or PI changes are made by Arch, and our team isn’t so fast to catch up with those changes. ... However, we recommend the Stable branch on productive systems. You should always keep an eye on our update announcements. Additional info can also be found on our wiki.”

Together, Manjaro’s package repositories offer rolling releases – a system in which packages are updated when ready, rather than holding them back for a general release – although official

releases are announced as milestones in development. “We like to move forward. Software never stops evolving. Maintaining a static distro might seem easier to maintain, but it isn’t. [Maintaining a static distro requires] security patches and fixes from upstream, which you mostly find in newer software, which needs to be backported – and then might not be as good as the original release,” adds Singer.

By contrast, Singer continues, “Normally you install Manjaro once and keep updating. In rare cases you may reinstall. But you get security fixes faster. While our release cycle for the stable branch might take longer, we will backport security features. However, since we push so many updates to your system, you may need a good Internet connection, as updates might be rather larger than other common distributions like Ubuntu LTS might have. With Manjaro, you get a relatively stable Linux distribution with the latest software.”

## Future Directions

Manjaro’s news page [4] seems to be rolling almost as much as its repositories. Unlike most distributions, Manjaro releases news almost like a commercial corporation, often issuing several news releases each month. As I write, Manjaro is focusing on extending its ARM and Plasma support and preparing for the upcoming Gnome 4. Already preinstalled on the Pinebook Pro and PinePhone, the Manjaro company is also looking for new business and community partnerships. Paraphrasing the distribution’s slogan “Enjoy the Simplicity,” the Manjaro management team through Singer says, “Our goal will be to go beyond our enjoyment of simplicity.” Just as Ubuntu stands on the shoulders of Debian, so Manjaro stands on the shoulders of Arch. In both cases, users benefit. ■■■

## Info

[1] Manjaro: <https://www.manjaro.org>

[2] Formation of Manjaro GmbH: <https://www.forbes.com/sites/jasonevangelho/2019/09/08/manjaro-linux-just-made-a-massive-announcement-about-its-future/?sh=3e90e80036f9>

[3] Manjaro Team: <https://www.manjaro.org/team/>

[4] Manjaro News: <https://manjaro.org/news/>



# ADMIN

Network & Security

## COMPLETE ARCHIVE DVD

ISSUES 00-49

ISSUE 50/2019

### SOLD OUT!

OVER  
**3,700**  
PAGES OF UP-CLOSE  
SYSTEM  
ADMINISTRATION

Smart and Practical Information  
for Network Professionals

THE COMPLETE

## raspberry pi GEEK ARCHIVE

ISSUE 233 APR 2020

### LINUX MAGAZINE

Although this Linux Magazine does not have better content  
in the best of our knowledge free of malicious software and  
defects, Linux Magazine cannot be held responsible for any  
and is not liable for any data loss, or damage to data and  
computer systems related to the use of this disc.

DVD  
ROM

More than  
**2,000 PAGES**

of information,  
projects, and fun!  
plus 2 bonus Special Editions!

THE COMPLETE

## UBUNTU user ARCHIVE

DVD

### ALMOST SOLD OUT!

- Searchable DVD
- Content Available  
in Both HTML and  
PDF Formats

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



# Complete Your Open Source Library with Archive DVDs!

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

Save hundreds off the single-copy rate  
with a convenient archive DVD!



**ORDER YOUR DVDS NOW!**  
<https://bit.ly/Archive-DVD>

## A programmable scratchpad

# Notepad Plus

Boop, a digital scratchpad, lets you convert data from your desktop instead of exposing confidential data online. You can even extend Boop with your own functions using JavaScript.

By Christoph Langner

If you need to count words or lines of text, calculate checksums and hashes, optimize program code, or perform any other conversion function, you can easily find an online source to do just that. However, you may not want to copy and paste confidential data to a potentially dubious website.

Instead, Boop [1], a digital notepad that can be scripted, performs this task locally. The open source program originally developed for macOS supports typing or pasting text via clipboard. At the touch of a button, a variety of predefined scripts or even your own small programs can then be applied to the entered text. This means that all the data stays on your computer, and you can save yourself the trouble of searching for the right utility online.

With the release of Boop-GTK [2], a port to the GTK toolkit, Linux and Windows users can now use Boop. The still very young application is not currently listed in the package sources of the major Linux distributions, with the exception of a *boop-gtk* entry in the Arch User Repository (AUR) for Arch Linux.

Users of other distributions need to build the program from the source code or use the Snap [3] or Flatpak packages [4] provided by the project.

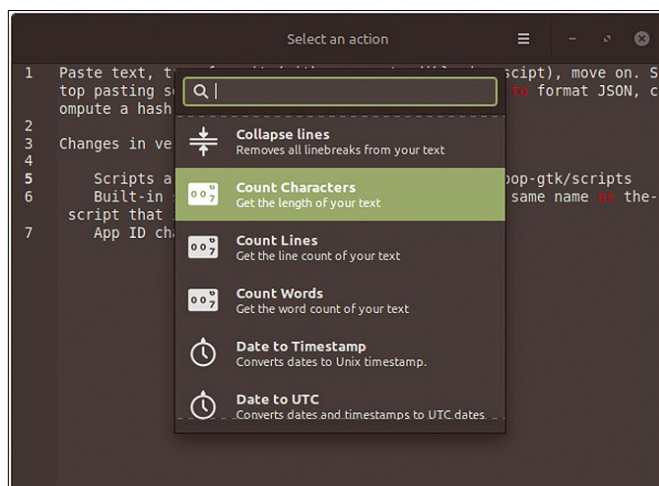
## Scriptable

Fundamentally, Boop works the same way regardless of the task: Copy text or code into the application window or type directly in the editor area, and then activate the script dialog via Ctrl + Shift + P. In the subsequent selection dialog, you can choose between (as of the editorial deadline) 59 different scripts (Figure 1).

The scope of these scripts ranges from *Add Slashes* (which prepends a backslash to certain special characters like ', ", \, or NUL) to a feature that

makes converting localization strings between iOS and Android formats easier. The search field in the dialog header lets you easily filter out a specific function.

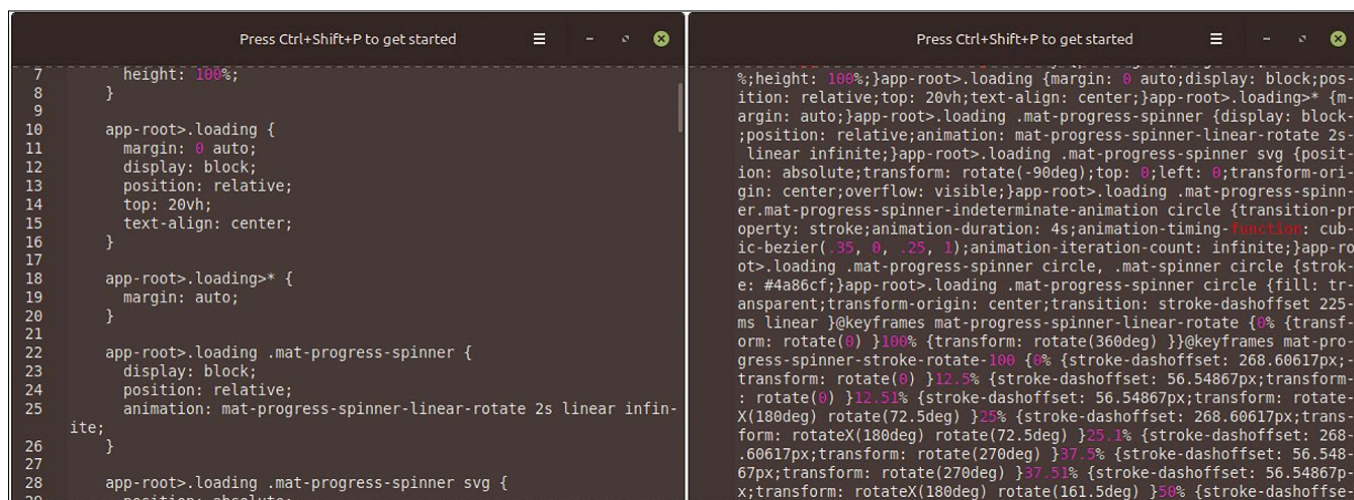
For example, web developers can clean up CSS files with *Format CSS*, prepare text for websites using *HTML Decode* and *HTML Encode*, or compress CSS markup via *Minify CSS*. You can then use *Format CSS* again to transform



**Figure 1:** If you frequently work with text, you can use Boop to count the number of characters, words, or lines.

Lead Image © iKO, Fotolia.com





**Figure 2:** Boop optimizes CSS code or converts the compressed code back into readable text.

the optimized CSS code back into a readable format (Figure 2).

Developers who specialize in programming applications can use *Eval Javascript* to execute JavaScript directly in the context of Boop and view the output (Figure 3). Similar to CSS code, Boop can also minimize JSON and XML and unpack them accord-

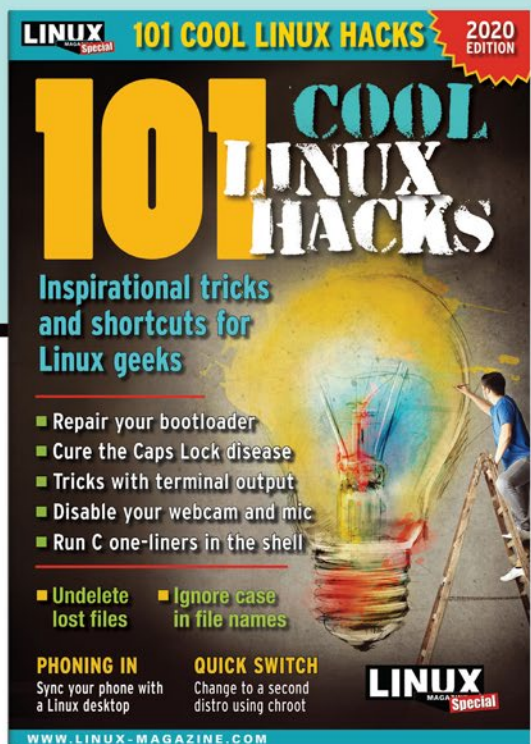
ingly, as well as transform JSON into various other formats.

Various text functions like *Camel Case*, *Kebab Case* (all words connected by minus signs), *Snake Case* (connected with an underscore), *Sponge Case* (random uppercase/lowercase), *Markdown Quote*, *Reverse Lines*, and *Reverse String* complete the repertoire for editors. If re-

quired, MD5 checksums or hash values (SHA-1, SHA-256, and SHA-512) can also be calculated for the entered data.

### Add Your Own

If you are missing a certain function, it can be integrated – with the appropriate know-how – directly into Boop. For this purpose, the *Open Config Directory*



SHOP THE SHOP  
shop.linuxnewmedia.com

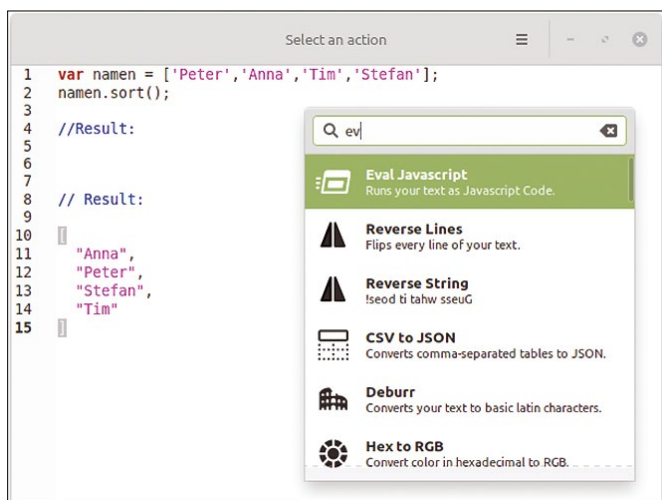
GET PRODUCTIVE WITH  
**101 LINUX HACKS**

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)



**Figure 3:** Use *Eval Javascript* to execute JavaScript code in the application window.

option in the hamburger menu in the upper right corner opens a file manager with the `~/config/boop-gtk` folder. Boop automatically loads JavaScript files stored in the `scripts/` subfolder into the application at program launch time.

For suggestions and details on how the scripts work, use the *Get More Scripts* menu item. It loads the GitHub repository maintained by the original Boop developer in your browser. You will find a number of scripts that have not yet made it directly into the program [5]. You can use these examples as a template to write your own script.

The Boop project also shows an example script on its homepage: The Poopificator randomly replaces words with a poop emoji [6] – if that’s a function you need. To do this, transfer the

contents from Listing 1 to a new file named `poop.js`, and then move it to `~/config/boop-gtk/scripts/`. After restarting Boop, you will find a new entry named *Poopificator* under the action drop-down menu.

The *Poopificator* action now replaces random words with the HTML code `&#128169;`. You then use a second action, *HTML Decode*, to turn the HTML snippets into the desired emoji (Figure 4). The script in Listing 1 requires the second action, because *Linux Magazine* uses typesetting software that does not allow this type of special character. If you transfer the script di-

**Listing 1:** `poop.js`

```
01 /**
02  {
03    "api":1,
04    "name":"Poopificator",
05    "description":"Swaps random words with poop emojis.",
06    "author":"Ivan",
07    "icon":"html",
08    "tags":["poop,emoji,hashtag"]
09  }
10 */
11
12 const poopFactor = 0.2
13
14 function main(state) {
15   var all = state.fullText.split(" ")
16   state.fullText = all.map(x => r() ? x : "&#128169;").
17     join(" ")
18 }
19
20 function r() {
21   return Math.random() >= poopFactor
22 }
```

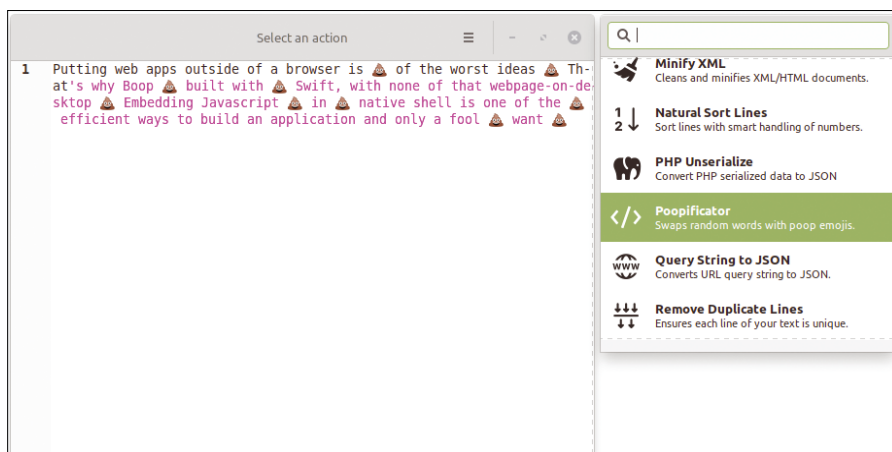
rectly from Boop’s project page, the second action is unnecessary.

## Conclusions

If you rely on web services for one or more functions discussed here, Boop offers a useful local solution. The program works without Internet access, requires fewer resources than a web browser, and all data remains on your computer. In addition, the Boop community continually adds new functions. And if you can’t find the function you need, you can add your own scripts with some basic programming knowledge. ■■■

## Info

- [1] Boop: <https://boop.okat.best/>
- [2] Boop-GTK: <https://github.com/mrbenshef/Boop-GTK>
- [3] Boop on Snapcraft: <https://snapcraft.io/boop-gtk>
- [4] Boop on Flathub: <https://flathub.org/apps/details/fyi.zoey.Boop-GTK>
- [5] More Boop scripts: <https://github.com/IvanMathy/Boop/tree/main/Scripts>
- [6] Pile of Poo: <https://emojipedia.org/pile-of-poo>



**Figure 4:** Using the *Poopificator* sample script to extend Boop’s feature set.

## The sys admin's daily grind: gping and Nextinspace

# Two Types of Round Trip

Due to the COVID-19 lockdown, Charly has time to devote to gadgets like graphical ping tools, flashing space stations, and space walks. *By Charly Kühnast*

Every now and then I notice tools that are useful, interesting, or ideally both, but sadly don't offer me enough material for a full-page article. This explains why there are two additions to the toolbox this month: gping [1] and Nextinspace [2].

## gping

A ping variant, gping graphically displays the determined round trip times (Figure 1). Admittedly, this is not exactly a new idea, but most tools of this kind require a graphical interface, while gping does its magic on the console, making it my tool of choice when I'm logged into a server via SSH.

First I need Cargo, the Rust package manager, because gping is written in Rust (Listing 1, first line). In addition, gping needs a few more components, in

particular *rustc*, but the package manager automatically fetches these as dependencies when you install Cargo.

Once everything is on board, it's time to set up gping (Listing 1, second line). It ends up in the `~/.cargo/bin/` directory. To use gping in a convenient way, either add this directory to `$PATH`, or create a symlink to a directory that exists in your `$PATH`.

Gping only supports a few parameters. IPv4 or IPv6 can be enforced with `-4` and `-6` respectively, while `-n 10` increases the ping interval from one to 10 seconds.

## Nextinspace

I like pointless but interesting gadgets and have more time on my hands than usual right now due to COVID-19.

What I wanted to do was build a small model space

station (e.g., made of Lego) where an LED flashes whenever the International Space Station passes over my home village. This requires a bit of software. During my research, I stumbled across Nextinspace, which didn't help me with my little project, but is interesting nonetheless. It notifies you of upcoming space-related projects around the world: rocket launches, satellite launches, spacewalks, and more (Figure 2).

Nextinspace, written in Python, can be beamed onto your device using the pip Python installer (Listing 2). If you then enter `nextinspace`, the next upcoming event is output. The `-v` switch brings additional information to light. Two other options help to classify the events: `-l` shows only rocket launches, while `-e` shows all other operations (but no launches). ■■■

## Info

[1] gping:  
<https://crates.io/crates/gping>

[2] Nextinspace:  
<https://pypi.org/project/nextinspace/>

## 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.



### Listing 1: Install gping

```
$ sudo apt-get -fym install cargo
$ cargo install gping
```

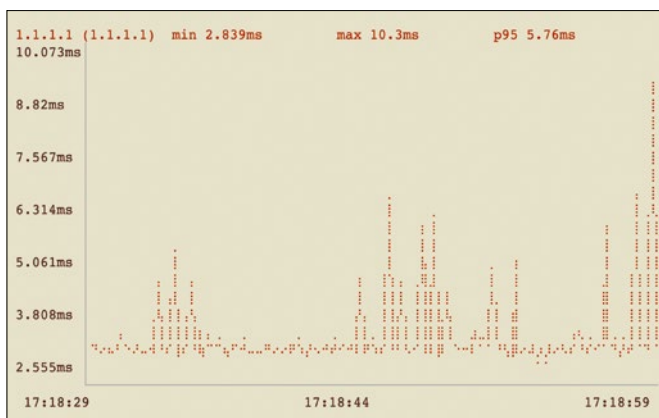


Figure 1: Gping: A graphical ping for the terminal.

### Listing 2: Installing Nextinspace

```
$ apt update
$ sudo apt-get install python3 python3-pip
$ pip install nextinspace
```

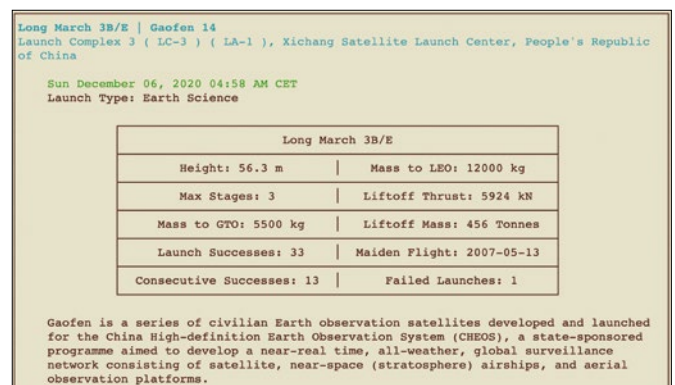


Figure 2: Nextinspace offers a roadmap for grounded astronauts.





Rendering images as text

Text Art

If you need to display an image in the terminal or as plain HTML, a variety of smart tools can help with the conversion. *By Frank Hofmann and Axel Beckert*

Thanks to increasingly sophisticated technology, displaying high-resolution images on screen is no longer a difficult task. However, these more detailed images (with a combination of greater image size, resolution, and color depth) come at a cost, consuming more storage space and taking longer to download from remote sources, such as web browsers and webcams.

Sometimes you just need an image to display quickly. You can save time and bandwidth by displaying images at a lower resolution and color depth as text (ASCII or Unicode characters) directly in the terminal and converting them with American National Standards (ANSI) color codes [1]. You can also convert the images to plain HTML and CSS and embed the results in a web page. Some

text browsers, such as ELinks, then display these images directly in the accessed web page. In a similar manner, the Browsht [2] browser does this internally and can also render images as text.

In this article, we will discuss the available tools for converting images to text and explore whether this approach is suitable for everyday use. This article follows up on a previously published article [3] that dealt with tools for creating ASCII art.

The Conversion

First, you need to convert an image into individual characters before embedding or displaying it. To do this, each pixel (or group of pixels) is assigned a suitably colored single letter, or more precisely a glyph (a graphic symbol).

You do the conversion via filters in the form of libraries. For example, you can use the `aview` and `asciiview` tools from the `Ascii Art Library (AALib)` [4] or `img2txt`

Authors

**Frank Hofmann** mostly works on the road as a developer, trainer, and author. His favorite places for working are Berlin, Geneva, and Cape Town. **Axel Beckert** is a Linux system administrator and network security specialist for ETH Zurich's IT services department. He is also involved with the Debian distribution, the Linux User Group Switzerland (LUGS), the Hacker-funk podcast, and various open source projects. Hofmann and Beckert are authors of *Debian Package Management*.

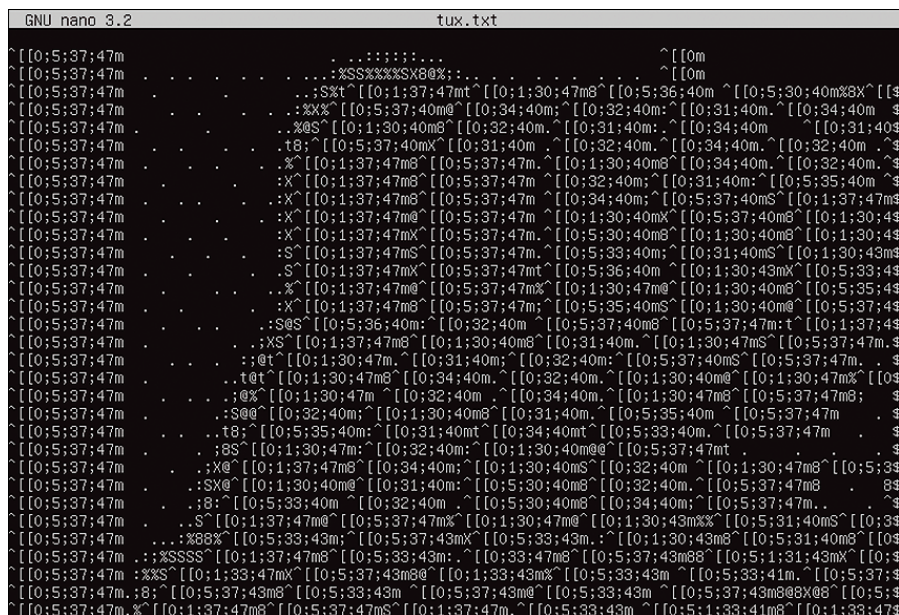


Figure 1: An excerpt of `tux.txt` shows a portion of the Tux logo in the form of ANSI control codes.

Lead Image © anan punyod, 123RF.com

**Table 1: img2txt Output Options**

Option	Description
ansi	ANSI with color codes
bbfr	BBCode [6]
caca	Internal libcaca format
html	HTML with support for CSS and DIV
html3	HTML with tables
irc	IRC with CTRL+K control codes
ps	PostScript
svg	Scalable Vector Graphics (SVG)
tga	Targa image format
utf8	UTF-8 with carriage return
utf8cr	UTF-8 with carriage return and line feed

and `cacaview` from the Colour Ascii Art Library (libcaca) [5]. The end result is a rendering of the image using letters with ANSI control codes. The resulting reduction in resolution and color depth not only reduces the volume of data to be transmitted, but it also means that the converted image can be displayed in a text terminal or text-based browser.

As an example, you can convert a PNG image of the Tux logo into letters with libcaca's `img2txt` (which replaces `imgtoppm` from previous library publications). To create a letter file named `tux.txt`, you use the following command:

```
$ img2txt tux.png > tux.txt
```

Figure 1 shows an excerpt from the content of the `tux.txt` file. Without additional switches in the call, the output text is 60 characters wide and formatted as colored ANSI.

You can use the `-W` (`--width`) option to regulate the width of the output. If you do not specify anything using `-H` (`--height`), `img2txt` scales the height of the output to match the original aspect ratio. The `-f` (`--format`) option lets you define the output format. Table 1 summarizes `img2txt`'s output options.

If you want to save ANSI images as normal images again, you can use the `AnsiLove` library [7]. By interpreting the ANSI codes, `AnsiLove` creates screenshots in PNG format, rather than forcing you to create a screenshot of the text terminal.

## Displaying Converted Images

For quite some time now, we have been experimenting with tools for displaying the converted images. In the Spring of

2020, Axel's toolbox [8] included `chafa` [9] and `aha` [10], along with the `catimg` [11] utility. Although the name is definitely a reference to the `cat/tac` commands, you might interpret `catimg` to be a tool for displaying cat pictures, but it can do considerably

more. During our research, `jp2a` [12], which works similarly to `img2txt`, also appeared on the scene. Functioning as image viewers for the terminal, `chafa` and `catimg` have only been an integral part of a stable release of the Linux distribution since Debian GNU/Linux 10. `chafa` displays one or more images as an unabridged slideshow in the terminal (Figure 2). It scales an image to match the current width and height of the terminal window. On the other hand, `catimg` orients an image based on its width resulting in the upper edge of the image disappearing from the terminal display during scrolling. The following two calls use `chafa` to show a single image and a slideshow of all PNG files in the current directory:

```
$ chafa linux.png
$ chafa *.png
```

`chafa` comes with a number of interesting options for effects. For example, you use `-c` (`--colors`) to set color mode to 2, 16, or 256 colors or to a 24-bit view mode. The `-d` (`--duration`) option determines how long an image remains in the slideshow (the default is three seconds). You can define the output size with `-s` (`--size`) `<width>x<height>`. By default, `chafa` uses the terminal's size or 80x25

characters if it cannot determine the size. The `--watch` option outputs the image again for each change. With `catimg`, you can use the `-l` option to control how often it plays an animated GIF.

Neither `chafa` nor `catimg` can handle crossfade effects to make the slideshows more interesting. Perhaps the developers will read this article and consider adding such a feature in the future.

## Differences

For all their similarities, the tools presented in this article have distinct peculiarities. For example, `aview` can only handle images in PNM format and only display images in grayscale since `AALib` does not support colors. `AALib`'s `asciiview` functions as a wrapper around `aview` that converts images to the PNM format required by `aview` up front using external tools. Animated GIFs do not work.

Unless you specify otherwise in the call, `img2txt` converts images to ASCII instead of Unicode glyphs and uses only 16 colors. Depending on the terminal, some characters might flash. Working similar to `img2txt`, `cacaview` is a plain vanilla image viewer, which opens a separate window to use the best possible terminal settings. As a result, nothing flashes here. However, animated GIFs do not work here either.

Both `chafa` and `catimg` display PNGs, JPEGs, GIFs, and many other image formats in more than 16 colors. If the terminal supports it, both can also use Unicode glyphs. Both can also handle animated GIFs. When displaying animated GIFs, the programs display the images in an endless loop instead of exiting by themselves.



**Figure 2:** Using `chafa` to show Tux as a text image.



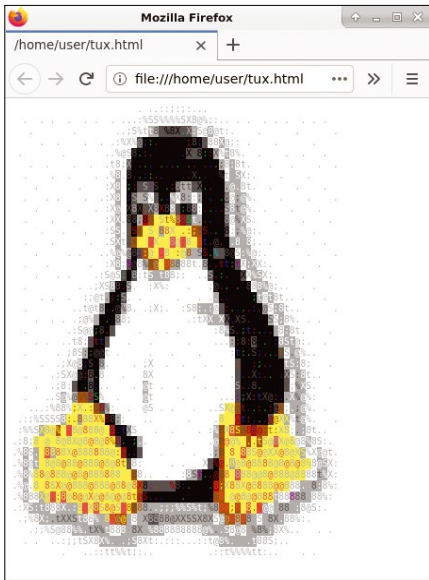


**Figure 3:** A text image of Linus Torvalds constrained to a width of 100 pixels with catimg.

Since catimg always uses the whole terminal width for display, you can use the -w option to specify the width of the output if necessary. Figure 3 shows this for a width of 100 pixels.

On the Web

If you're familiar with HTML code, you will have seen the HTML <img> tag. The code in Listing 1 references the tux.png graphics file and embeds it for display at the current location on a webpage.



**Figure 4:** Tux as a text image on a web page.

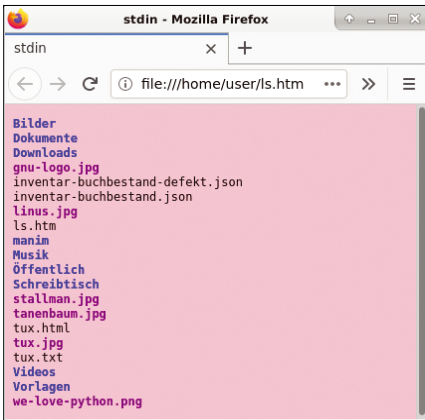
The alt attribute (which has been mandatory since 2011) specifies an image's alternative text that the web browser will display if the image cannot (or is not supposed to) load. For instance, if you view the website with a screen reader such as Orca [13], it will read out the alternative text.

The aha ANSI HTML adapter, img2txt, and jp2a (by specifying appropriate options) convert text with ANSI color codes (or images) into text-based HTML sequences and HTML tags for setting the color. You can then copy this output directly into your HTML file (Figure 4).

Why would you want to use text images on the web? You might want to integrate terminal content as a screenshot on a web page without having to use an image, saving data – this way the screenshot is more or less in the original format, namely text.

Figure 5 shows a directory listing with special pink background. Using the command sequence in Listing 2, you pipe the output of the ls command to aha, which gives it a pink background and outputs it to the ls.html file.

Theoretically, these results can also be read by a screen reader. In testing, how-



**Figure 5:** Terminal output as an image in the web browser.

Listing 1: Including an Image in HTML

```

```

Listing 2: HTML with Aha

```
$ ls --color=always | aha --pink > ls.html
```

ever, this currently only works reliably with black and white images. Presumably, the contrast is not high enough for the screen reader if you use other color combinations.

Videos

You can also use these techniques with videos. To do this, you use AALib and libcacaca as video output plugins for MPlayer and VLC (both), xine (AALib), and mpv [14] (libcacaca). The Hascicam project [15] renders images from a TV card or camera as an ASCII image.

The mpv player also comes with its own format, True Color Text (TCT). However, TCT's documentation is quite sparse; it is only mentioned in mpv's help page. (In fact, we were only able to conclude from comments in the source code the meaning of the TCT acronym.) More information on the subject would be useful.

Table 2 shows some sample commands for playing movie sequences; Figure 6 shows the matching output using aaxine.

The whole thing works quite well, but it does reach its limits if you need to view recorded keynote presentations, for example. If slides are embedded in the presentation recording, the text on the slides is often unreadable. To read these slides, you would need OCR capabilities in addition to the ability to reduce the resolution and convert to a different format.

QR Codes

What works with images and videos also works with QR codes. Put simply, QR codes are actually just special images that can be displayed at the command line. The advantage here is that you don't have to render the QR code's rough "pixels" as glyphs. You simply need to convert each of the blocks into an empty space or a half or whole Unicode block character.

This means that QR codes look just as crisp on a text-based terminal as they do

**Table 2:** Converting Videos with AALib and libcacaca

Program	Sample Command
mpv with TCT library	mpv --vo=tct https://youtu.be/Qd_1t7kw5EA
mpv with libcacaca	mpv --vo=caca https://youtu.be/Qd_1t7kw5EA
MPlayer with AALib	mplayer -vo aa video.mp4
xine with AALib	aaxine video.mp4





Figure 6: Using aaxine to play a video.



Figure 7: The foobar string as a QR code in text form.

as real images. The same applies to the text representation on web pages. Helpful tools for doing this include qrencode [16] and qrcode [17] from the Debian *go-qrcode* package. Figure 7 shows the foobar string as a QR code in text form.

A possible application for this is provided by the pass-otp one-time password plugin [18] in the pass [19] password manager. With the help of qrencode and qrcode, secrets transferred via QR code

can also be output as QR codes – even on a text-based terminal.

## Conclusions

A few last suggestions for interesting applications: you might want to take a look at asciinema [20], MapSCII [21], and ASCIIQuarium [22].

With asciinema, you can record terminal sessions as video. In addition, the project website acts as a YouTube replacement and preserves your recorded sequences in 8-bit. MapSCII (Figure 8), a digital atlas based on OpenStreetMap, offers a zoom function (Figure 8): Use A to zoom in, Z to zoom out, and the arrow keys to navigate. The coordi-

nates of the image's center at the bottom of the screen help with orientation.

For a little entertainment at the terminal, try the ASCIIQuarium (Figure 9), which turns your terminal into a virtual aquarium. Happy fishing! ■■■

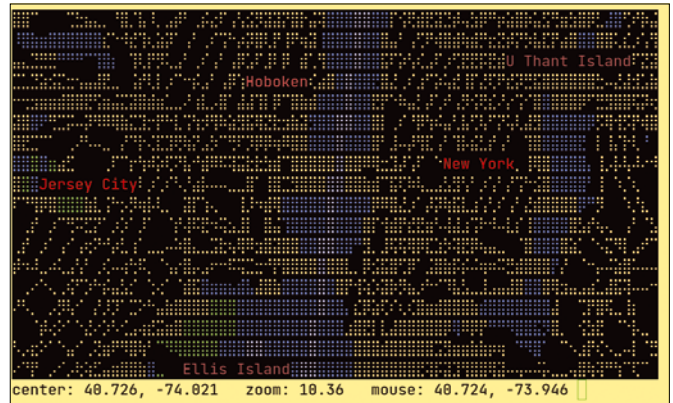


Figure 8: MapSCII is an ASCII-based digital atlas with zoom functions.

## Info

- [1] Escape Sequences: <https://www.nayab.xyz/linux/escapecodes.html>
- [2] "Call web pages in the terminal with Browsh" by Tim Schürmann, *ADMIN Magazine*, issue 53, 2019, <https://www.admin-magazine.com/Archive/2019/53/Call-web-pages-in-the-terminal-with-Browsh>
- [3] "Creating artistic images with ASCII art" by Frank Hofmann and Thomas Winde, *Linux Magazine*, issue 168, November 2014, <https://www.linux-magazine.com/Issues/2014/168/ASCII-Art/language/eng-US>
- [4] AALib: <http://aa-project.sourceforge.net/aalib/>
- [5] libcacat: <http://caca.zoy.org/wiki/libcacat>
- [6] BBCode: <https://en.wikipedia.org/wiki/BBCode>
- [7] AnsiLove: <https://www.ansilove.org/>
- [8] "Pictures in pure HTML with chafa and aha" by Axel Beckett: <http://noone.org/blog/English/Computer/Pictures%20in%20pure%20HTML%20with%20chafa%20and%20aha.futile>
- [9] chafa: <https://hpjansson.org/chafa/>
- [10] aha: <https://github.com/theZiz/aha>
- [11] catimg: <https://github.com/posva/catimg>
- [12] jp2a: <https://github.com/cslarsen/jp2a>
- [13] Orca: <https://help.gnome.org/users/orca/stable/>
- [14] mpv: <https://mpv.io>
- [15] Hasciicam: <http://ascii.dyne.org>
- [16] qrencode: <https://fukuchi.org/works/qrencode/>
- [17] qrcode: <https://github.com/skip2/go-qrcode>
- [18] pass-otp: <https://github.com/tadfisher/pass-otp>
- [19] pass: <https://www.passwordstore.org/>
- [20] asciinema: <https://asciinema.org/>
- [21] MapSCII: <https://mapscii.me/>
- [22] ASCIIQuarium: <https://robobunny.com/projects/asciiquarium/html/>

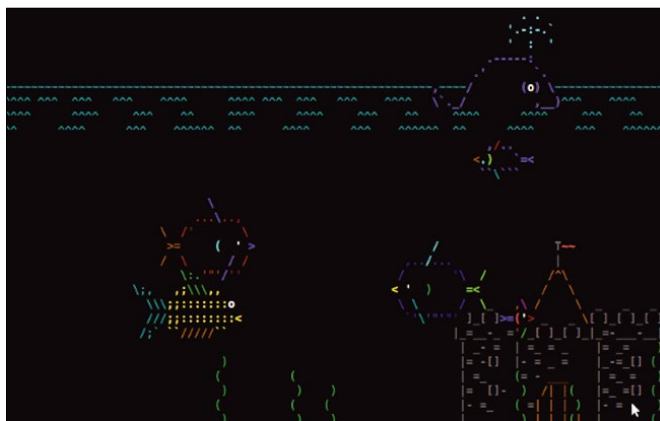


Figure 9: ASCIIQuarium puts diverse aquatic species on your terminal.



An efficient command-line email client

# Mutt for Beginners

Mutt, a command-line email client, can do anything a desktop client can with less overhead and a smaller attack surface. Here's how to get started. *By Bruce Byfield*

I subscribe to half a dozen forums for email distribution. Hardly a week goes by without one of them including a complaint about a desktop email client like KMail or Thunderbird. For many, the ultimate solution is to go back to simpler days and install Mutt [1]. Not only does the command-line interface give users full control over the settings, but Mutt's lack of a prepackaged rendering engine for web browsers or a JavaScript interpreter makes for a smaller attack surface. Another major advantage is its smaller, more consistent memory usage.

Mutt was first written by Michael Elkins in 1995. It was based on Elm, an-

other popular command-line email client. Even today, Mutt remains very much a product of its time, often using other applications rather than adding functionality to its code. In particular, emails are composed in the external editor of your choice, while encryption depends on GnuPG [2]. In addition, Mutt is fully operable from the keyboard alone. You can tell a lot about Mutt's design philosophy by the slogan it has carried from its earliest days: "All mail clients suck. This one just sucks less." Efficiency and economy are very much its priority.

However, Mutt does require some setup. This article covers the minimal information you need to have Mutt up and running smoothly in a typical case.

Should you have an atypical case, the man page for muttrc [3], Mutt's configuration file, lists dozens of possible alternative settings.

## Basic Configuration

In deference to its age, Mutt is included in the repositories of most distributions. Preparing it for use consists primarily of

editing its configuration files by adding commands and a few fields. To begin configuring, create the basic directories and the configuration file:

```
mkdir -p ~/.mutt/cache/headers
mkdir ~/.mutt/cache/bodies
touch ~/.mutt/certificates
```

The configuration file, muttrc, can have several locations that are detected automatically: ~/.muttrc, ~/.mutt/muttrc, and \$XDG\_CONFIG\_HOME/mutt/muttrc, each with or without -MUTT\_VERSION appended. Use touch to create the muttrc file with the path of your choice. For example, you can use:

```
touch ~/.mutt/muttrc
```

If you want to place muttrc in a nonstandard place, you can set the location by adding to muttrc the line:

```
source /path/to/other/config/file
```

## 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/>.

Next, open the newly made `muttrc` in a text editor. As with most configuration files, start a line with `#` to add a comment, and use single or double quotation marks around an entry that contains spaces or special characters. Add the following lines to set Mutt's environment:

```
set realname = "NAME"
set from = "YOUR EMAIL ADDRESS"
set use_from = yes
set envelope_from = yes
set editor = "EDITOR"
set charset = "CHARACTER-CODE"
```

If you are using an IMAP server, add:

```
set smtp_url = "EMAIL-ADDRESS:PORT/"
set smtp_pass = "PASSWORD"
set imap_pass = "PASSWORD"
set folder = "PATH:PORT"
set spoolfile = "+INBOX"
set record = +Sent
mailboxes = +INBOX
bind index imap-fetch-mail
```

For `folder`, use the directory where messages are stored; `spoolfile` is where Mutt looks for incoming mail. The port is only needed if the folder is not local. The plus sign indicates that any subdirectories will be used as necessary.

Finally, set the `mbox` type and the structure for receiving messages as follows

```
set mbox_type=Maildir [or Mbox]
set folder=~/.mail
set spoolfile=+/
set header_cache=~/.cache/mutt
```

where `spoolfile` should be the same as the spool file set for IMAP; `header_cache` stores email headers to increase the speed in which headers are displayed.

These lines of code will set up a bare-bones Mutt configuration with one or two additions for convenience. Before you go any further, send a message to check whether you have basic functionality. If you made any typos during setup, it will be easier to troubleshoot before you add more to your configuration.

## Setting Passwords

Account passwords can be added with:

```
set my_pass = "PASSWORD"
```

The prefix `my_` is used for any variables that you define. However, `muttrc` is unencrypted, so storing the password in it leaves your password visible to anyone. The simplest alternative is to enter your password each time you login. Better yet, set up encryption for it by adding

```
set my_pass = "PASSWORD"
```

to a new file. Other passwords can also be added to this file. Assuming that the email is encrypted with GPG, at the start of `.muttrc`, add the line

```
source "gpg -dq FILE |"
```

This line sets the variable for all password commands.

## Contact Management

Mutt has two ways of setting up an address book. The first is to create aliases for each contact in a new file, one alias per line, with the structure:

```
alias NICKNAME LONGNAME ADDRESS
```

The optional `LONGNAME` is the contact's full name. In Mutt, you use the `NICKNAME` to send an email to the contact. Pressing `a` when an address is entered in the `To:` field will also create a new file when aliases are set up in `muttrc` with the lines:

```
set alias_file = "PATH"
set sort_alias = alias
set reverse_alias = yes
source $alias_file
```

The alias file is where aliases are stored. The sort file determines whether aliases are listed by alias or address, while `reverse_alias` set to `yes` displays the long name if one is given. Adding the source allows Mutt to autocomplete when you enter an alias for the `To:` field. If the alias you enter is nonexistent, then a list of all aliases displays.

If you want a more sophisticated address book, you can use an external application such as Abook, GooBook, or Khard.

## Enhancements

To enhance your use of Mutt, you will probably

want to add a few other commands to `muttrc`. For example,

```
set signature="PATH"
```

points to a file that contains a signature that is added automatically to the end of every email you send. To avoid any conflict between Mutt's use of UTF-8 for a character set and the editor in which you write emails, you should also add:

```
set send_charset="utf-8"
```

However, by far the most useful step is to set up encryption for sent mail. If you have not already done so, create `~/.mutt/gpg.rc`, then copy to it the file `/usr/share/doc/mutt/samples/gpg.rc`, and add the following line

```
source ~/.mutt/gpg.rc
```

to `muttrc`.

With this setup, you can press `p` when composing to use basic GnuPG options. The `muttrc` man page [3] lists additional encryption options.

## Sending Email

All emails sent from Mutt are in plain text for security. However, if you want an HTML message, either compose it in a separate email editor or add the tags manually.

Emails can be sent in two ways. First, you can send an email directly from the command line (see Table 1 for command options), with:

```
mutt OPTIONS"RECIPIENT-OR-ALIAS"
```

When you press the Enter key, Mutt asks for confirmation of the options and then opens in the default editor so that you can type the message (Figure 1).

The `-R` option lets you open a mailbox and select a message to reply to. If you are unsure of the available mailboxes, typing `-y` will provide a list of available

**Table 1: Command-Line Email Options**

<code>-a FILE</code>	Attach a file
<code>-b ADDRESS</code>	Add a blind carbon copy (BCC)
<code>-c ADDRESS</code>	Add a carbon copy (CC) recipient
<code>-i FILE</code>	Include a file in the body of the email
<code>-s SUBJECT</code>	Add the subject of the message



ones. If necessary, `-f MAILBOX` sets the current mailbox.

An easier way to use Mutt is to type the basic command of mutt, which opens a text-based interface (Figure 2).

The interface is entirely mouse driven, with a list of available actions along the top, and a summary of the current mailbox along the bottom. When Mutt runs from the command line, pressing `m` to

start a message runs you through a series of prompts for the headers and then opens Mutt's default editor.

Regardless of whether you are running from the command line or the text interface, when you are finished writing your message in the editor, save the file and quit the editor (the exact commands for doing so depend on the editor). A screen appears in which you can make last minute changes, using the options listed at the top of the page, with the message described as the attachment of a file stored in `/tmp/mutt` (Figure 3). Press `y` when you are ready to send the message.

```

TW /tmp/mutt-nanday-1000-5293-7860455411086450003 (Modified)(mail) Row 6 Col 8
Hi, John:

Do you still have the rose-breasted cockatoo you advertised recently? If so,
what is your asking price?

Thanks,

--
Bruce Byfield 604.421.7189 (Pacific time)
Writer of "Designing with LibreOffice"
http://designingwithlibreoffice.com/

```

**Figure 1:** Mutt can use any command-line editor, including Vim, Emacs, Nano, or Joe (shown here).

```

q:Quit u:Undel u:Undel s:Save m:Mail r:Reply g:Group f:Help
1870 0 Dec 06 Cron Daemon ( 2) Cron <root@nanday> /usr/lib/prey/prey.sh >/var/log/prey.l
1872 0 Dec 14 Anacron ( 1) Anacron job 'trim.weekly' on nanday
1873 0 Dec 19 Anacron ( 23) Anacron job 'cron.monthly' on nanday
1874 0 Dec 21 Anacron ( 1) Anacron job 'trim.weekly' on nanday
1875 0 Dec 28 Anacron ( 1) Anacron job 'trim.weekly' on nanday
1876 0 Jan 04 Anacron ( 1) Anacron job 'trim.weekly' on nanday
1877 0 Jan 11 Anacron ( 1) Anacron job 'trim.weekly' on nanday
1878 0 Jan 15 Cron Daemon ( 4) Cron <root@nanday> /usr/lib/prey/prey.sh >/var/log/prey.l
1879 0 Jan 18 Anacron ( 1) Anacron job 'trim.weekly' on nanday
1880 0 Jan 19 Anacron ( 23) Anacron job 'cron.monthly' on nanday
1881 N + Jan 20 Mail Delivery S ( 45) Mail delivery failed: returning message to sender
1882 N Jan 25 Anacron ( 1) Anacron job 'trim.weekly' on nanday
1883 N + Jan 25 Mail Delivery S ( 46) Mail delivery failed: returning message to sender
1884 N + Jan 25 Mail Delivery S ( 51) Mail delivery failed: returning message to sender

--Mutt: /var/mail/bb [Msgs:1884 New:4 Old:1878 3.6M]--(threads/date)----- (end)---

```

**Figure 2:** Mutt can be run from the command line, or, more conveniently, through a keyboard-navigated text interface.

```

y:Send q:Abort t:To c:CC s:Subj a:Attach file d:Descrip ?:Help
From: bb <bb@nanday.nanday>
To: bruce.byfield@gmail.com
Cc:
Bcc:
Subject: Test
Reply-To:
Fcc: ~/sent
Mix: <no chain defined>
Security: None

-- Attachments
I 1 /tmp/mutt-nanday-1000-5293-7860455411086 [text/plain, 7bit, us-ascii, 0.2K]

-- Mutt: Compose [Approx. msg size: 0.2K Atts: 1]-----

```

**Figure 3:** Before you send an email, Mutt gives you one last chance to edit the headers and displays a summary of the email.

## Other Configuration

The information provided will get you up and running, but it is only a fraction of what Mutt can do. You can add small touches, such as a sidebar that lists mailboxes or configuring Mutt to use Gmail, or more elaborate ones like encrypting email, displaying messages in threads, using custom headers, or running Mutt for local system messages. For additional useful information, see the Mutt documentation [4] and the ArchWiki notes on Mutt [5]. You may have to do some digging, but, to the best of my knowledge, Mutt can do anything the average desktop mail client can do – and with considerably less overhead or problems. ■■■

## Info

- [1] Mutt: <http://www.mutt.org/>
- [2] GnuPg: <https://gnupg.org/>
- [3] muttrc: <https://linux.die.net/man/5/muttrc>
- [4] Mutt documentation: <http://www.mutt.org/doc/manual/>
- [5] ArchWiki notes: <https://wiki.archlinux.org/index.php/Mutt#Configuration>



# 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)





Getting started with the ELK Stack monitoring solution

# Elk Hunting

ELK Stack is a powerful monitoring system known for efficient log management and versatile visualization. This hands-on workshop will help you take your first steps with setting up your own ELK Stack monitoring solution. *By Tomasz Szandala*

**T**oday's networks require a monitoring solution with industrial-strength log management and analytics. One option that has gained popularity in recent years is ELK stack [1]. The free and open source ELK Stack collection is maintained by a company called Elastic. (According to the website, the company has recently

changed the name of the project to Elastic Stack, but the previous name is still in common usage.) ELK Stack is not a single tool but a collection of tools (Figure 1). The ELK acronym highlights the importance of the collection's three most important utilities. At the heart of the stack, Elasticsearch collects and maintains data, providing an engine, based

on Apache Lucene, for searching through it. Logstash serves as the log processing pipeline, collecting data from a multitude of sources, transforming it, then sending it to a chosen "stash." (Keep in mind that, despite its name, Logstash itself does not preserve any data.) Kibana provides a user-friendly interface for querying and visualizing the data.

A bundle of tiny apps called beats specialize in collecting data and feeding it to Logstash or Elasticsearch. The beats include:

- Filebeat – probably the most popular and commonly used member of the beats family. Filebeat is a log shipper that assigns subordinates, called harvesters, for each log to be read and fed into Logstash.
- Heartbeat – an app that asks a simple question: Are you alive? Then it ships this information and response time to Elasticsearch. In other words it is a more advanced ping.
- Winlogbeat – is used for monitoring a Windows-based infrastructure. Win-

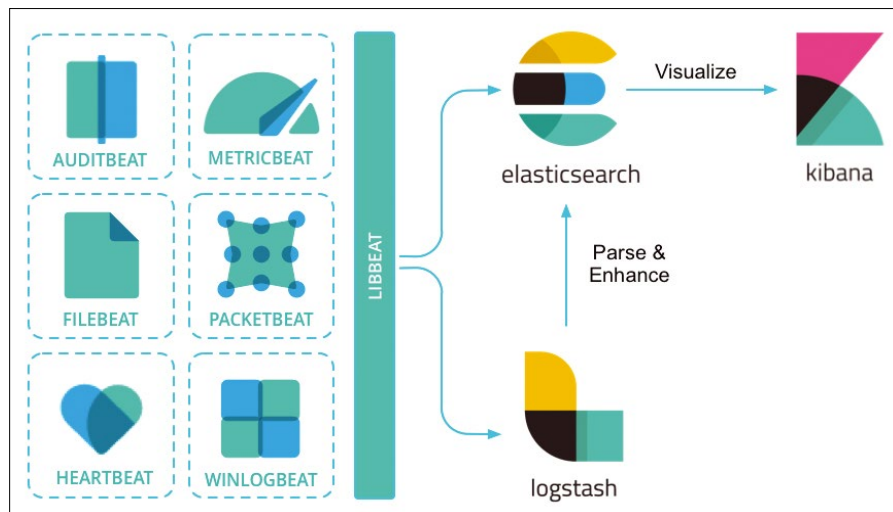


Figure 1: The ELK family and its relatives.

Photo by David Santoyo on Unsplash



logbeat streams Windows event logs to Elasticsearch and Logstash.

- **Metricbeat** – collects metrics from your systems and services. Metrics include CPU and memory disk storage, as well as data for Redis, Nginx, and much more. Metricbeat is a lightweight way to collect system and service data.

The collection also comes with several plugins that enhance functionality for the entire stack.

ELK Stack is popular in today's distributed environments because of its strong support for log management and analytics. Before you roll out a solution as complex and powerful as ELK Stack, though, you'll want to start by trying it out and experimenting with it in a test environment. It is easy to find overviews and short intros to ELK Stack, but it is a little more difficult to study the details. This workshop is a hands-on look at what it takes to get ELK Stack up and running.

## ELK Installation

ELK Stack has lots of pieces, so it helps to use an automated deployment and configuration tool for the installation. I will use Ansible in this example. I hope to write this in a simple way that will be easy to follow even if you aren't familiar with Ansible, but see the Ansible

project website [2] if you need additional information.

Listing 1 shows an Ansible playbook for installing the ELK Stack base applications. The first few lines define a few settings specific to Ansible itself, such as declaring that the execution will be local (and won't require an SSH network connection). `become: true` asks Ansible to run all commands with Sudo, which will allow you to run this playbook as a default Vagrant user instead of relogging to root. The tasks section lists the steps that will be executed in the playbook. There are multiple ways to install ELK Stack; Listing 1 uses the yum package manager and specifies a package repository. I specify the exact version numbers for the Elasticsearch, Logstash, and Kibana packages to make it easier to install the correct plugins later.

Once the software is installed, you need to run it as a service. You could use `systemctl`, but Listing 2 carries on using Ansible.

The command in Listing 3 checks to ensure that Elasticsearch is running locally at the default port 9200.

## Configuring ELK

ELK Stack is set up in a virtual machine and is only listening on localhost, so if you try to open Kibana or Elasticsearch in the host's browser, it won't work. You

need to change the `network.host` setting in the YAML file to `0.0.0.0` to enable network operations.

The Elasticsearch YAML file is usually `/etc/elasticsearch/elasticsearch.yml` and Kibana and Logstash follow the same pattern. (The YAML config files installed with the RPM packages are quite verbose, though many of the settings are commented out.)

The most important change is to set `network.host` to `0.0.0.0`. Keep in mind that Elasticsearch considers this change as enabling a production environment, therefore ELK Stack will expect a production environment to be running in a cluster. And since I am working in a single-node cluster, I need to set the value `discovery.seed_hosts: []` – an empty list, in order to disable cluster discovery features.

The same applies to the Kibana dashboard. You need to modify the value `server.hosts` to `0.0.0.0` and restart the service.

You can use Ansible to help you get the default config YAML files for Kibana and ES (Listing 4). Store them in the `files` subdirectory of the playbook directory. Then you can make the required updates and use Ansible to replace the files. You'll need to restart the service if you make changes to the configuration.

**Listing 1: Ansible Playbook: elk-setup.yml**

```
01 ---
02 - hosts: localhost
03   connection: local
04   gather_facts: false
05   become: true
06   tasks:
07     - name: Add Elasticsearch OpenSource repo
08       yum_repository:
09         name: Elasticsearch-OS
10         baseurl: https://artifacts.elastic.co/packages/
11               oss-7.x/yum
12         description: ELK OpenSource repo
13         gpgcheck: false
14     - name: Install ELK stack
15       yum:
16         name: "{{ item }}"
17       loop:
18         - elasticsearch-oss-7.8.0-1
19         - logstash-oss-7.8.0-1
20         - kibana-oss-7.8.0-1
```

**Listing 2: Are Elasticsearch and Kibana Enabled?**

```
01 - name: Start ELK services
02   service:
03     name: "{{ item }}"
04     enabled: true
05     state: started
06   loop:
07     - elasticsearch
08     - kibana
```

**Listing 3: Is Elasticsearch Running?**

```
01 [vagrant@ELK ~]$ curl localhost:9200
02 {
03   "name" : "ELK",
04   "cluster_name" : "elasticsearch",
05   "version" : {
06     "number" : "7.8.0",
07     "minimum_wire_compatibility_version" : "6.8.0",
08     "minimum_index_compatibility_version" : "6.0.0-beta1"
09   },
10   "tagline" : "You Know, for Search"
11 }
```

**Listing 4: elk-setup.yml: Getting the Files**

```

01 - name: Copy file with Elasticsearch config
02   copy:
03     src: files/elasticsearch.yml
04     dest: /etc/elasticsearch/elasticsearch.yml
05     owner: root
06     group: elasticsearch
07     mode: '0660'
08   notify: restart_elasticsearch
09
10 - name: Copy file with Kibana config
11   copy:
12     src: files/kibana.yml
13     dest: /etc/kibana/kibana.yml
14     owner: root
15     group: kibana
16     mode: '0660'
17   notify: restart_kibana
18
19 handlers:
20 - name: Restart Elasticsearch
21   service:
22     name: elasticsearch
23     state: restarted
24     listen: restart_elasticsearch
25
26 - name: Restart Kibana
27   service:
28     name: kibana
29     state: restarted
30     listen: restart_kibana

```

**Listing 5: Provision a Monitored Node**

```

01 ---
02 # ...
03 tasks:
04 - name: Add epel-release repo
05   yum:
06     name: epel-release
07     state: present
08
09 - name: Install Nginx
10   yum:
11     name: nginx
12     state: present
13
14 - name: Insert Index Page
15   template:
16     src: index.html.j2
17     dest: /usr/share/nginx/html/index.html
18
19 - name: Start Nginx
20   service:
21     name: nginx
22     state: started

```

Listing 4 uses a notify directive to create notifications that will be monitored in the handlers section.

**Collecting Data with Beats**

Now that the ELK services are up and running, I'll show you how to use Metricbeat and Filebeat to collect data. As I mentioned previously, Metricbeat is designed to collect system and service metrics, and Filebeat collects data from logfiles.

The first step is to set up a dummy Nginx application that will serve as a monitored node (Listing 5).

Most of the tasks in Listing 5 are self-explanatory except the third one, which takes a local file with jinja2 formatting and renders it into the chosen destination format. In this case, I insert a hostname to display it on an HTTP page (Listing 6).

**Listing 6: index.html.j2: Minimal HTML File**

```

01 <!doctype html>
02 <html>
03   <head>
04     <title>{{ hostname }} dummy page</title>
05   </head>
06   <body>
07     <h1>Host {{ hostname }}</h1>
08     <p>Welcomes You</p>
09   </body>
10 </html>

```

**Listing 7: metricbeat.yml**

```

01 metricbeat.modules:
02 - module: system
03   period: 30s
04   metricsets:
05     - cpu          # CPU usage
06     - load          # CPU load averages
07     - service       # systemd service information
08   # Configure the metric types that are included by these
    metricsets.
09   cpu.metrics: ["percentages", "normalized_percentages"]
10 - module: nginx
11   metricsets: ["stubstatus"]
12   period: 10s
13   hosts:
14     - "http://127.0.0.1"
15   server_status_path: "/nginx_status"
16 tags:
17 - slave
18 - test
19 #fields:
20 #  hostname: ${HOSTNAME:?Missing hostname env variable}
21 processors:
22 - fingerprint:
23     fields: ['.*']
24     ignore_missing: true
25 output.elasticsearch.hosts: ["172.22.222.222:9200"]
26 setup.kibana.host: "http://172.22.222.222:5601"
27 setup.dashboards.enabled: true

```

**Listing 8: nginx.conf Excerpt**

```

21     location /nginx_status {
22         stub_status on;
23         access_log off;
24         allow 127.0.0.1;
25         allow ::1;
26         deny all;
27     }

```

I'll use Metricbeat to collect statistics on the monitored node. The YAML file in Listing 7 shows a Metricbeat configuration file that will collect data on the CPU, RAM, disk usage, and a few other metrics.

Metricbeat supports several different modules dedicated to monitoring different services. One of the most commonly used modules is the `system` module,

which collects metrics related to the system. Some of the metrics have individual configuration settings, such as `cpu` and `core`, which you can see in lines 21-22.

The Metricbeat config file contains three sections: `tags`, `fields`, and `processors`. The `tags` section adds new list-type fields. In the `fields` section, you can append key-value entries to send to JSON.

Beats environment variables behave like environment variables in Bash and take the form `${VAR_NAME}`. You can provide a default value to use if no other value is found with `${VAR_NAME:some_default_value}`. To enforce the presence of the variable, use `${VAR_NAME:?error message}`, in which case Metricbeat will fail to start and log an error message if the environment variable is not found. The most advanced modifiers are in the `processors` section. Processor settings can dynamically adjust to events, in this case: compute fingerprints from chosen fields. There are many variations of processors that perform tasks such as conditionally adding or removing fields or even executing simple JavaScript snippets that modify our event data.

Another popular module for metrics collection is Nginx, which collects numbers from the Nginx status page. However

before you can use the Nginx module, you need to enable the status page for scraping.

Listing 8 shows the section of the `nginx.conf` configuration file that will enable metrics and configure security so that attempts to reach the status page must come from the host itself. Because the scraper will collect metrics every few seconds, there is no point in logging each entry in `access_log`, therefore the `access_log` setting is turned off.

Listing 9 shows the Ansible playbook section that deploys Nginx and Metricbeat.

**Log Management**

Filebeat attends to tasks related to log collection. Filebeat has some built-in parsers for commonly recognized logfile types, such as `syslog`, `Nginx logs`, and a few more.

The `filebeat.yml` file in Listing 10 shows two modules for handling system logs and Nginx logs. Both modules are built into the base Filebeat application and provide functionality to break the lines of the log into events that can be sent directly to Elasticsearch.

Listing 11 shows an event stored to Elasticsearch by Filebeat (with some insignificant parts removed for brevity). This sample event originates from a log entry like the following:

```

Sep 28 13:49:07 slave0 sudo[17900]:  vagrant : TTY=pts/1 ;  PWD=/home/vagrant ; USER=root ;  COMMAND=/bin/vim  /etc/filebeat/filebeat.yml

```

Keep in mind that Filebeat is designed to work with structured and known log

**Listing 9: Deploying Nginx and Metricbeat**

```

01 (...)
02 - name: Copy Nginx config
03   copy:
04     src: nginx.conf
05     dest: /etc/nginx/nginx.conf
06     owner: root
07     group: root
08     mode: '0644'
09   notify: restart_nginx
10
11 - name: Install Beats
12   yum:
13     name: "{{ item }}"
14     loop:
15       - metricbeat-7.8.0-1
16       - filebeat-7.8.0-1
17
18 - name: Start Beats services
19   service:
20     name: "{{ item }}"
21     enabled: true
22     state: started
23   loop:
24     - metricbeat
25     - filebeat
26
27 - name: Copy file with Metricbeat config
28   copy:
29     src: metricbeat.yml
30     dest: /etc/metricbeat/metricbeat.yml
31     owner: root
32     group: root
33     mode: '0644'
34   notify: restart_metricbeat

```

**Listing 10: filebeat.yml**

```

01 filebeat.modules:
02 - module: nginx
03   access:
04     var.paths: ["/var/log/nginx/access.log"]
05   error:
06     var.paths: ["/var/log/nginx/error.log"]
07 - module: system
08   syslog:
09     var.paths: ["/var/log/messages"]
10   auth:
11     var.paths: ["/var/log/secure"]
12   setup.kibana.host: "http://172.22.222.222:5601"
13   setup.dashboards.enabled: true
14   output.elasticsearch.hosts: ["172.22.222.222:9200"]

```



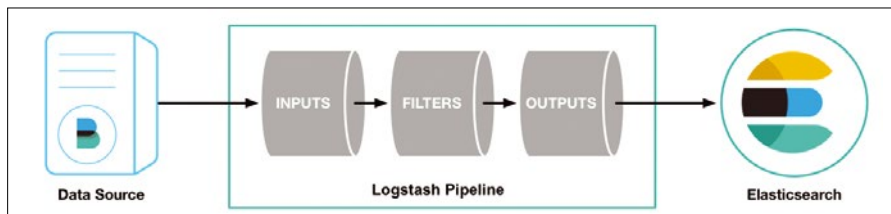


Figure 2: Logstash processing flow.

#### Listing 11: Store in Elasticsearch via Filebeat

```
01 $ curl -s "localhost:9200/filebeat-7.8.0-2020.09.28/_
    search?pretty=true&sort=@timestamp&size=1"
02
03 "_source" : {
04   "agent" : {
05     "hostname" : "slave0",
06     "type" : "filebeat"
07   },
08   "process" : {
09     "name" : "sudo",
10     "pid" : 17900
11   },
12   "log" : {
13     "file" : {
14       "path" : "/var/log/secure"
15     }
16   },
17   "fileset" : {
18     "name" : "auth"
19   },
20   "input" : {
21     "type" : "log"
22   },
23   "@timestamp" : "2020-09-28T13:49:07.000Z",
24   "system" : {
25     "auth" : {
26       "sudo" : {
27         "tty" : "pts/1",
28         "pwd" : "/home/vagrant",
29         "user" : "root",
30         "command" : "/bin/vim /etc/filebeat/filebeat.yml"
31       }
32     }
33   },
34   "related" : {
35     "user" : [
36       "vagrant"
37     ]
38   },
39   "service" : {
40     "type" : "system"
41   },
42   "host" : {
43     "hostname" : "slave0"
44   }
45 }
```

types. If you aim to track unspecified logs, you need to use Logstash.

## Logstash

In order to track and ship unstructured logs, you have to use a simple log module, and its output should mainly go through Logstash, unless you are

#### Listing 12: Exceptions from logstash.yml

```
01 $ cat files/logstash.yml | grep -vP '^#'
02 path.data: /var/lib/logstash
03 pipeline.id: main
04 path.config: "/etc/logstash/conf.d/pipeline.conf"
05 http.host: 0.0.0.0
```

#### Listing 13: pipeline.conf

```
01 input {
02   beats {
03     port => 5044
04   }
05 }
06 filter {
07   if [fileset][module] == "system" {
08     if [fileset][name] == "auth" {
09     (...)
10   }
11   else if [fileset][name] == "syslog" {
12     grok {
13       match => {
14         "message" => ["%{SYSLOGTIMESTAMP:[system][syslog]
15                       [timestamp]} %{SYSLOGHOST:[system]
16                       [syslog][hostname]} \
17                       %{DATA:[system][syslog][program]}(?:\
18                       [%{POSINT:[system][syslog][pid]})?: \
19                       %{GREEDYMULTILINE:[system][syslog]
20                       [message]}"]
21       }
22     }
23   }
24   else if [fileset][module] == "nginx" {
25     (...)
26   }
27 }
28 output {
29   elasticsearch {
30     hosts => localhost
31     manage_template => false
32     index => "logstash-%{[@metadata][beat]}-%{[@
33               metadata][version]}-%{+YYYY.MM.dd}"
34   }
```

only interested in the timestamp and message fields. You need Logstash to act as an aggregator for multiple logging pipelines.

Despite its name, Logstash does not stash any logs. It receives log data and processes it with filters (Figure 2). In this case, *processing* means transforming, by removing unnecessary elements or splitting objects into terms that can serve as JSON objects and be sent to a tool like Elasticsearch. When the logs are processed, you have to define the output: The output could go to Elasticsearch, a file, a monitoring tool like StatsD, or one of several other output options.

The `logstash.yml` file shown in Listing 12 only presents non-default values. You need to define the data and specify where logstash will keep temporary data.

Pipelines are sets of input/filter/output rules. You can define multiple pipelines and list them in file `pipelines.yml`. When you have a single pipeline, you can specify it directly in the main config. The `pipeline.conf` file (Listing 13) lets you specify the input, filter, and output for the pipeline. The input options include reading from a file, listening on port 514 for syslog messages, reading from a Redis server, and processing events sent by beats. A pipeline can also receive input from services such as the Cloudwatch monitoring tool or the RabbitMQ message broker.

Filters transform data in various ways to prepare it for later storage or processing. Some popular filter options include:

- `grok` filter – transforms unstructured lines into structured data.
- `csv` – converts csv content into a list of elements.
- `geoip` – assigns geographic coordinates to a given IP addresses.

The output settings define where the data goes after filtering, which might be to Elasticsearch, email, a local file, or a database.

As you can see in Listing 13, the sample pipeline starts with waiting for beat input on port 5044. The most complex part of Listing 13 is the filter. In this case, the filter will parse syslog, audit, log, Nginx, and error logs, and each log has different syntax.

Most filters are self-explanatory, but `grok` [3] requires a comment: it is a plugin that modifies information in one

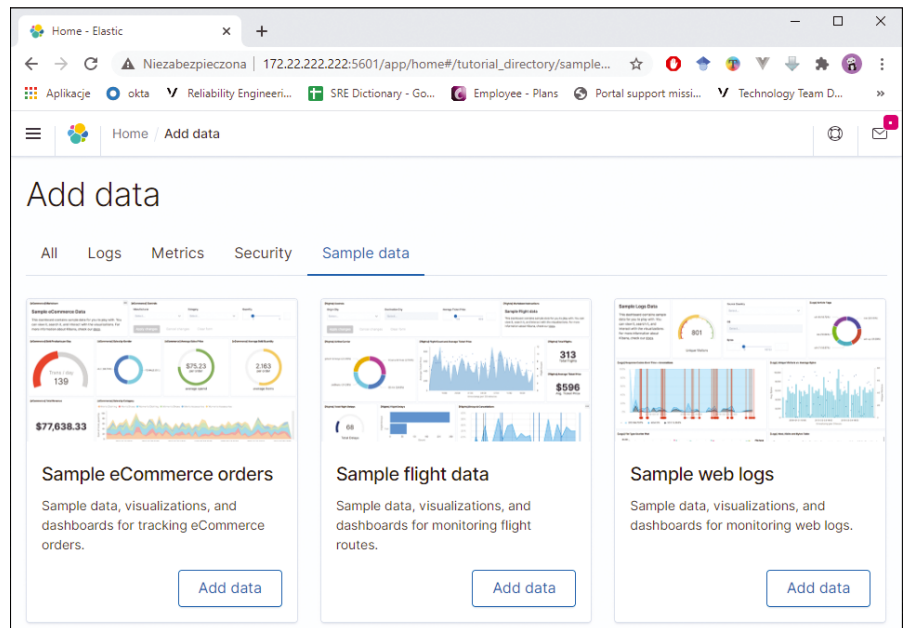


Figure 3: Kibana dashboard in a browser window.

format and immerses it in another (JSON, in this case). To speed up the process, you can use a built in pattern, like `IPORHOST` or `DATA`. There are already hundreds of grok patterns available, but you can define your own, like the `GREEDYMULTILINE` pattern in Listing 13.

A pattern in grok has the format `%{SYNTAX:SEMANTIC}`, where `SYNTAX` is a regex (or another `SYNTAX` with regex) and `SEMANTIC` is a human-acceptable name that you will want to bind to the matched expression. When transformation is completed, you can output the data, in this case to Elasticsearch.

## Kibana and Logtrail

Kibana is a visualization dashboard system that helps you view and analyze data obtained through other ELK components (Figure 3 and 4). Kibana supports hundreds of dashboards, allowing you to visualize different kinds of data in many useful ways. A system of plugins makes it easy to configure Kibana to display different kinds of data.

In this example, I'll show you how to set up Kibana to use Logtrail, a popular plugin for searching and visualizing logfiles.

You have to call `kibana-plugin` to install Logtrail (Listing 14). (Of course,

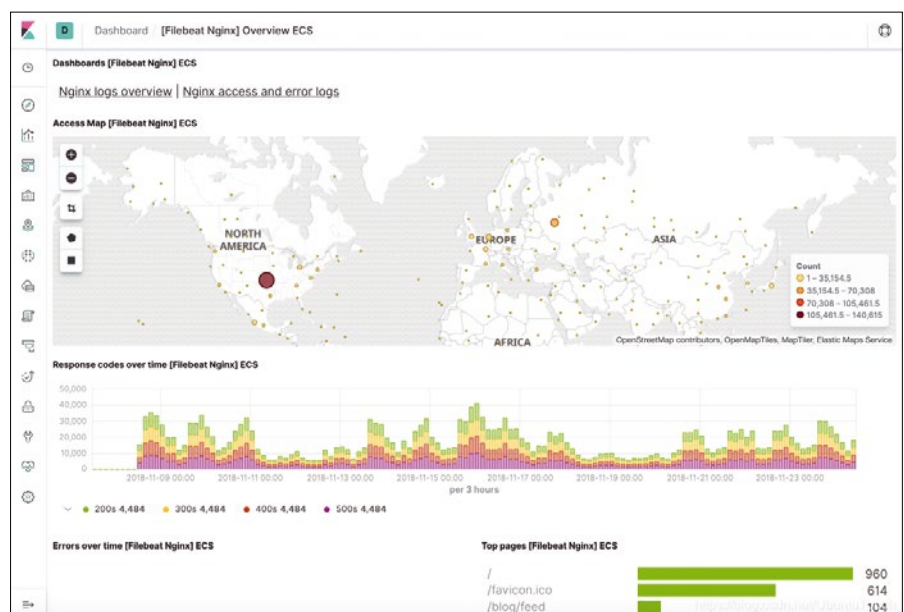


Figure 4: A Kibana dashboard visualizes Nginx log data obtained from Filebeat.

you also could have installed it with Ansible.)

The Logtrail config file (Listing 15) lets you define which index/indices it should take data from, as well as some display settings and mappings for essential fields such as the timestamp, hostname, and message. You can also add more fields and define custom message formats.

## Security

The last step is to introduce some security to the stack. Until now, if you enabled access to the stack from all networks, it would mean that anyone could mess with the data. The ELK base configuration does not include any kind of access restrictions, but you can add security through plugins. Two options are the paid Elastic X-Pack Security plugin [4] and the OpenDistro [5] security plugin.

It is worth noting that another option would be to use a proxy service like Apache or Nginx to enforce authorization, but for consistency, I'll stick with a dedicated solution.

The basic scenario is, a user presents credentials that are verified against ac-

### Listing 14: Installing Logtrail

```
01 [root@ELK ~]# cd /usr/share/kibana/bin/
02 [root@ELK bin]# ./kibana-plugin --allow-root install \
03 https://github.com/sivasamyk/logtrail/releases/download/v0.1.31/
   logtrail-7.8.0-0.1.31.zip
```

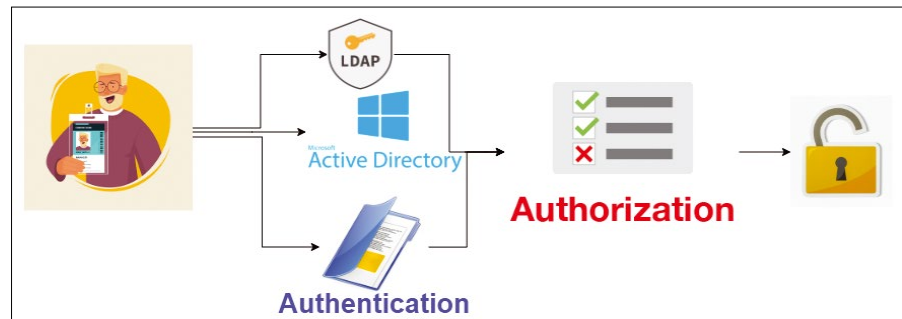


Figure 5: Authentication and access with the OpenDistro security plugin.

cess backends. When the user's identity is confirmed, the security plugin assigns privileges and roles for the user (Figure 5).

When the OpenDistro plugin is enabled, Kibana presents a login panel (Figure 6).

The configuration for the OpenDistro plugin is stored in a few YAML files in `/usr/share/elasticsearch/plugins/opendistro_security/securityconfig/`.

As you can see in Listing 16, the YAML file for the security plugin is organized by user account. The hash is an encrypted password generated with the `hash.sh` script, which is located in the `tools` subdirectory of the plugin directory. The `opendistro_security_roles` entry lets you specify any of the predefined roles. Most of the roles are self-explanatory, but a word is needed for the `logstash` role, since it also includes permissions to write Beats indices. If you want to create your own roles, you have to modify the `action_groups.yml`, `roles.yml`, and `roles_mapping.yml` file, which are located in the plugin's

`securityconfig` subdirectory. The config file can also refer to roles assigned in an authentication system such as LDAP or ActiveDirectory.

You can mark a user, role, role mapping, or action group as reserved. Resources that have the reserved flag set to true can't be changed using the REST API or Kibana. Reserved resources are not returned by the REST API and are not visible in Kibana.

In order to further harden your ELK stack, you can generate certificates to use with SSL and enable them in Elasticsearch, then add user credentials to the Kibana server as well as all beats. In the long run, however, it is a good idea to plug your stack into a company authentication service, such as Okta or LDAP.

## Summary

ELK is an amazing solution that allows users to swiftly explore the sta-

### Listing 15: logtrail.json

```
01 {
02   "index_patterns" : [
03     {
04       "es": {
05         "default_index": "logstash-*"
06       },
07       "tail_interval_in_seconds": 10,
08       "display_timestamp_format": "MMM DD HH:mm:ss",
09       "fields" : {
10         "mapping" : {
11           "timestamp" : "@timestamp",
12           "hostname" : "agent.hostname",
13           "message": "message"
14         }
15       },
16       "color_mapping" : {
17         "field": "log.file.path",
18         "mapping" : {
19           "/var/log/nginx/access.log": "#00ff00",
20           "/var/log/messages": "#0000ff",
21           "/var/log/secure": "#ff0000"
22         }
23       }
24     }
25 ]
26 }
```

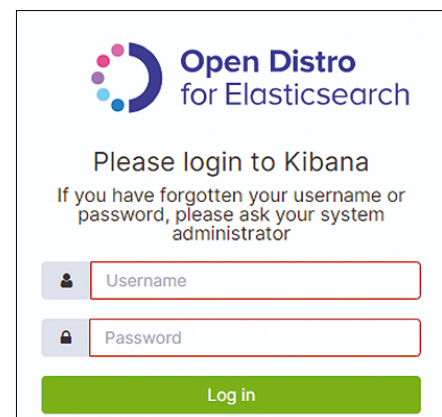


Figure 6: Kibana login panel with OpenDistro.



tus of the infrastructure. Although it was originally designed to handle logging, with later iterations and plugins, it has become a fully functional MAL tool (Monitoring-Alerting-Logging). This paper has touched on a few of the many potential options. Other notable features include fully configurable alerting, machine learning, anomaly detectors, and a performance analyzer. ■■■

### Author

#### Tomasz Szandala

is a PhD student at Wrocław University of Science and Technology and a Site Reliability Engineer at Vonage in Wrocław, Poland. When he isn't studying or working his day job, he spends his time learning and improving Ansible, Jenkins, and other open source tools. Because a man cannot live by learning alone, he sometimes enjoys games like World of Warcraft and Civilization.



### Info

- [1] ELK Stack: <https://www.elastic.co/elastic-stack>
- [2] Ansible: <https://www.ansible.com/>
- [3] grok Filter: <https://github.com/logstash-plugins/logstash-patterns-core/blob/master/patterns/grok-patterns>
- [4] X-Pack Security Plugin: <https://www.elastic.co/guide/en/elasticsearch/reference/current/setup-xpack.html>
- [5] OpenDistro: <https://opendistro.github.io/for-elasticsearch-docs/>
- [6] Code in this Article: <https://github.com/szandala/ELK>

### Listing 16: internal\_users.yml

```
01 # All passwords are:
02 # qwerty
03 _meta:
04   type: "internalusers"
05   config_version: 2
06
07 admin:
08   hash: "$2y$12$N5/i8SBuGv9c8vI5fYNWFe2otKwYPbAfBpN0bFjCDpRJQp0k55bfc"
09   reserved: true
10   hidden: true
11   opendistro_security_roles:
12     - all_access
13   description: "Demo admin user"
14
15 kibanaserver:
16   hash: "$2y$12$N5/i8SBuGv9c8vI5fYNWFe2otKwYPbAfBpN0bFjCDpRJQp0k55bfc"
17   reserved: true
18   hidden: false
19   opendistro_security_roles:
20     - kibana_server
21   description: "Demo kibanaserver user"
22
23 kibana:
24   hash: "$2y$12$N5/i8SBuGv9c8vI5fYNWFe2otKwYPbAfBpN0bFjCDpRJQp0k55bfc"
25   reserved: false
26   opendistro_security_roles:
27     - kibana_user
28     - readall_and_monitor
29   description: "Demo kibana user"
30
31 logstash:
32   hash: "$2y$12$N5/i8SBuGv9c8vI5fYNWFe2otKwYPbAfBpN0bFjCDpRJQp0k55bfc"
33   reserved: true
34   hidden: false
35   opendistro_security_roles:
36     - logstash
37   description: "Demo Logstash & Beats user"
```



## Installing Xubuntu on a Dell notebook

# Upgrade

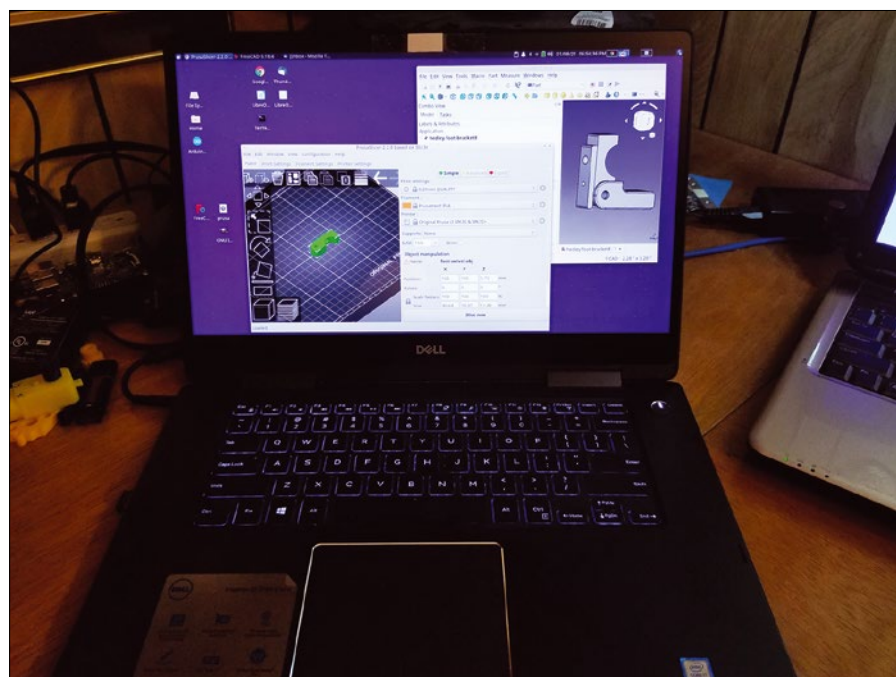
Combining a Dell notebook with Xubuntu and an M.2 SSD makes for a reasonably priced, high-performance system for all your 3D printing projects. *By Rob "drtorq" Reilly*

**A**fter a decade of daily hammering on my old workhorse ASUS notebook, I decided it was time for an upgrade, especially since I've started using FreeCAD [1] with both resin and filament printers for my Steampunk gadget projects [2].

My daughter had a Dell 7573 [3] notebook with Windows 10 (running AutoCAD and Revit) that she no longer needed, but wanted to save the data. So we worked a deal: I swapped in a new M.2 SSD (and my daughter kept the old drive with her data), installed Xubuntu,

and viola, I had an updated notebook (Figure 1).

In this article, I'll walk you through installing Xubuntu [4] on a late-model Dell Inspiron 2-in-1 notebook. In addition, I'll demonstrate how the new notebook works for my Steampunk projects.



**Figure 1:** The shiny new Xubuntu Dell notebook (with the old ASUS notebook in the background).

## Swapping in a Board

I've used Xubuntu on my old ASUS notebook for a long time. Even on ancient hardware, the native Xfce desktop is clean, lightweight, and reliable. When new, the X83VM ASUS was state-of-the-art with a duo-core Intel P8400 CPU at 2.26GHz, 4GB of memory along with 1GB of dedicated video memory for the NVidia GeForce 9600M GS GPU, 802.11 a/b/g/n WiFi, a nice touchpad, a big battery, and an LED backlit 1080x800 display. I upgraded the original 2.5" 320GB SATA disk drive to a screaming 7200RPM, 750GB model a few years ago.

The Dell 7573 sports a quad-core Intel i7-8550U CPU running at 1.8GHz with a maximum frequency of 4.0GHz, 16GB of RAM, and an NVidia GeForce graphics chip. In addition, the Dell comes with a display that shows resolutions up to 3840x2160 on its sleek 15.6" touchscreen, 802.11 a/b/g/n/ac

Lead Image © lightwise, 123RF.com

WiFi, Bluetooth v4.2, and a 250GB solid state disk.

Without thinking, I ordered a run-of-the-mill 500GB SSD “disk” from Amazon to replace the wimpy 250GB offering from Dell. When the disk arrived, I opened the Dell’s case and promptly couldn’t find the existing hard disk. What the? Yup, notebook technology has marched on, and I was still stuck in the late 2010s. The new disk went back to Amazon, and I instead ordered a SK hynix Gold P31 SSD (~\$75) to fit in the PCIe NVMe M.2 slot.

In Figure 2, you can see the spot for the traditional hard drive to the left of the battery. Because the Dell came with an optional 56W battery, there isn’t enough room to mount a traditional 2.5” SATA hard drive. Instead, Dell simply equipped the machine with a Toshiba 250GB M.2 board. You’ll notice that there isn’t even a cable for the hard disk, just a socket. Swapping out the original Toshiba M.2 board for the new SK hynix was easy: Remove a single screw, plug in the new board, and bolt it back down.

In the future, I’ll crack the case first to see what I need before ordering. Alternatively, you could run Xubuntu from the USB stick and verify the hardware with the `lshw` or `inxi -Fxz` commands. Using Xubuntu from a stick is also a great way to make sure everything works on the notebook before a regular installation. While I did that step, I never thought to check for a spinning HDD. Oh, well, you learn something new every day, but I did feel pretty silly.

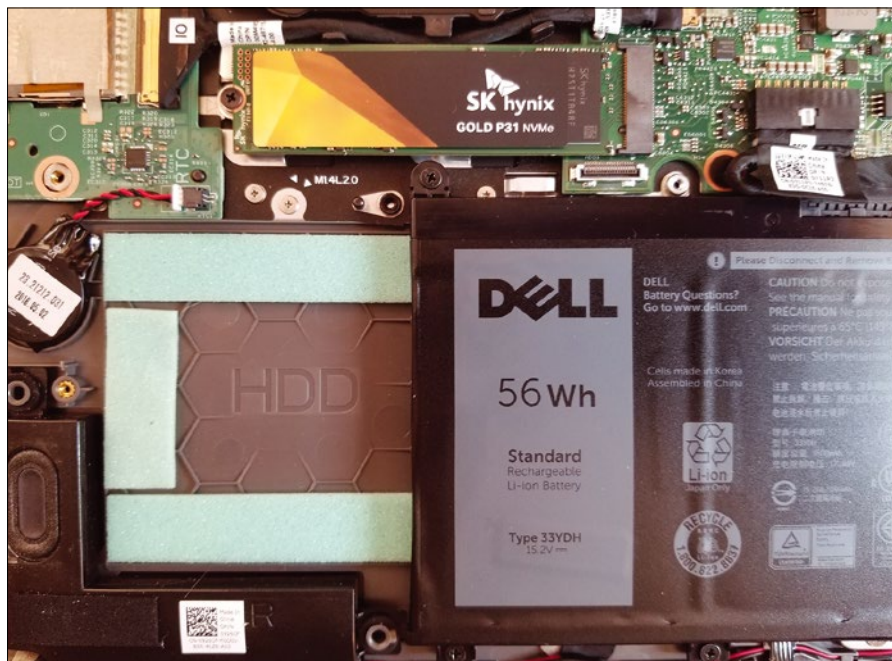
## Installing Xubuntu

The first step to installing Xubuntu is to download an .iso image and burn it to a USB stick (any size over about 2GB will work). I did this on the ASUS using the `dd` command:

```
rob-asus% sudo dd \
if=xubuntu-20.04.1-desktop-amd64.iso \
of=/dev/mmcblk0 bs=32M
```

If you didn’t download version 20.04, use the Xubuntu version you did download. In addition, use the `mmcblk0` designation to choose the whole USB stick and not just a partition; otherwise, it won’t boot.

Once the download is complete, remove the USB stick from the ASUS (or whatever computer you are using for downloading). Next, you’ll need to go



**Figure 2:** The Dell with an empty HDD bay, HDD connector, and the new SK hynix M.2 SSD board.

through the Dell BIOS setup menu to set the Dell to boot from the stick.

Power on the Dell. Don’t plug the stick into the USB port just yet. Press F12 when the Dell logo screen appears to get into the BIOS menu. Under Other Options, select *BIOS Setup* (Figure 3).

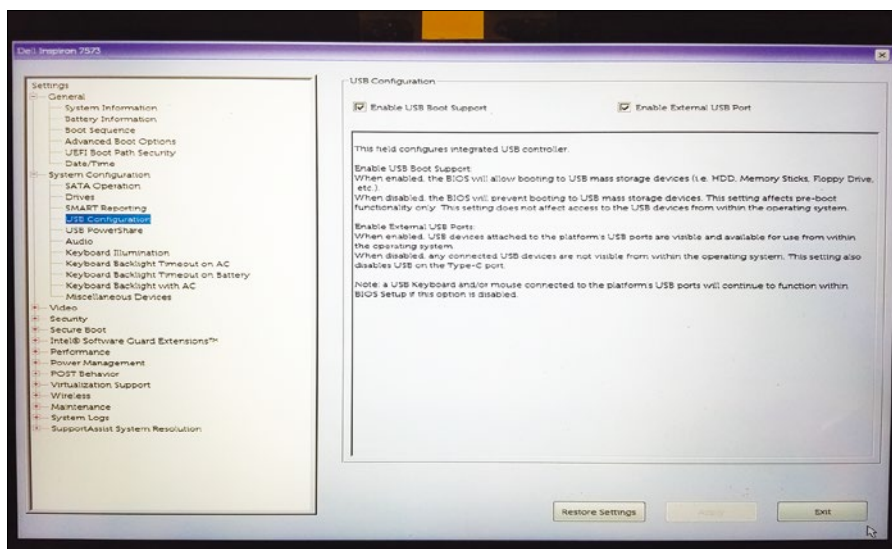
Navigate to *System Configuration*, then select *USB Configuration*, and hit Enter. Make sure *Enable USB Boot Support* and *Enable External USB Port* are selected. These allow booting to a USB-connected device, which makes the USB drives visible after bootup. Save and exit the BIOS Setup menu; power down the Dell.

Now plug the USB stick with your

Xubuntu image into a Dell USB port and power the Dell back up. Again, press F12 to get to the Dell boot menu. This time, select *UEFI USB Boot* and hit Enter.

Booting to the USB stick will take about five minutes, because it has to go through a bunch of filesystem checks. USB memory stick storage is also notoriously slow.

Eventually, the Xfce desktop will appear. With a default resolution of 3480x2160, you may need to squint to click on the Xfce logo button in the upper-left corner of the screen. After selecting *Settings*, choose *Display* and press Enter to change the resolution: A good



**Figure 3:** The Dell BIOS Setup screen.



Table 1: My Desktop Applications

LibreOffice	Word processing, spreadsheet, and presentation application suite
Chrome	Web browsing and YouTube
Thunderbird	Email client
FreeCad	Project physical part development
Prusa Slicer	3D printing slicer program for the Prusa MK3S+ filament printer
Arduino IDE	Arduino and clones microcontroller development environment
Processing	Language for developing quick visually interactive programs
CHITUBOX	3D printing slicer program for the Elegoo Mars resin printer
kdenlive	Video editor
Gimp	Bitmapped graphics editor

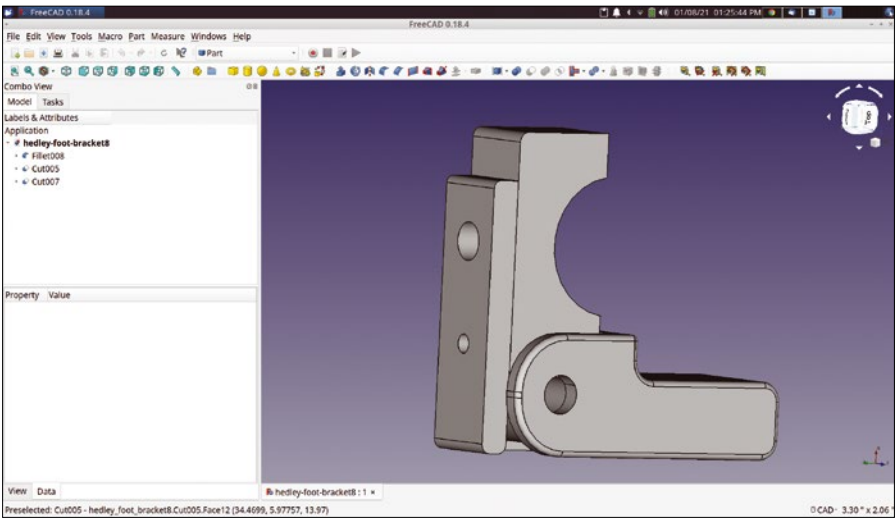


Figure 4: My Steampunk lunchbox foot bracket model shown in FreeCAD.

usable resolution is 1920x1080. Click the Apply button. If all looks good, choose to *Keep* the new resolution and close the pop-up window.

You should now see an installer icon on the desktop. To begin, press the installation icon and then just use the defaults for everything.

Set your WiFi to access your in-house router for software updates. The Dell 7573 doesn't have a traditional wired Ethernet port, so network connections are all done via WiFi.

Desktop Applications

I use a variety of applications for my consulting, writing, and speaking activities (Table 1). The old ASUS was still capable of running all my programs, although compute-heavy operations bogged it down just a bit. On the Dell, I've had no glitches with any of these packages. It's actually quite a tribute to the stability and reliability of the Linux-based ecosystem.

After a new system build, I always immediately install the Synaptic package application manager. Synaptic is easy to

use and lets me add or upgrade programs with a nice user interface. I also use apt at the command line for certain quick or one-off program loading tasks.

Other useful programs related to my Steampunk projects include the Mosquitto MQTT server and client package, the *lucvview* package for viewing video streams from webcams (and the JeVois smart sensor) and the gqr software-defined radio program.

It pretty much goes without saying that LibreOffice, Chrome, and Thunderbird all work great on the Dell. Running all three at once doesn't even trigger the Dell's fan.

FreeCAD ran fine on the ASUS, but it is a real screamer on the much more pow-

erful Dell. FreeCAD is a parametric computer-aided design (CAD) application that runs on Linux, Windows, and macOS. Parametric means that the objects you build are based on properties, such as length, width, and diameter, and can even depend on other objects. As you add features and change properties, those properties propagate through and adjust the part seamlessly. FreeCAD doesn't work very well on a Raspberry PI 4, even with 4GB of RAM. My experience is that the program will load, but rendering and working with models of any size is slow or causes FreeCAD to freeze. Consequently, I suggest sticking with a regular, fairly current notebook, like a Dell, for the best results.

The only real challenge I've had with FreeCAD is finding a workable mouse navigation model. I ended up choosing the CAD model as opposed to Blender or Revit models (which mimic the way you move around in the respective modeling applications). The CAD version uses the Shift key with the right mouse button for rotating the view and the Ctrl key with the right mouse button for panning around the model. Selection is done with the left mouse button, while the mouse roller handles zooming.

FreeCAD on the Dell has turned out to work exceeding well with the Prusa MK3S+ filament printer [5]. It's straightforward to build my models (Figure 4), like the two-piece foot bracket for my Steampunk lunchbox notebook computer (Figure 5).

You simply develop your model and then save the object as a wavefront (.obj)



Figure 5: The foot bracket after 3D printing on the Prusa MK3S+.

file. Next, pull the .obj into the Prusa slicer program; set the print quality, speed, brims and so on; and then hit the *Slice now* button (Figure 6). If everything looks good, you then export the G-code and save the file to a regular SD card. Move the G-code file (on the SD card) to the Prusa printer, select the file on the front panel, and let it rip. The whole process works great and is a pretty quick workflow. At some point, I might try OctoPrint on a Raspberry Pi, which creates a network connection between the Dell and the Prusa printer.

## Cost, Performance, and Usage

I was pleasantly surprised that everything worked right out of the box. Linux newbies are frequently intimidated by the mysterious Linux installation process. I didn't have any issues, other than getting the BIOS set up and installing the newfangled SSD board-styled disk. This was probably the easiest Linux installation ever.

Two years ago, the 7573 went for about \$950 at Best Buy. The Dell's performance with Xubuntu is phenomenal and the thing boots in about 25 seconds. I'm able to not only jump instantly between desktop panels, but do it while

simultaneously running any number of large programs like LibreOffice, FreeCAD, Thunderbird, and Chrome.

Using default power, CPU, and system settings, I get a solid four hours of battery operation on a full charge.

## Only a Few Challenges

There are a few challenges. Getting the Dell notebook to boot from USB might be a little confusing. Remember to use F12 after the Dell logo to get to the BIOS setup/boot menu.

While editing documents in LibreOffice, the large Dell touchpad caused me some grief, with frequent errant characters and mouse jumps. I tried a variety of touchpad settings under *Mouse* and *Touchpad* in the Settings menu, but it just seemed way too sensitive while touch typing. The cursor would move to an odd spot, and I'd have to go back and correct out-of-place typing before proceeding. I simply turned the touchpad off and used an external mouse (I've used the M185 Logitech USB mouse for a long time and it works very well). To use the external mouse, I just disable the touchpad. The mouse, touchpad, and touchscreen can all be turned on and off independently.

Finally, you'll want to change the default ultra-high resolution, which makes the display difficult to read. You can hook up an HDMI big screen monitor, although at that resolution, it's still hard to read. Choosing the 1920x1080 resolution at 120Hz gives a crisp desktop look on the notebook screen.

## Wrap Up

My new notebook setup has been a success. Xubuntu is easy to use and has a good desktop layout. FreeCAD and the Prusa slicer program have certainly streamlined my 3D-printed parts production process.

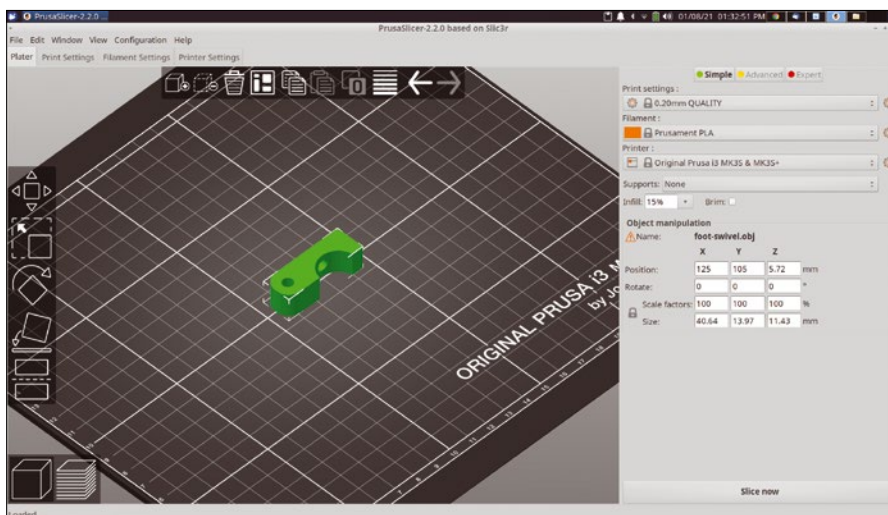
If you are in the market for this type of setup, teaming up Xubuntu with the Dell 7573 notebook and a M.2 SSD board is a winning combination for a reasonably priced, high-performance Linux notebook. ■■■

## Info

- [1] FreeCAD: <https://www.freecadweb.org/>
- [2] Steampunk projects: <https://thenewstack.io/author/rob-reilly/>
- [3] Dell Inspiron 7573: <https://www.bestbuy.com/site/dell-2-in-1-15-6-4k-ultra-hd-touch-screen-laptop-intel-core-i7-16gb-memory-nvidia-geforce-mx130-256gb-ssd-abyss-black/6208307.p?skuld=6208307>
- [4] Xubuntu: <https://xubuntu.org/>
- [5] Prusa 3D printers: <https://www.prusa3d.com/original-prusa-i3-mk3/>

## Author

**Rob "drtorq" Reilly** is an independent consultant, writer, and speaker specializing in Linux/OSS, physical computing, hardware hacking, tech media, and the DIY/Maker movement. He provides a variety of engineering, business, and special project services to individual clients and companies. As a long-time veteran of the tech media, Dr. Torq has posted hundreds of feature-length articles for top-tier tech media and print outlets. He's also presented tech talks at OSCON and other industry venues. Contact him at [doc@drtorq.com](mailto:doc@drtorq.com) or 407-718-3274.



**Figure 6:** The Prusa slicer program showing part of the lunchbox foot bracket, ready for slicing.

## Creating and solving mazes with Go

# Amazed

Mazes fascinated even the ancient Greeks. Mike Schilli uses his Go programming skills to create a maze and then efficiently travel through it. *By Mike Schilli*

In Europe, mazes are said to have come into fashion during the 15th century, mostly on manorial estates where guests were politely invited to “lose themselves” in gardens segregated by high hedges. However, winding paths (some of which even lead you in circles), where you need to make it from a starting point to an end-point, can also be drawn on paper or simulated on computers. Algorithms usually represent a maze’s path system internally as a graph along the edges of which the virtual explorer traverses from one node to the next, discarding dead ends, breaking out of endless loops, and eventually arriving at the destination node.

Computational handling of mazes is divided into two tasks: creating mazes and traversing them as effectively as possible by using a solver. You wouldn’t think it possible, but on Amazon you can actually find a book entitled *Mazes for Programmers* [1]. The Kindle version

## 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.



of this book is a total failure due to font problems, but the paper edition shows some useful methods for creating and solving mazes.

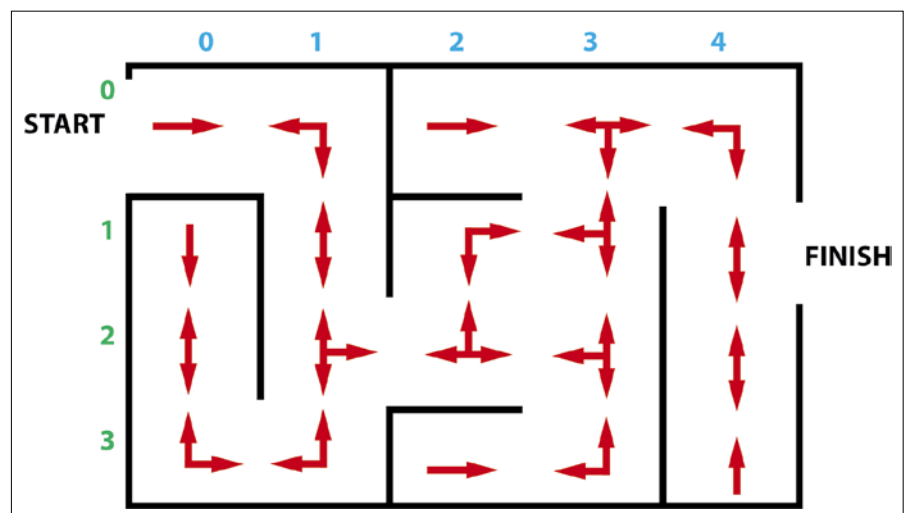
## From Cell to Cell

A maze consists of a matrix of  $M \times N$  cells. In Figure 1, red arrows show the directions in which a traveler advances from cell to cell. These options then define the walls that are drawn to separate a cell from its neighbors.

For example, the starting cell at coordinates  $(0, 0)$  defines that there’s only one path to exit the cell in an eastward direction. Based on this restriction, the algorithm for drawing the maze concludes that the cell’s east wall is miss-

ing and only draws in the south wall as a thick line. The program later defines the allowed direction changes as *North*, *South*, *East*, and *West*. Note that travelers are allowed to walk the maze in either direction, so if a cell specifies *East*, the cell to its right has to allow at least *West*.

The algorithm’s internal representation of the maze as a graph is shown in Figure 2. Only one path leads away from the starting node  $(0, 0)$ , and that is to the node  $(0, 1)$ , going east from the starting node. The first element of the coordinate is the y value that determines the row, the second is the x value for the column – this is how programs usually store two-dimensional arrays.

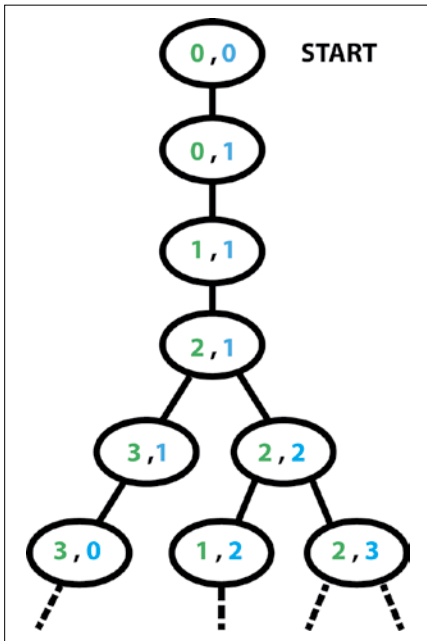


**Figure 1:** A computer-generated 5x4 maze including the permitted moves between cells.

Photo by Henri Lajarrige Lombard on Unsplash



The main program shown in Listing 1 [2] creates a new maze model with `newMaze()` in line 37. This is followed by the definition of the start and finish coordinates, with the cells top left and bottom right. `makeMaze()` in line 41 fires up the random generator and builds the walls. The call to `Solve()` in line 43 finds a path from the start to the finish. The



**Figure 2:** A maze as a graph: the cells as nodes and permitted paths as edges.

result is an array slice of cells to be traversed along this path. The `for` loop starting in line 44 marks them in the internal representation that `Print()` later prints onto the terminal in line 47.

#### Listing 1: maze.go

```
01 package main
02
03 import (
04     "fmt"
05     "math/rand"
06     "time"
07 )
08
09 const (
10     North = 1 << iota
11     South
12     West
13     East
14 )
15
16 type Cell struct {
17     X, Y      int
18     Options   int
19     Visited   bool
20     Solution   bool
21 }
22
23 type Maze struct {
24     Cells [][]Cell
25     Width int
26     Height int
27     Start *Cell
28     Finish *Cell
29 }
30
31 var Turns = []int{North, South, West, East}
32
33 func main() {
34     width := 5
35     height := 4
36     rand.Seed(time.Now().UnixNano())
37     maze := newMaze(width, height)
38     maze.Start = &maze.Cells[0][0]
39     maze.Finish = &maze.Cells[height-1][width-1]
40
41     makeMaze(maze)
42
43     solution := maze.Solve()
44     for _, step := range solution {
45         step.Solution = true
46     }
47     fmt.Print(maze.toString())
48 }
```

The remaining functions, summarized in Listing 2, help both the generator and later the solver to work their way through the cells. For example, each cell has a Boolean `Visited` field that algo-

#### Listing 2: manip.go

```
001 package main
002
003 import (
004     "errors"
005     "math/rand"
006 )
007
008 func newMaze(width int, height int) *Maze {
009     var cells [][]Cell
010     for y := 0; y < height; y++ {
011         row := make([]Cell, width)
012         for x := 0; x < width; x++ {
013             row[x].X = x
014             row[x].Y = y
015         }
016         cells = append(cells, row)
017     }
018     return &Maze{
019         Width: width,
020         Height: height,
021         Cells: cells
022     }
023
024 func (maze *Maze) Reset() {
025     for x := 0; x < maze.Width; x++ {
026         for y := 0; y < maze.Height; y++ {
027             maze.Cells[y][x].Visited = false
028         }
029     }
030 }
031
032 func (maze Maze) Neighbors(cell *Cell) []*Cell {
033     neighbors := []*Cell{}
034     for _, direction := range Turns {
035         tcell, err := maze.Move(cell, direction)
036         if err != nil || tcell.Visited {
037             continue
038         }
039         neighbors = append(neighbors, tcell)
040     }
041     return neighbors
042 }
043
044 func (maze Maze) Move(cell *Cell, turn int) (*Cell, error) {
```

rithms set to true once they have processed a cell. The `Reset()` function, starting in line 24, resets all the fields in the matrix to their original state after the work is done.

The `Neighbors()` function starting in line 32 returns all direct neighbors of a cell in all four cardinal directions as an array slice of pointers to `Cell` types. `Move()` starting in line 44 starts traversing from a given cell in the specified cardinal direction and returns a pointer to the target cell. If the move ends up outside the matrix universe, an error is returned instead.

Figure 3 shows three successive calls to the compiled binary `maze` generated from the listings. The algorithm generates a different maze each time, and the solver always finds a way from the start to the finish.

## Creator

So how do you go about creating a computer-generated maze? Most of the algorithms for this were developed more than 50 years ago. Wikipedia lists the best-known generators [3] and solvers [4]. The listings presented in this issue implement the iterative algorithm.

Listing 1 defines a maze of the `Maze` type with a two-dimensional array slice of cells of the `Cell` type in line 23 for this purpose. In addition to the dimensions of the cell matrix, the structure also stores the starting point and endpoint of the solution path we are looking for.

The maze's cells, in turn, in line with their definition in line 16, consist of the *x* and *y* coordinates of the cell, the path options, and markers stating whether the algorithm has already traversed them

and whether they form part of the path to the solution.

Beginning at the starting point, the program works its way to neighboring cells, randomly changing direction as it goes. If it reaches a dead end where it cannot find any more cells that have not been visited previously as neighbors, it backtracks and starts searching for new candidates that are attached to cells that it has already traversed. In Listing 2, `makeMaze()` pushes all the cells that still need to be processed onto a stack, from where it later picks them up to traverse the cell neighbors and search for new candidates.

Line 83 takes a random neighbor of the current cell to be examined, calls `cellConnect()` to tear down the separating wall, and then allows connections

### Listing 2: manip.go (continued)

```

045  var dx, dy int
046
047  switch turn {
048  case North:
049      dy = -1
050  case South:
051      dy = 1
052  case West:
053      dx = -1
054  case East:
055      dx = 1
056  }
057
058  if cell.X+dx > maze.Width-1 ||
059     cell.X+dx < 0 ||
060     cell.Y+dy > maze.Height-1 ||
061     cell.Y+dy < 0 {
062     return cell, errors.New("Stepped outside")
063  }
064
065  return &maze.Cells[cell.Y+dy][cell.X+dx], nil
066 }
067
068 func makeMaze(maze *Maze) {
069     stack := []*Cell{}
070
071     maze.Start.Options = East
072
073     stack = append(stack, maze.Start)
074     for len(stack) > 0 {
075         lastIdx := len(stack) - 1
076         cur := stack[lastIdx]
077         stack = stack[:lastIdx]
078
079         ncells := maze.Neighbors(cur)
080
081         if len(ncells) > 0 {
082             stack = append(stack, cur)
083             idx := rand.Intn(len(ncells))
084             rcell := ncells[idx]
085
086             rcell.Visited = true
087             cellConnect(cur, rcell)
088             stack = append(stack, rcell)
089         }
090     }
091 }
092
093 func cellConnect(c1 *Cell, c2 *Cell) {
094     if c1.X == c2.X {
095         if c1.Y < c2.Y {
096             c1.Options |= South
097             c2.Options |= North
098         } else {
099             c1.Options |= North
100             c2.Options |= South
101         }
102     }
103
104     if c1.Y == c2.Y {
105         if c1.X < c2.X {
106             c1.Options |= East
107             c2.Options |= West
108         } else {
109             c1.Options |= West
110             c2.Options |= East
111         }
112     }
113 }

```

in the respective cardinal direction thanks to the `Options` cell variable. In doing so, both cells go through changes that reflect the direction. For example, to set up a connection between two cells stacked on top of each other, the lower cell is given the option North as a viable path, while the upper cell receives the South option.

## Complicated Pop

Removing the last element from an array slice in Go and storing it in a variable (as shown in line 75-77 of Listing 2) takes a surprising number of steps. Scripting languages like Python do this in a snap with `pop()`.

However, Go insists on first addressing the last index of the array (`len(slice)-1`) and fetching the value there. Then it is a matter of reassigning the partial slice

from index numbers 0 to `len(slice)-1` (exclusively, that is, without the last element) to the same slice variable (Listing 3). Thanks to Go's internal slice arithmetic, which only moves pointers but does not recopy elements, the whole thing is very efficient – but not that easy to read in the program code.

## Efficient ORing

To efficiently store a set of options in an integer variable, Listing 1 defines the cardinal directions as constants. The keyword `iota` with the `<<` operator in line 10 sets the values of the constants North, South, West, and East as powers of two (i.e., to the values 1, 2, 4, and 8).

This allows a variable (such as `Options`) to be set to more than one of these values by binary ORing. For example, if a cell is to allow transitions to both north and east, the program sets its value to `1 | 8` (i.e., 9). If the program then wants to test whether east is an option, it checks `Option & 8`, which is always non-zero if an 8 was previously ORed with it.

## Randomness

When randomly selecting elements from an array slice, like in line 83 of Listing 2, it is important to take great care to en-

### Listing 3: Array Slice Pop in Go

```
last = slice[len(slice)-1]
slice = slice[:len(slice)-1]
```

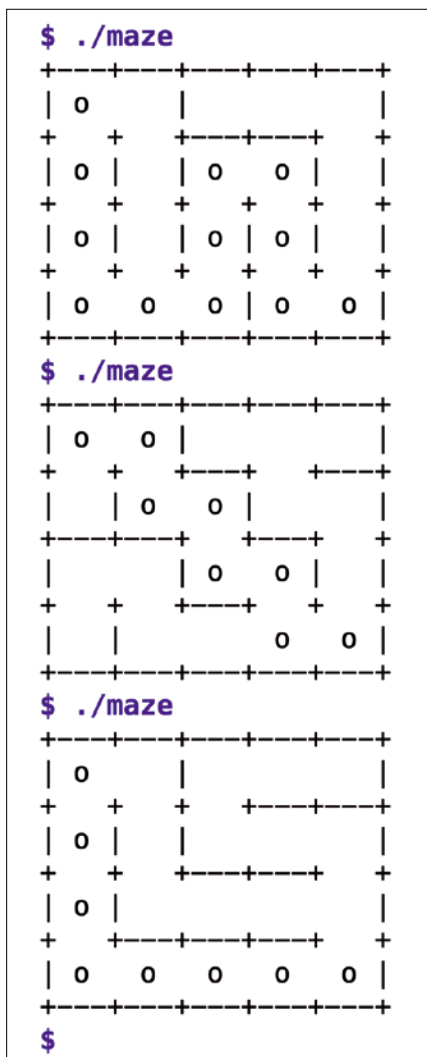
### Listing 4: Random Selection

```
idx := rand.Intn(len(slice))
randElement := slice[idx]
```

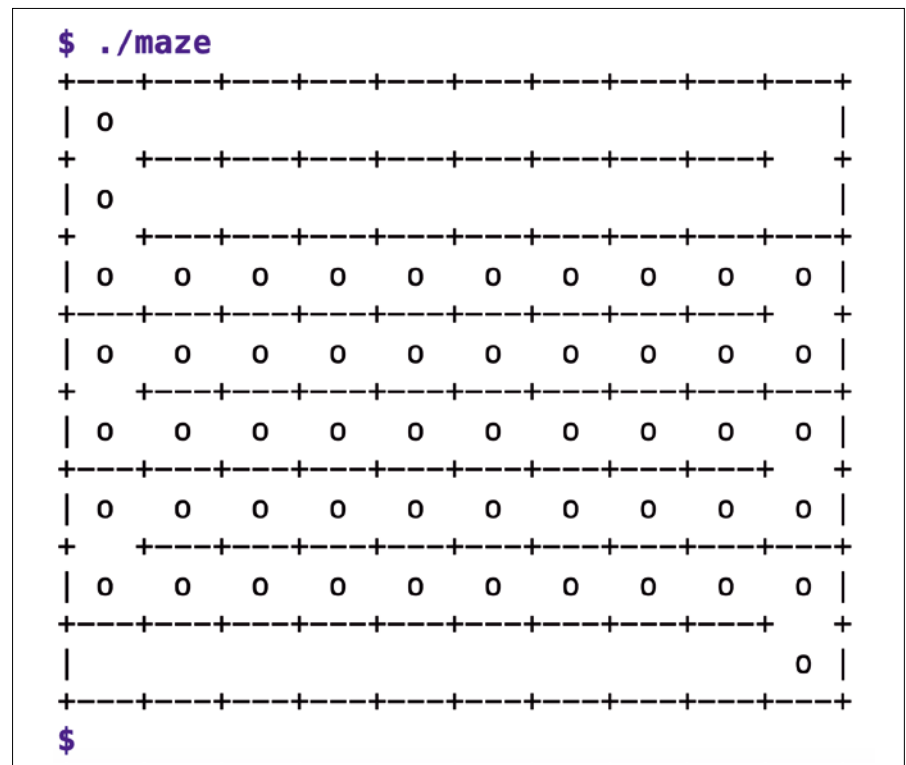
sure that the pick is truly random, rather than some elements appearing preferentially. The `Intn(x)` function from Go's standard `math/rand` package returns equally distributed random values in the form of integers between 0 (inclusively) and the integer value `x` (exclusively) for this purpose.

In the code snippet shown in Listing 4, `len(slice)` returns the number of elements in the array; however, the index of the last element is one less than this. The random generator returns equally distributed index values between 0 and `len(slice)-1`, inclusively – because the random value lands between 0 (inclusively) and `len(ncells)` (exclusively) – which is exactly what the doctor ordered to ensure selection of a random array element.

What happens if the random generator does not output totally random values,



**Figure 3:** Three different 5x4 mazes with paths from the start, located top left, to the finish, at bottom right.



**Figure 4:** A broken random number generator with a bias to only generate boring mazes.



but has a bias, as shown in Figure 4?

The result is a boring maze that no human would ever tackle – but the computer does a good job anyway, of course.

To prevent the generator from generating the same maze every time, when the program is run again, `main()` needs to initialize it to a value that varies from call to call. Line 36 in Listing 1 therefore uses `rand.Seed()` to set the value to the current time in nanoseconds, which ensures distribution over a relatively wide range.

## Journey Without a Destination?

The algorithm for generating the maze only receives the starting point and does

not seem to need an endpoint; this is due to the fact that the procedure generates paths along which the explorer can reach all points of the maze. The procedure ensures that no compartmentalized sections are created. This guarantees that the solver later can reach any randomly defined endpoint.

The generator also avoids infinite loops, because it never turns onto paths that it has already taken. If you do want mazes with loops, you will find a wide range of different algorithms that are also easy enough to implement.

## Painting Lessons

To display the maze on screen, the program can either use a graphics library or,

in the simplest case, write characters into the terminal as ASCII art. Since individual cells in the maze share their walls with neighboring cells, you don't need to bother with displaying all four walls:

Two walls are enough if a cell takes care of its south and east walls, and the missing bits are provided by the neighbors to the north and west.

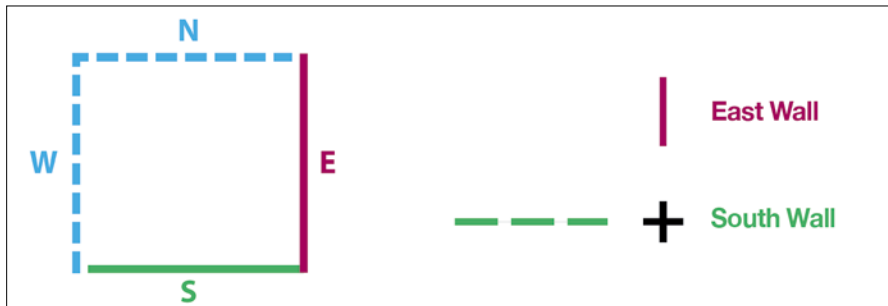
The algorithm for drawing the maze needs to iterate through the cells of the internal representation from left to right and from top to bottom, as shown in Listing 5. It draws a wall at the bottom of each cell unless the cell is open in the direction of South and draws a wall on the right if East is not an option. If a cell is at the edge of the maze (i.e., at the far

**Listing 5:** `print.go`

```
01 package main
02
03 func (maze Maze) toString() string {
04     output := ""
05
06     for x := 0; x < maze.Width; x++ {
07         output += "+---"
08     }
09     output += "\n"
10
11     for y := 0; y < maze.Height; y++ {
12         upper := "|"
13         lower := "+"
14         for x := 0; x < maze.Width; x++ {
15             wall := "|"
16             if (maze.Cells[y][x].Options & East) != 0 {
17                 wall = " "
18             }
19             sol := " "
20             if maze.Cells[y][x].Solution {
21                 sol = "o"
22             }
23             upper += " " + sol + " " + wall
24
25             wall = "---"
26             if (maze.Cells[y][x].Options & South) != 0 {
27                 wall = "  "
28             }
29             lower += wall + "+"
30         }
31         output += upper + "\n"
32         output += lower + "\n"
33     }
34
35     return output
36 }
```

**Listing 6:** `solve.go`

```
01 package main
02
03 func (maze Maze) Solve() []*Cell {
04     maze.Reset()
05     stack := []*Cell{maze.Start}
06
07     for len(stack) > 0 {
08         cur := stack[len(stack)-1]
09         cur.Visited = true
10
11         if cur.X == maze.Finish.X &&
12            cur.Y == maze.Finish.Y {
13             return stack
14         }
15
16         var moveto *Cell
17         for _, turn := range Turns {
18             if (cur.Options & turn) != 0 {
19                 trycell, err := maze.Move(cur, turn)
20                 if err == nil && !trycell.Visited {
21                     moveto = trycell
22                     break
23                 }
24             }
25         }
26
27         if moveto == nil {
28             // nowhere to go, backtrack
29             stack = stack[:len(stack)-1]
30             continue
31         }
32
33         stack = append(stack, moveto)
34     }
35
36     return stack
37 }
```



**Figure 5:** On the left, a maze cell with closed south and east walls; on the right, a closeup of the ASCII characters for each wall.

left, right, top, or bottom), the algorithm also puts walls in those directions.

Figure 5 shows a maze cell with closed east and south walls (shown on the left). Each cell consists of five columns and three rows, but an individual cell is only responsible for its south wall (four columns) and east wall (two rows).

For a closed south wall, the bottom row contains the string "----+"; if it is open, it contains " +". For a closed east wall, the second line from the top contains the pipe character |; for an open east wall, it contains a space instead. As mentioned previously, the cell does not need to worry about the north and west walls; the south and east walls of the neighboring cells take care of that.

## Finding Solutions

Once the maze is defined, following the call to `makeMaze()` in line 41 of Listing 1, the `Solve()` function in Listing 6 looks for a solution to traverse the maze from start to finish.

### Listing 7: solve.go

```
$ go build maze.go print.go solve.go manip.go
```

The solver does a depth-first search (i.e., it first digs down in order to find a solution with a quick path). The alternative would be to try out all possibilities at every turn in order to reach the goal by the shortest path (a breadth-first search).

If some paths in the maze take the explorer in circles, this can pose a challenge to more primitive solvers; they have to avoid getting trapped into running around in circles forever. Our trusty solver, `Solve()`, on the other hand, is well equipped to avoid this trap; it tags the `Visited` field of previously processed cells and would never think of looking at any of them again.

This means that the solver dashes forward from the starting point along allowed paths, takes the first available branch in each case, and pushes the corresponding cell onto the stack. If it happens to pass the finish point, it cancels the chase and claims victory, using the cells stacked on the stack as the solution path. This is not necessarily the shortest path to the solution, but at least the algorithm finds it quickly.

If, on the other hand, it does not arrive at the end of the maze, but at a dead end, it has to retreat along its own stack. It lopes back to a turnoff taken previously and explores an alternative path from there. In this way, it eventually runs into the end-point and returns the stack built up to that point to the caller as the solution path.

The main program uses the solution cell array to tag the cells along the path, in each case setting the `Solution` flag to true. The print function writes an o to these cells.

## All Together Now

Listings 1, 2, 5, and 6 are compiled by the command shown in Listing 7 to create a `maze` binary; this in turn outputs a 5x4 maze, complete with a solution, when called. The dimensions can easily be changed by manipulating the values for `width` and `height` in `main()`. Interested hobbyists are also encouraged to try out different algorithms for generating and solving new exciting mazes [5]. ■■■

### Info

- [1] Buck, Jamis. *Mazes for Programmers: Code Your Own Little Twisty Passages*, Pragmatic Bookshelf, 2015: <https://www.amazon.com/dp/B013HA1UY4>
- [2] Listings for this article: <ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/245/>
- [3] Maze generation algorithm: [https://en.wikipedia.org/wiki/Maze\\_generation\\_algorithm](https://en.wikipedia.org/wiki/Maze_generation_algorithm)
- [4] Maze solving algorithms: [https://en.wikipedia.org/wiki/Maze\\_solving\\_algorithm](https://en.wikipedia.org/wiki/Maze_solving_algorithm)
- [5] Go GitHub project for creating and solving mazes: <https://github.com/itchyny/maze>

An ergonomic laptop keyboard

# Custom Comfort

A first for laptops, Keyboardio Atreus offers an ergonomic, portable keyboard with customizable key programming. *By Bruce Byfield*

Generally speaking, laptops are not designed for typists. On many laptops, the keys are smaller than on a full-sized keyboard. Almost always, laptops use chiclet keys (small, flat squares) that slow typing, take a toll on fingers and hands after a few hours, and can cause serious repetitive stress injuries (RSI) over time. Ergonomic keyboards for laptops simply do not exist. Or rather they did not until recently, when Keyboardio teamed up with keyboard designer Phil Hagelberg of Technomancy to produce the Keyboardio Atreus (Figure 1) [1]. Although not integrated into a laptop frame, this portable USB keyboard is the length of a banana, making it easy to carry. It fulfills the long-ignored need for an ergonomic laptop keyboard – although you might want to change the factory layout to suit your needs.

Keyboardio first gained attention in 2017 with its Model 01 [2], a split keyboard with ergonomic, programmable mechanical keys and backlights mounted on two slabs of maple. Able to go head-to-head with most other keyboards available for sale, the Model 01 outshines all its rivals in terms of sheer

beauty. Sadly, the Model 01 is currently out of production, with a more advanced successor not due until the end of 2021, although slightly used ones are still available at just under the original price of \$329.

Meanwhile, Keyboardio is selling its modification of Technomancy's Atreus design for \$149. Although you might assume that a keyboard is a keyboard, the Atreus and the Model 01 could not be more different in design. True, both are programmable and use high-end mechanical keys. However, in contrast to

the maple-mounted Model 01, the Atreus is starkly functional, black plastic, with key mechanisms plainly visible and no backlights. More importantly, keys are mounted on a single platform, instead of on two. In addition, the Atreus's keys are all cut from the same design instead of being individually sculpted like the Model 01's keys. To a certain extent, these differences do seem to lessen the Atreus's ergonomics: In my experience, while I can type for 10 hours on the Model 01, the repetition begins to affect my wrists after about seven hours on the



**Figure 1:** The Keyboardio Atreus is a portable, ergonomic keyboard – one of the first of its kind.

Lead Image © damedeeso, 123RF.com



Atreus. However, that is still almost twice the time I can type unaffected on a standard keyboard.

In its favor, the Atreus's keys are banked so that many users should be able to reach half the keys simply by stretching out one hand – even though the keys are full-sized. In fact, many users can probably reach two-thirds or more of the keys. Equally importantly, like the Model 01, the Atreus' keys are banked on a diagonal instead of being staggered like the keys on a traditional keyboard. Together, these arrangements are enough to reduce repetitive stress by reducing finger movement. Of course, any ergonomic features are far better than the typical laptop's total lack of such features.

## Key Programming

The Atreus reduces finger movement by having an extremely minimalist layout. Where most keyboards have 101-104 keys, the Atreus has only 44 keys compared to the Model 01's 66. If you examine the Atreus, you will notice that a keypad and arrow, number, and function keys are missing, as well as other navigation keys like Page Up or Home. Instead, the top three rows of keys contain the basic alphabet and standard punctuation marks. Along the bottom of the keyboard, you'll find the Enter, Shift, Space, Esc, Tab, Ctrl, and Alt keys, plus the Fun key and Super key.

The Fun and Super keys give you access to additional key arrangements or layers, in the same way that the Shift keys give you access to uppercase letters on any keyboard. For example, pressing the Fun key followed by the *U*, *I*, and *O* keys gives you 7, 8, and 9 respectively, while *S*, *D*, *F*, and *C* act as arrow keys. Pressing the Fun key plus *Z* and *X* will produce square brackets. Similarly, press the Super key followed by *U*, *I*, and *O* to get the function keys F7 through F9. For easy reference, you'll find these layers displayed on a laminated layout card (Figure 2).

These layers help to reduce the physical size of the Atreus, although they do take some getting used to. The names “Fun” and “Super” do not automatically suggest the function of the keys – in what way are numbers fun, for example? Moreover, they are not as conveniently organized as the Model 01's layers,

which can be mostly ignored unless you want a number pad or function key. Another problem is that the keys' arrangement seems illogical. For instance, why is the *I* key linked with *M*, at the bottom of the keyboard, or *7* with *U* at the top?

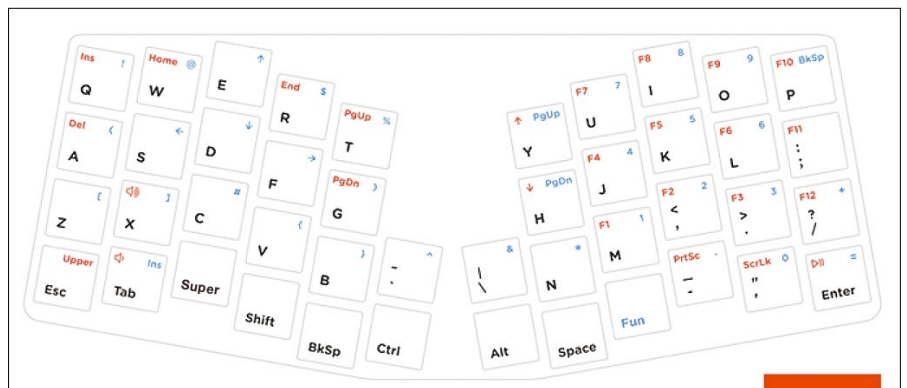
Fortunately, you can program the layers and keys to suit yourself. For instance, if you want, you can switch the key assignments so that the number and function keys are placed across the top rows of keys – an arrangement that is close enough to that of a full-sized keyboard that you are unlikely to forget them. You can also choose to develop macros and assign them to a key. Not all keys on all three default layers are assigned, so unless you create a dozen or more macros, you are unlikely to have to make decisions about which key assignments to keep.

You can reassign keys in two ways. First, you can install Chrysalis (Figure 3) [3], the

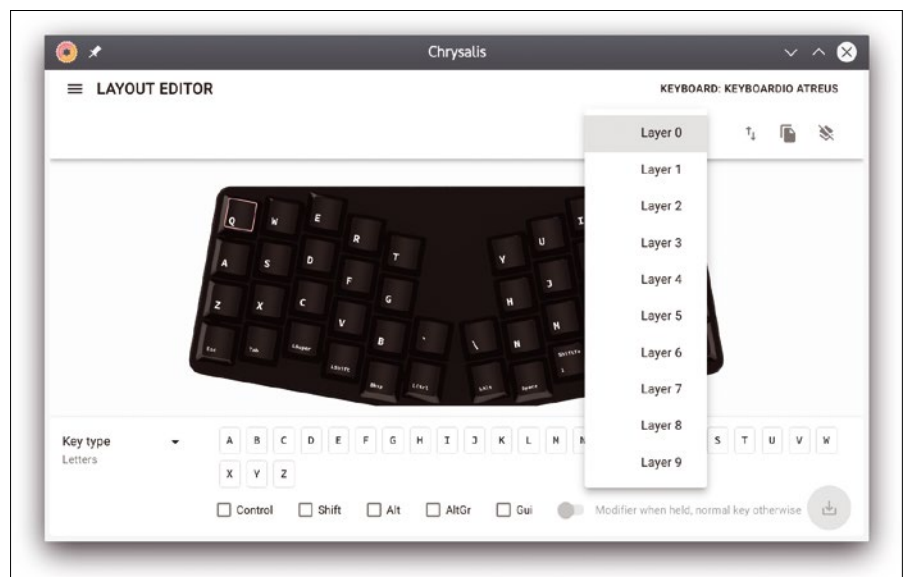
in-development graphic key layout application. In practice, Chrysalis seems occasionally flaky, but mostly it is serviceable.

Alternatively, you can work with Keyboardio's Kaleidoscope code [4] to flash the firmware directly, downloading the Atreus Sketch [5] – the firmware for the keyboard's ATmega32U4 MCU microcontroller. In the Arduino IDE, you can edit the text-based layout in the Atreus Sketch, using a standard key code [6]. Listing 1 shows the default layout.

In the same way, you can edit the other two default layers and even add another six layers, including macros for your favorite games or an alternative to the default QWERTY layout. Then, you can follow the instructions to flash the revised firmware while pressing the Esc key. On the whole, editing the Atreus Sketch is the more reliable way to change key assignments and not much harder than Chrysalis to learn.



**Figure 2:** The Atreus comes with three default layers, with room for another six.



**Figure 3:** To program your Atreus, you can download the Chrysalis graphical application.

### Listing 1: The Default Layout

```
[QWERTY] = KEYMAP_STACKED
(
    Key_Q    ,Key_W    ,Key_E    ,Key_R    ,Key_T
    ,Key_A    ,Key_S    ,Key_D    ,Key_F    ,Key_G
    ,Key_Z    ,Key_X    ,Key_C    ,Key_V    ,Key_B, Key_Backtick
    ,Key_Esc ,Key_Tab ,Key_LeftGui ,Key_LeftShift ,Key_Backspace ,Key_LeftControl

    ,Key_Y    ,Key_U    ,Key_I    ,Key_O    ,Key_P
    ,Key_H    ,Key_J    ,Key_K    ,Key_L    ,Key_Semicolon
    ,Key_Backslash,Key_N    ,Key_M    ,Key_Comma ,Key_Period ,Key_Slash
    ,Key_LeftAlt ,Key_Space ,MO(FUN) ,Key_Minus ,Key_Quote ,Key_Enter
```

### Other Features

Three other products are available for the Atreus. If you have added many changes to the layout, you may prefer to order a set of blank keycaps, rather than deal with keys whose etched labels are no longer accurate [7]. You probably will want to order a travel case, although the keyboard and a USB cable are a tight fit (Figure 4) [8]. If you miss the beauty of the Model 01's maple mounts for keys, you may also want to order the walnut palm rest (Figure 5) [8]. The keyboard

fits into the palm rest [9], adding a touch of beauty when you are not on the road. The palm rest does not fit into the travel case, although it might still fit into your laptop case.

### Last Words

A laptop with a built-in ergonomic keyboard still does not exist. Until one does, the Atreus is an acceptable substitute, if sometimes rough around the edges here and there. Be warned, though, that it may take a few tries to configure the keys to your liking, and as long as a week to get used to the layout. After that, you may find other keyboards awkward and lacking. All the same, if you

use a laptop and prefer to customize your keyboard layout and type with fewer injuries, the Atreus is likely to be a must-have piece of hardware. ■■■

### Info

- [1] Keyboardio Atreus: <https://shop.keyboard.io/products/keyboardio-atreus>
- [2] Keyboardio Model 01: <https://shop.keyboard.io/products/model-01-keyboard?variant=30996744405065>
- [3] Chrysalis: <https://github.com/keyboardio/Chrysalis/releases>
- [4] Kaleidoscope: <https://github.com/keyboardio/Kaleidoscope>
- [5] Atreus Sketch: <https://github.com/keyboardio/Kaleidoscope/blob/master/examples/Devices/Keyboardio/Atreus/Atreus.ino>
- [6] Firmware key codes: <https://kaleidoscope.readthedocs.io/en/latest/customization/keycodes.html>
- [7] Blank keycaps: <https://shop.keyboard.io/products/blank-keycaps-for-the-atreus>
- [8] Atreus travel case: <https://shop.keyboard.io/collections/keyboardio-atreus/products/keyboardio-atreus-travel-case>
- [9] Atreus palm rest: <https://shop.keyboard.io/products/atreus-palmrest>



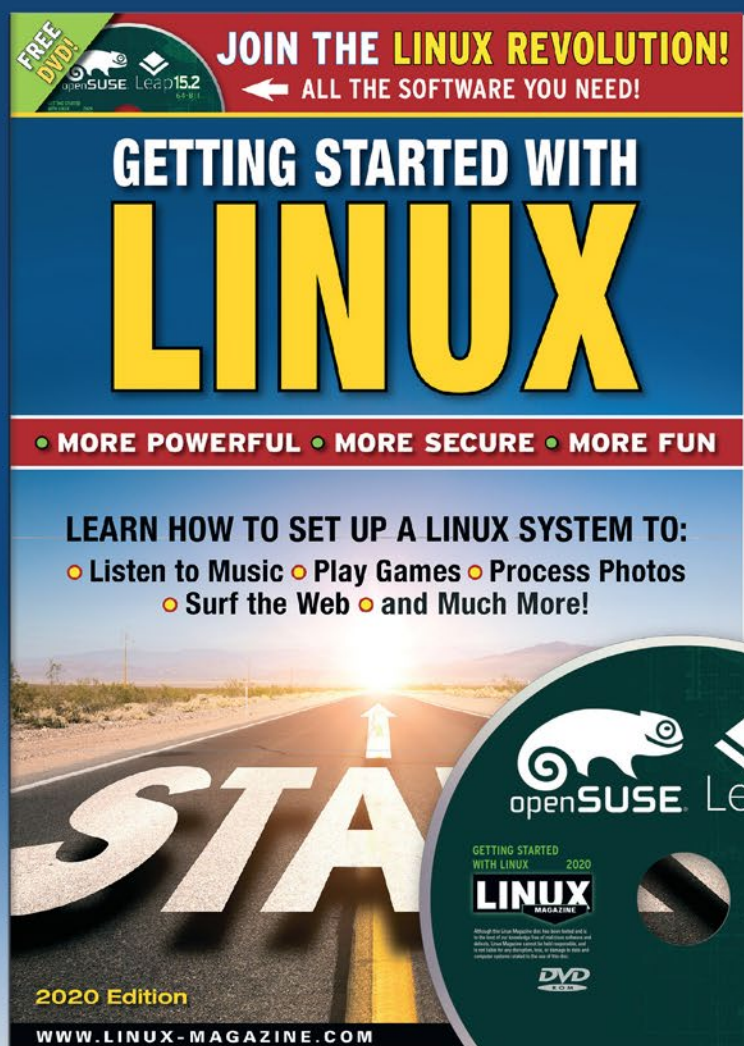
**Figure 4:** The Atreus's travel case is an optional extra.



**Figure 5:** The optional walnut palm rest adds elegance for home use.



# Hit the ground running with Linux



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

This single issue shows beginners how to:

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

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





# MakerSpace

## Build a minimal Raspberry Pi OS from source

# Raspberry Pi DIY

The detLFS project provides an ideal foundation for compiling Linux from source code, either to experience the fundamentals of how Linux works or to prepare an operating system for a project with very specific requirements. *By Bernhard Bablok*

**T**he Linux From Scratch (LFS) project has been around for more than 20 years [1]. It does not deliver software as a result, but detailed instructions on how to create an executable Linux system from the source code itself. Why might you want to take this route? (1) You might be a masochist with too much time on your hands. (2) You have a project with very specific requirements. (3) You want to learn fundamentally how Linux works. The detLFS [2] project by Thomas Dettbarn answers these needs.

Unlike the pure theory of LFS, detLFS makes life far easier for anyone wanting to take on the challenge because the project provides a set of scripts instead of a manual. The scripts create a runnable, minimal Linux system for the Raspberry Pi almost without interaction. As a prerequisite you need a desktop Linux with the usual build tools (i.e., a compiler and `make`), as well as ImageMagick/Netpbm for your own kernel logo. These tools can be found on any normal developer's computer; if in doubt, install them from your package manager. They are typically pseudo-packages named *build-essentials*, or something similar.

Simply downloading the repository and running the scripts is boring and ba-

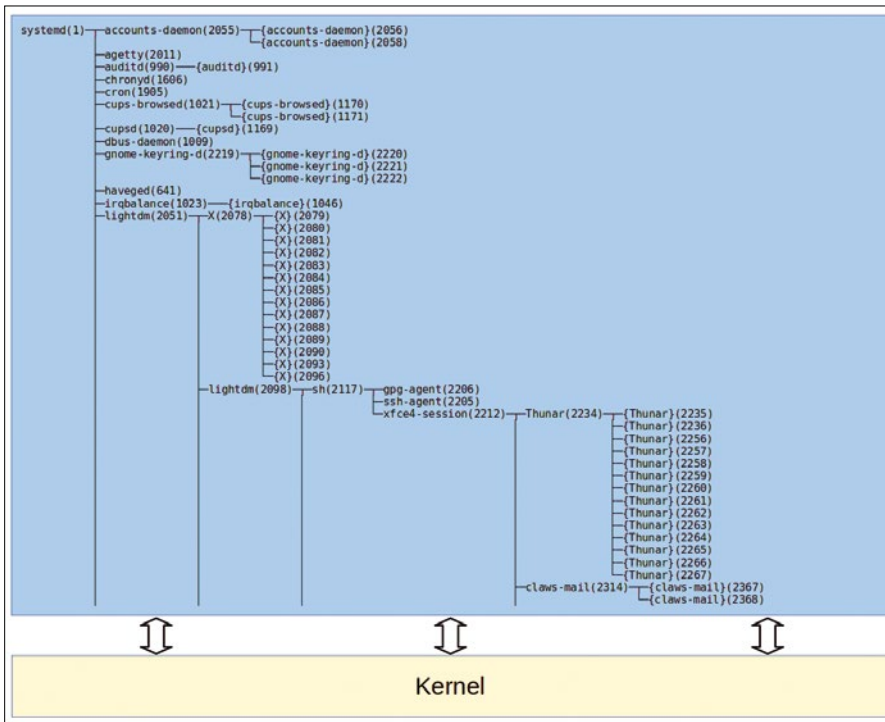
sically pointless, because nobody gains anything from the minimal system the process creates. Understanding the process and knowing how to adapt it to your own needs is far more important.

### How Linux Works

Whether on a Raspberry Pi or on a supercomputer, Linux basically always works in the same way. First, the hardware boots the kernel from disk. This part depends heavily on the platform. Without firmware, also known as BIOS in the desktop area, nothing works.

After startup, the kernel initializes its subsystems and drivers then passes control to the init process by starting the `/sbin/init` program with process ID 1. On modern Linux systems, the `systemd` process adopts this ID shortly afterward. The init program then gradually starts all the application programs and processes that make up a system. The kernel itself remains in the background as a silent worker, stepping in whenever a program needs access to resources, such as during a program startup or for reading a file (Figure 1).

A minimal system comprises three components: the firmware, a kernel, and at least one init program. For a Raspberry Pi, whose only task is to control and



**Figure 1:** The launched user processes interact with the kernel.

switch something, this setup might be enough. detLFS is not quite as minimal. In addition to the init program, standard Linux programs or honed down versions (e.g., ls, cp, etc.) are also present.

To create your own minimal system with very little overhead, a script first collects all the required program sources. Subsequently, a second script generates a cross compiler along with companion programs, which are needed because you want to create an executable kernel and application programs for the Raspberry Pi ARM architecture on an x86 system. The schematic flow is shown in Figure 2. All scripts need to run within a single session; otherwise, you will have

to repeat the export commands, which are valid only within a single session.

### Improvements

For this project, I provide a fork of the original scripts on GitHub [3], mainly because various bugfixes and optimizations significantly speed up the download and the process of building the system. Furthermore, the fork supports all hardware variants of the Raspberry Pi, not just versions 2 and 3. You can retrieve the software with the commands:

```
$ git clone https://github.com/
bablokb/pi-detLFS
$ rm -fr /data/projects/detLFS
```

```
$ cp -a pi-detLFS/detLFS
/data/projects
$ cd /data/projects/detLFS
```

The last two commands copy the directory with the scripts to a suitable project directory and change to that location. If the directory is on a filesystem formatted with XFS or Btrfs, all of the later scripts run faster and they require less space because these filesystems only copy the references when duplicating files, not the data blocks.

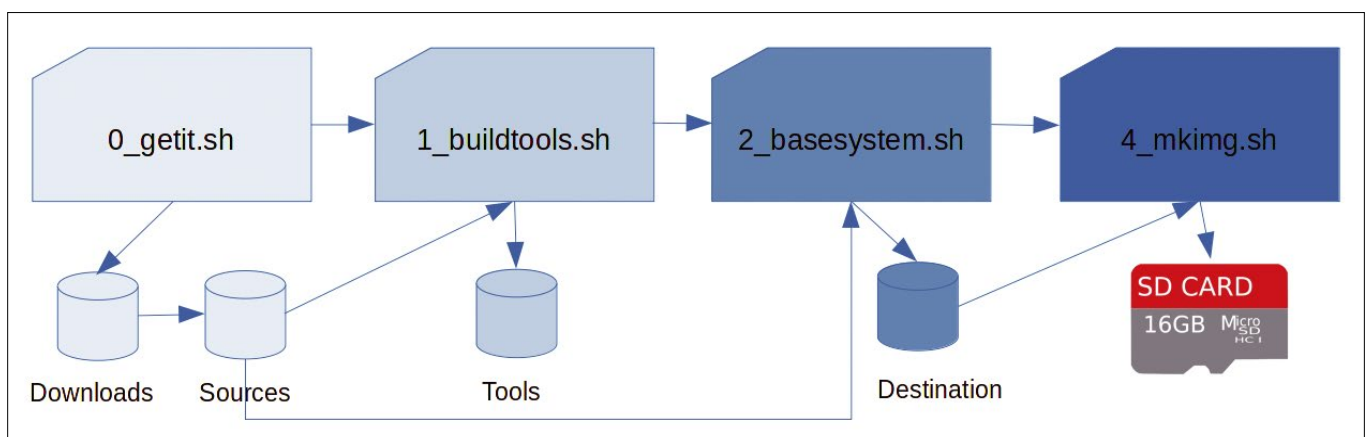
After these preparations, run:

```
$ export BRANCH=rpi-4.19.y
$ ./0_getit.sh |& tee 0_getit.log
```

The optional first line specifies the kernel branch the script fetches from the repository (various lines of development exist in separate branches). Without the export command, the script uses the current default branch. Instead of setting the branch manually, you can alternatively copy the kernel configuration file .config from a running Raspberry Pi into the detLFS directory:

```
$ sudo modprobe configs
$ zcat /proc/config.gz > .config
```

The 0\_getit.sh script in the earlier set of commands downloads about 1.1GB of compressed sources, which when unpacked add up to about twice that. For each download, the script creates a file in the Downloads/ directory that follows the pattern .detlfs.<xxx> (e.g., .detlfs.kernel) (Figure 3). If you want to download something again, maybe because you want to try out a different kernel



**Figure 2:** The schematic flow for creating a system generated with detLFS.

Name	Size	Type	Modified date
linux	526 Bytes	Folder	2020-08-15 14:50:02
boot	380 Bytes	Folder	2020-08-15 14:50:05
busybox-1.31.1.tar.bz2	2.3 MiB	Tar archive (bzip-compressed)	2019-10-25 10:42:40
binutils-2.32.tar.xz	19.8 MiB	Tar archive (XZ-compressed)	2020-08-15 14:45:42
.detlfs.binutils	0 Bytes	Adobe Bridge Workspace File	2020-08-15 14:45:44
glibc-2.29.tar.xz	15.8 MiB	Tar archive (XZ-compressed)	2020-08-15 14:45:57
.detlfs.glibc	0 Bytes	Adobe Bridge Workspace File	2020-08-15 14:45:59
gmp-6.1.2.tar.xz	1.9 MiB	Tar archive (XZ-compressed)	2020-08-15 14:46:02
mpfr-4.0.2.tar.xz	1.4 MiB	Tar archive (XZ-compressed)	2020-08-15 14:46:05
mpc-1.1.0.tar.gz	684.8 kiB	Tar archive (gzip-compressed)	2020-08-15 14:46:08
gcc-8.3.0.tar.gz	109.0 MiB	Tar archive (gzip-compressed)	2020-08-15 14:47:24
.detlfs.gcc	0 Bytes	Adobe Bridge Workspace File	2020-08-15 14:47:29
.detlfs.kernel	0 Bytes	Adobe Bridge Workspace File	2020-08-15 14:50:02
.detlfs.bootloader	0 Bytes	Adobe Bridge Workspace File	2020-08-15 14:50:20
.detlfs.busybox	0 Bytes	Adobe Bridge Workspace File	2020-08-15 14:50:27
.detlfs.make	0 Bytes	Adobe Bridge Workspace File	2020-08-15 14:50:30
make-4.2.1.tar.gz	1.9 MiB	Tar archive (gzip-compressed)	2020-08-15 14:50:30

**Figure 3:** The Downloads/ directory is used by the detLFS scripts as a cache for all required files.

version, delete the file in question beforehand. In any case, check the `0_getit.log` file after execution – this applies in kind to all further steps.

Only the downloaded source packages end up in the Downloads/ directory. With the exception of the kernel, these are archives. The script immediately copies or unpacks the sources into the Sources/ directory. The scripts downstream of this step then make use of them.

## Toolbox

The second step in the workflow creates the tools that the downstream scripts need and relates primarily to the cross compiler, which runs on a desktop system (x86), but generates code for the ARM variant:

```
$ ./1_buildtools.sh | & 2
tee 1_buildtools.log
```

Fortunately, the normal desktop compiler also builds this cross compiler in the most time-consuming step in the whole process. Ready-made cross compilers can be found online and from the Raspberry Pi Foundation, but users report in forums having difficulties with these tools time and time again. Therefore, the detLFS project prefers building the programs itself.

The computer I used in this project, an AMD Ryzen 5 2400G, is not necessarily a top performer; in fact, it is basically geared toward office work with its integrated GPU. The quad-core processor claims to be an eight-core CPU to the operating system because of hyperthreading. If all the cores are used (to find out, use the `-j 8` option with `make`), the call described above takes about 18 minutes. If only one core is doing the work, it takes 62 minutes. However, hyperthreading doesn't offer too many benefits with this CPU because of the fairly small CPU cache: with four cores, the compiler run takes 22 minutes, which is only marginally longer than with the full core count.

Once created, the tools do their job no matter how many different kernel and system versions you want to try, so the runtime is only consumed once.

## Linux Emerges

During the download, you specified the kernel version but not the variant that matches the hardware. Raspberry Pi models belong to three hardware categories. At the lower end are the Raspberry Pi 1, the Pi Zero variants, and Compute Module 1 (CM1). At the upper end are the Raspberry Pi 4 and 400. In between are the Raspberry Pi 2 and 3 and CM3.

The next script creates the kernel and the base system:

```
$ export target=pi2
$ ./2_basesystem.sh |& 2
tee 2_basesystem.log
```

The first line provides the correct mapping. Depending on the variant, the script generates its own kernel image (`kernel.img`, `kernel7.img`, and `kernel8.img`) and corresponding modules. If the variable is missing, the script generates a kernel for the Raspberry Pi 2/3/CM3 hardware group.

If you set the branch with a variable at download time as described above, then the script completes without any further interaction and creates a kernel with the default configuration. However, if you copy a `.config` file to the detLFS directory, the script calls `make oldconfig`. Because the sources are usually more recent than when the kernel configuration and some drivers were added; you need to answer a handful of questions. The simplest (and usually most sensible) variant is to accept the default value by pressing Enter.

For full control of the kernel configuration, set the `MENUCONFIG` variable. The value does not matter, the main thing is that the variable is not empty. In this case, the script calls `make menuconfig`, after which the kernel can be customized to your liking.

People who go to the trouble of making their own kernel often also change the boot logo. To do this, copy an image to the `logo/` folder in the root directory of the project. The name must be `mylogo.<xxx>`; for a common JPEG file, this would be `mylogo.jpg`. If the `convert` tool from the ImageMagick package and the tools from the Netpbm package are on the system, the call in the last script converts the logo to the format expected by the kernel.

## On the Card

After running the script to create the base system, all the files reside in the `Destination/` folder. The original project copies the files directly to an SD card with the `4_mksdcard.sh` script. Alternatively, `4_mkimg.sh` creates an image file that differs from the usual Raspberry Pi images in terms of size and content only. You can then transfer these to an SD



card with the usual tools. Both scripts need root privileges, so you will have to run them with `sudo`.

If you use the original `4_mksdcard.sh` script, you have to enter the correct device name of your SD card reader beforehand into the script with an editor and remove the premature abort built in for safety.

After this step comes the moment of truth. If successful, the system boots with your own logo (Figure 4). On the first and second consoles, which you select with `Alt + F1` and `Alt + F2`, respectively, a login for the root user with password `root` appears.

## First Boot

When you boot, the fast startup process is immediately noticeable: After the kernel hands over control to `/sbin/init`, not much happens. The SysVinit system starts the few processes from `/etc/inittab`; besides the two consoles mentioned earlier, these processes include the `/etc/rc5` script, which just prints a *hello world* message.

At this point, you could have a control program that takes control of the Raspberry Pi's GPIOs, for example. The original project also includes a script that can be used to send a few additional packets to the Pi and turn the system into a WiFi repeater. Essentially, the LFS system provides only an extremely minimal basis on which you can build your own projects.

Thanks to the multifunction binary BusyBox software suite, the system comes up with a number of programs that have all the basic functions of the standard programs but do without all the options to save space. Although it was no problem to configure the Pi's Ethernet interface and open a connection to the Internet, I had to look up the commands and syntax in my cheat sheets



**Figure 4:** If you stored your own logo, it accompanies the boot messages.

from more than 20 years ago. The usual distributions offer the luxury of simple configuration files.

## Conclusions

The focus of the (det)LFS process is to create a small, compact system without complex dependencies for running stably and reliably in critical situations. Of interest to normal users is that once the cross-compile toolchain is up and running, a new kernel can be created within a short time. Moreover, detLFS simplifies the process of porting available software as source code that is not (yet) available in the Debian or Raspberry Pi OS repositories.

Cross compiling is worth the work because a desktop Linux still runs far faster than even the fastest Raspberry Pi 4. However, if you prefer calm and slow, you can work through the detLFS process on a Pi. A manual on the PiLFS project page [4] closely follows the LFS

manual. This project also provides scripts for a simplified semi-automated process. Although it will take hours to run, at least you don't have to watch the small-board computer do its work. ■■■

## Info

- [1] Linux from Scratch: <http://www.linuxfromscratch.org>
- [2] detLFS project: <http://www.dettus.net/detLFS/>
- [3] Author's detLFS fork: <https://github.com/bablokb/pi-detLFS>
- [4] PiLFS: <https://intestinate.com/pilfs/>

## Author

Bernhard Bablok works at Allianz Technology SE as an SAP HR developer. When he's not listening to music, cycling or walking, he deals with topics related to Linux, programming and small computers. He can be reached at [mail@bablokb.de](mailto:mail@bablokb.de).



# MakerSpace

## Home automation with a Raspberry Pi Home Director

Control devices from different manufacturers of home automation devices from a single interface by combining free software and a Raspberry Pi. *By Erik Bärwaldt*

**S**mart homes with many computer-controlled devices are in vogue, with increasing numbers of manufacturers jumping on the bandwagon. Smart light bulbs and individually controllable sockets are no longer the only items available: Air conditioners, washing machines, ovens, and multimedia and alarm systems can also be integrated through smart home apps.

Not all smart homes are created equal: Different specifications (e.g., the Zigbee and Z-Wave standards) render many components from different manufacturers incompatible. Ultimately, you can end up juggling several corresponding matching control devices and apps for the individual end devices.

However, armed with just a Raspberry Pi and ioBroker home automation software [1], you can use components from different manufacturers intelligently, while bundling a wide variety of standards – including legacy WiFi – in a single device. You no longer need several different apps on your smartphone to control each component, and you don't need a high-powered, permanently running, conventional PC to control the devices.

### ioBroker for the Smart Home

ioBroker is written in JavaScript and is considered the bedrock among free software solutions for home automation, having been under continuous development for more than six years and available back

when the Internet of Things (IoT) was still making strangely hesitant steps into private households.

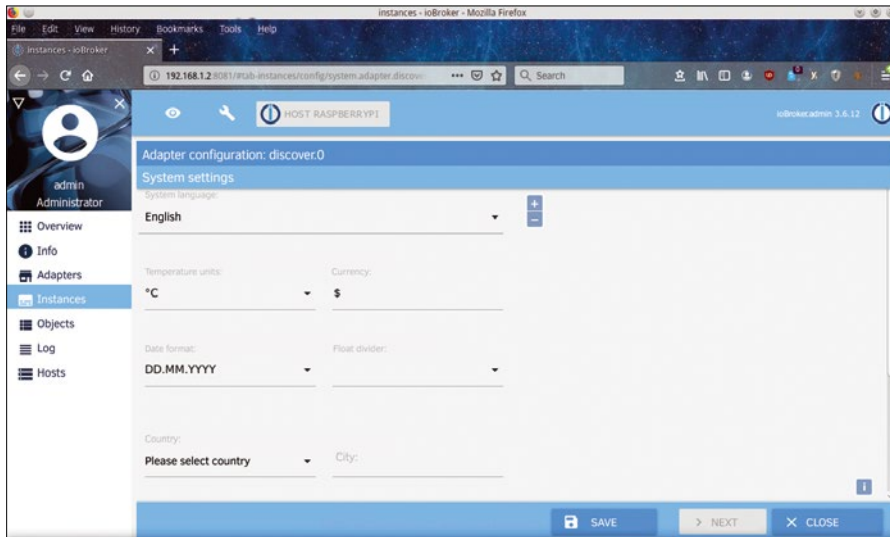
The cross-platform software is modular, with each virtual adapter (now available for more than 300 components and services) communicating with its complementary device. ioBroker supports the integration of new adapters on the fly. Additionally, you can combine several ioBroker servers to create a multihost, an option that is likely to be of particular interest in large environments such as enterprise infrastructures.

Several ZIP archives for the Raspberry Pi cover the spectrum from the Raspberry Pi 2 to the current fourth generation device, although some of the archives are based on older versions of Raspbian. In addition to the images, the developers also provide a repository that supports integrating the software into an existing Raspberry Pi OS. The install is documented on the project's website [2].

The ZIP archives usually contain a minimal installation of Raspbian, without a graphical user interface and a login of username *pi* and password *raspberry*. The US keyboard layout is the default after installation.

### Configuring the Software

ioBroker is managed entirely in a web browser, for which the client-server application integrates a web server. You



**Figure 1:** ioBroker's interface requires no training.

can reach the management interface at `http://<Pi IP>:8081` from a workstation on your home network. A license agreement appears, and you must agree to anonymous statistical data being collected before you can access the dashboard. This setting can be changed later.

The system then guides you to the basic settings menu, where you can modify some system options (Figure 1). By default, only the Admin, Discovery, and Info adapters are active in ioBroker. For a list of installed adapters with some adapter-specific status and management information, click *Instances* in the vertical options bar on the left side of the browser window.

To discover your existing smart home devices – and have the corresponding adapters installed – select the *discovery.0* entry in the *Instances* view (Figure 2), which opens an area with a list of methods for adapter detection (e.g., Ping, UPnP, mDNS). You can enable/disable the individual options by checking or unchecking the boxes.

After clicking the *Discover* button, ioBroker finds responsive devices and services. Please note

that it only detects components and control devices (so-called gateways) on the local and WiFi networks, but not end devices, such as smart switches or lamps with intelligent controls, addressed over other protocols.

The system lists the gateways and components it finds in a new window after the search, where you can enable the desired components by checking the boxes for the individual entries and then pressing the *Create Instances* button at the bottom of the window. The

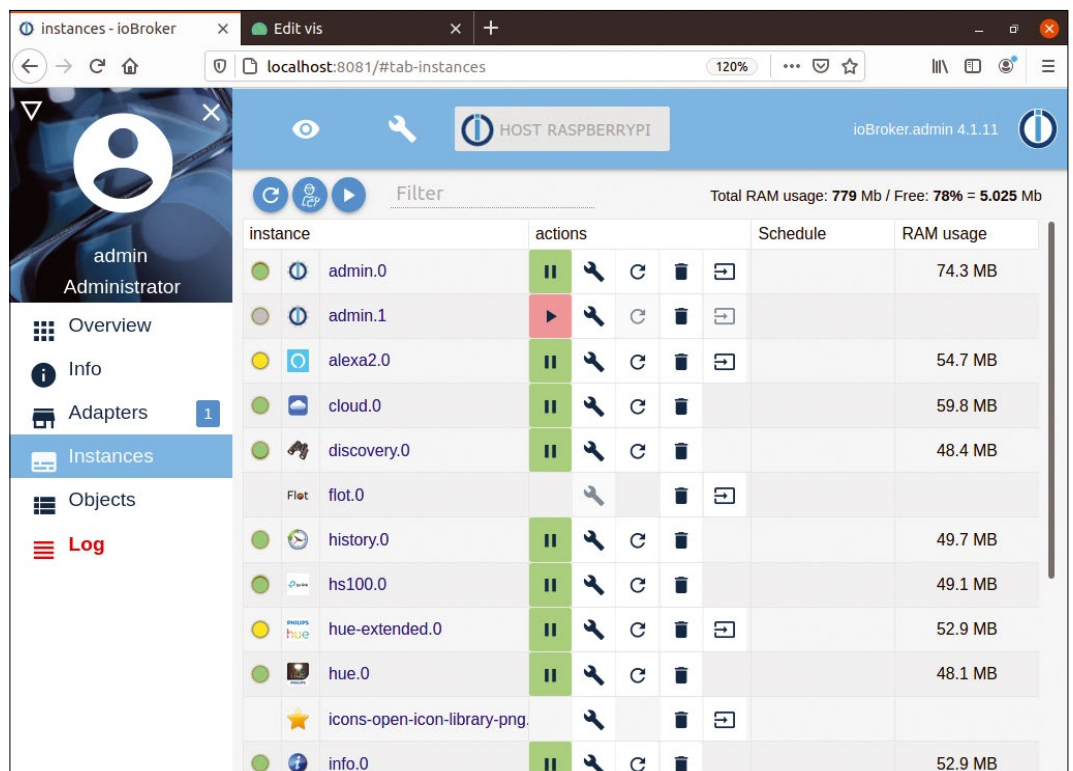
software then downloads the required packages and integrates them into the system.

## Updating Adapters

After the initial installation of the main adapters, it makes sense to look at the *Updates* box in the *Info* group in the sidebar. If updates are available for installed adapters or components, you can install them by clicking on the update (circular arrows) icon in front of the adapter name. ioBroker then updates the software in question.

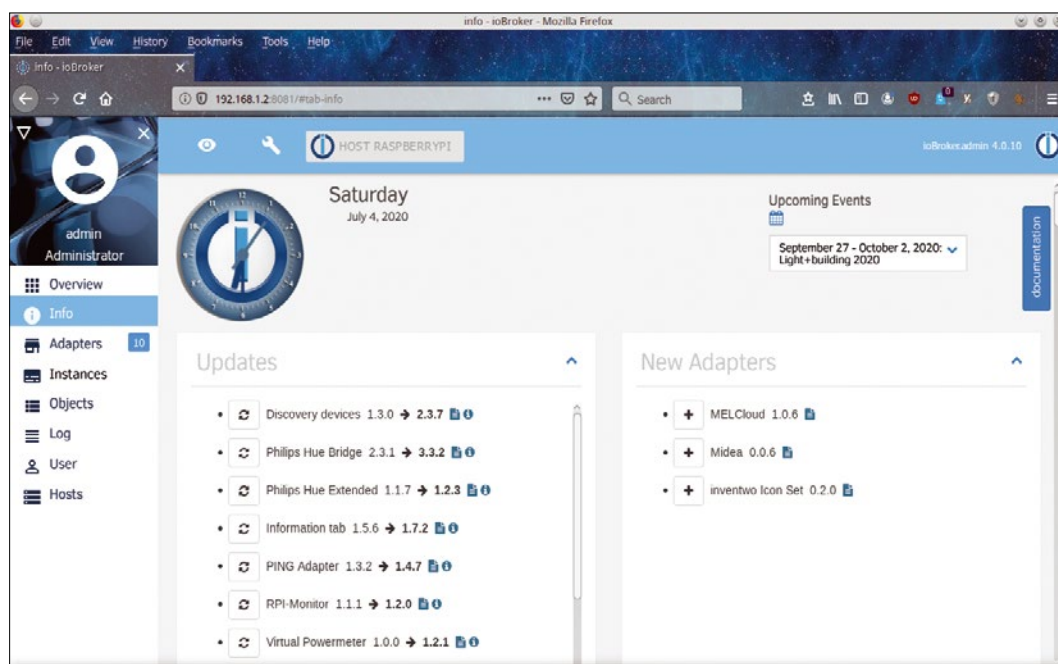
On the right side of the window, the *New Adapters* list shows adapters newly implemented in the software. If you have what were previously unsupported smart home components, it makes sense to check here to see if an adapter is now available for this device (Figure 3).

To insert adapters retroactively, use the *Adapters* entry in the sidebar (Figure 4). ioBroker lists all the available adapters on the right side of the window, with the previously installed adapters at the top of the list. Once you have found the right module for your smart home component, click on the small menu in a blue circle on the right above the name of the adapter to see a short description of the respective adapter. Pressing the



**Figure 2:** The preinstalled Discovery adapter searches for existing smart home devices.





**Figure 3:** ioBroker is actively maintained and developed.

plus button at bottom left in the description block installs the adapter. The module fires up automatically and appears in the list of available instances.

For most adapters, a configuration window opens where you can adapt the module to suit the hardware. Above all, the individual manufacturers' control devices first need to be connected to the Raspberry Pi in the corresponding dialogs after installing the adapter, so you can access the connected devices. To do this, you usually have to specify at least the IP address of the adapter and the port number on which it can be accessed.

You can discover the device's IP address either in your router's settings menu or with tools that show active devices on the local network, like Angry IP Scanner [3] for Linux or Fing [4] for your smartphone. For some adapters, you also need to make settings on the gateway during configuration in ioBroker. If this is the case, a corresponding message box usually appears (Figure 5).

In this case, the settings that appear are context-sensitive; the options displayed are specific to the device in question. After finishing the configuration, close the dialog and switch to the *Objects* category.

## Objects and Custom Scripts

The *Objects* dialog lists all the existing adapters with their respective configuration options. They are summarized in a tree below the respective instance

(Figure 6). You can typically modify the configuration of the end devices in the individual folders by adjusting the predefined values. Because the individual attributes of the devices are also required for creating corresponding controls in the Vis editor visualization dialogs (discussed in the next section), you will want to assign meaningful names to the object attributes and enter them in the corresponding lines of the attribute view to ensure a better overview of the features in the Vis editor interface.

Because the current configuration of a component does not yet allow auto-

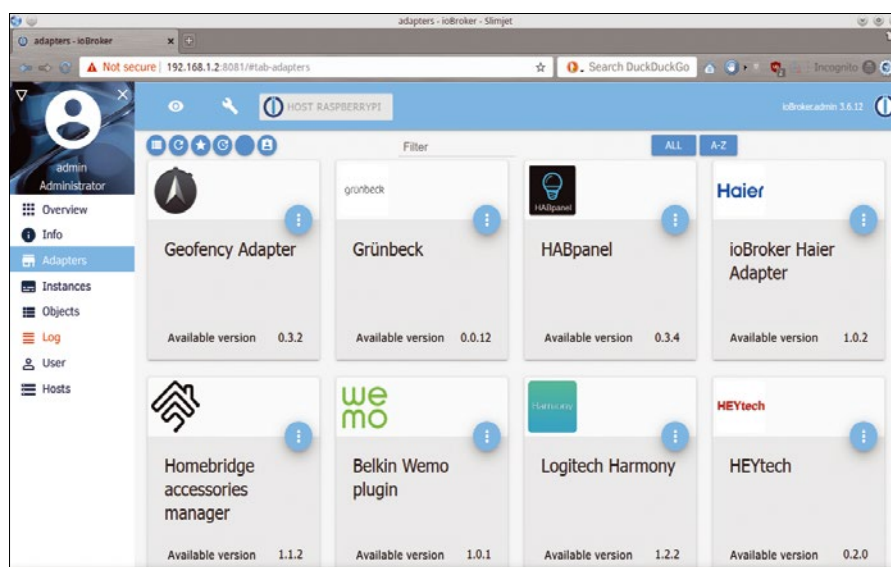
mated control, ioBroker offers a scripting option for installing a convenient interface: the JS Script Engine adapter. In the interface, you can choose whether you want to create the script in JavaScript, Blockly, or TypeScript. If you choose Blockly, ioBroker integrates a modular system that allows programming of components through the underlying JavaScript back end with graphical puzzle pieces.

Once you have integrated the script engine into the system, a new *Scripts* entry appears in

the sidebar, where you can call the engine. Script programming requires some prior knowledge, but the ioBroker developers do offer detailed documentation to help you out [5].

## Uniform Interface

With the help of the Vis interface, you can combine your entire private IoT into a unified control center. Vis is also implemented as an adapter and installed when the control software in ioBroker is installed. The visualization module combines all common control elements such as switches, scripts, and scenes of the smart home infrastructure in a uniform,



**Figure 4:** ioBroker supports more than 300 adapters from all areas.

web-based interface that you can access with any end device (Figure 7). To access the interface, either call the corresponding adapter in the *Instances* group from the adapter web page icon, or enter the IP address of the Raspberry Pi followed by port number 8082 in the address bar of a browser.

If you call the Vis adapter from the Instances view, the Vis editor appears, where you can define the individual attributes for the end devices to be controlled. The editor has complex op-

tional settings for the respective actuators, switches, sensors, and terminal devices; it is a good idea to read the manual up front to avoid creating faulty controls. The individual device attributes can be found by choosing the *Objects* item in the ioBroker sidebar (Figure 6).

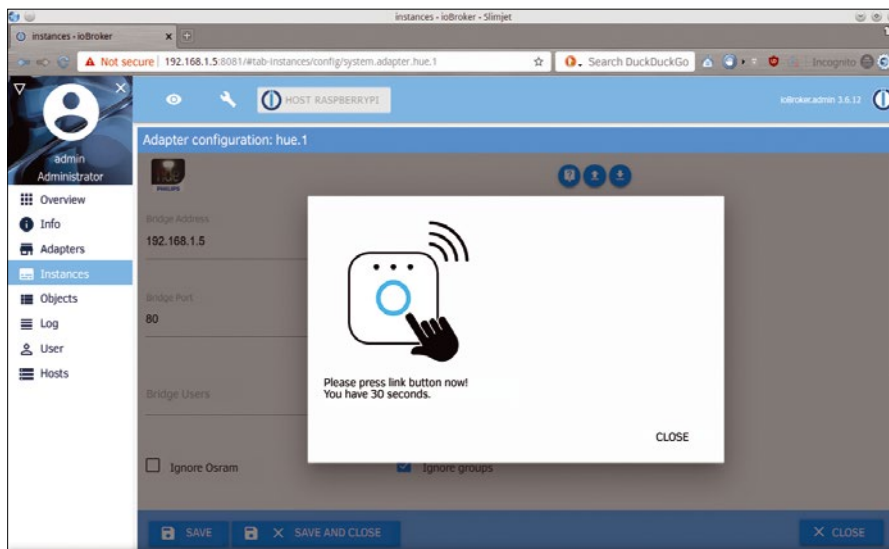
If you call the Vis adapter manually from the web browser, you are taken to a dialog that presents an editor; clicking the *VIS RUNTIME* button opens the preconfigured control interface, which

you can use to switch the individual devices on and off or to display information. A tablet that displays this interface in the browser is ideal for this task, letting you monitor and control your infrastructure conveniently, simply by tapping.

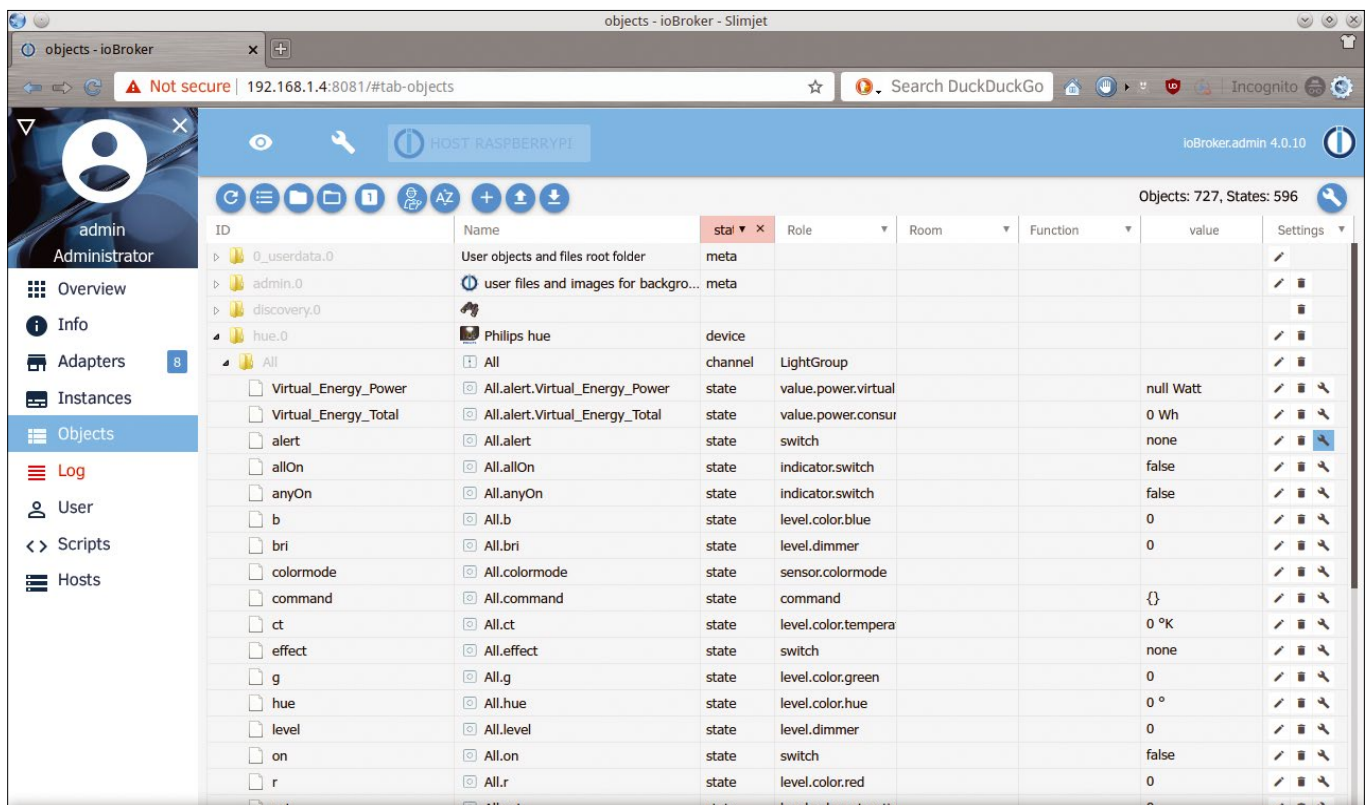
In the Vis interface, you can also include widgets in the editor that provide you with information such as the weather situation or a time display. Widgets are enabled in the same way as traditional device controls [6]. The system lets you freely design the widgets in terms of appearance; the same applies to the Vis interface background. The developers provide basic layout templates that can be freely customized.

## Developing Scenarios

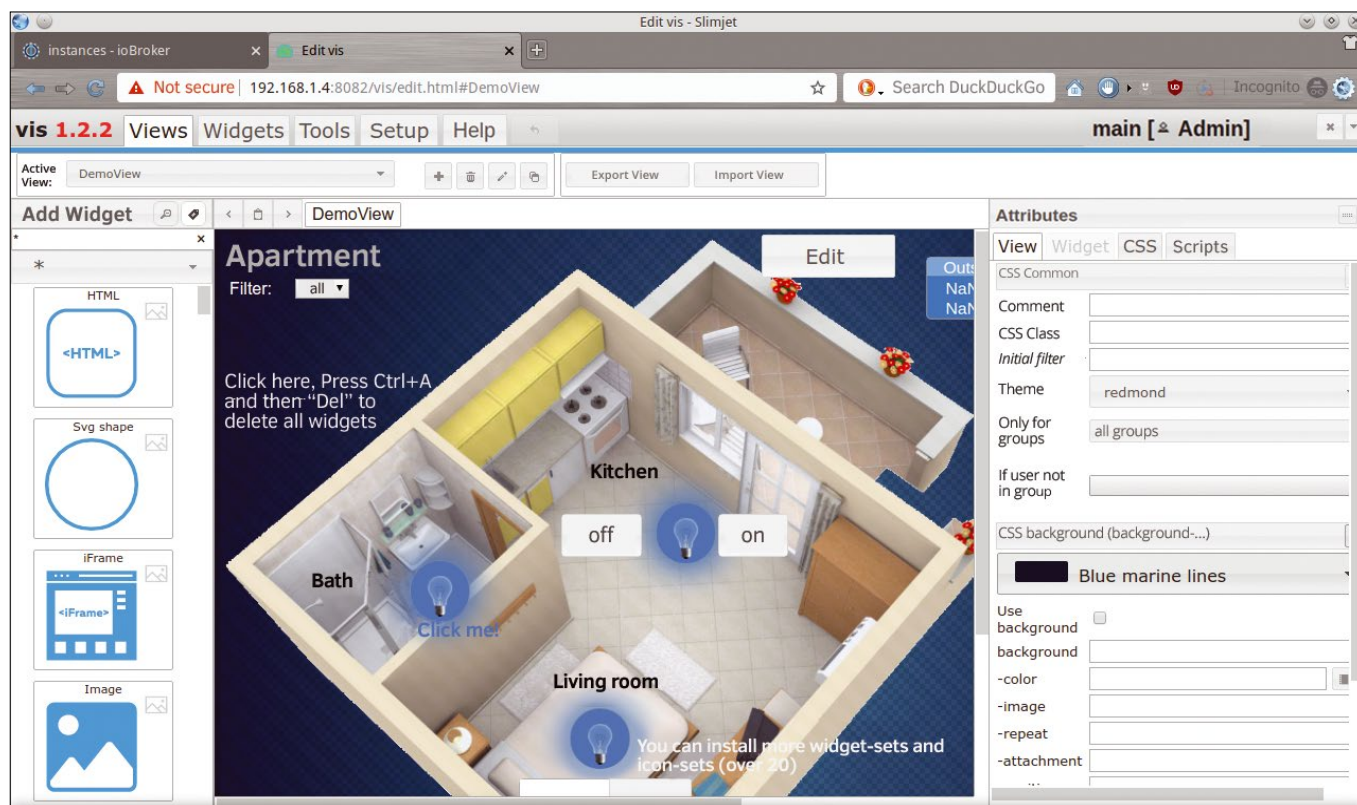
ioBroker lets you program and use scenes to control your device groups individually. To do so, use the *Adapters* selection to integrate the scenes adapter into the system. After doing so, the new *Scenes* option appears in the vertical bar on the left side of the program window. On opening, you are taken to the scene editor, which lets you define individual, changing settings profiles for individual components or component groups.



**Figure 5:** ioBroker guides the configuration with device-specific dialogs.



**Figure 6:** The Objects view shows the bundled device-specific attributes.



**Figure 7:** Build your own smart home control interface in the Vis editor.

ioBroker stores the settings profiles in a tree structure. Once you have captured one or more of these scenes, incorporate them into your control interface in the Vis editor; then, you can manage multiple devices together by predefined scene profiles (e.g., automatically dimming the lights and lowering the blinds for a home theater setting).

## Conclusions

ioBroker and the Raspberry Pi bring order to the jungle of smart home components

from a wide variety of manufacturers. The very compactly programmed and at the same time well-thought-out software requires hardly any resources. Thanks to the many available adapters, it nevertheless unites numerous vendor-specific home automation devices under a unified interface. The only downer is the huge technical complexity of the program because of its feature set, which means a correspondingly steep learning curve for users, for which careful reading of the documentation is necessary. ■■■

## Info

- [1] ioBroker: <https://www.iobroker.net/>
- [2] Installation: <https://www.iobroker.net/#en/download>
- [3] Angry IP Scanner: <https://angryip.org>
- [4] Fing: <https://www.fing.com/products/fing-app>
- [5] Blockly documentation: <https://github.com/ioBroker/ioBroker.javascript/blob/master/docs/en/blockly.md>
- [6] Widgets: <https://github.com/ioBroker/ioBroker.docs/blob/master/docs/en/viz/widgets.md>



If you're too young to remember the old Internet before the web, start up a terminal window and enter some commands at the Bash prompt. The whole Internet used to look like that: no colors, no hyperlinks, no videos, no digital home photos – all a lot clunkier, and all in black and white. But guess what the old Internet *didn't* have: no flashing banners, no cookies, no identity marketing, no ad scripts, and no phishing attacks. The truth is, the power and complexity of today's World Wide Web has led to some practices that many users find downright annoying.

Now a group of devoted but disillusioned web users is fighting back by bringing back the old Gopher protocol, a simple text-based menu system that was often used to publish online information before the arrival of HTML and HTTP. Or, if you're not in a hurry to return to the days of Gopher, the new Gemini protocol has some of the powers of the modern age in a simpler and less dangerous form. Read about these solutions in "Beyond the Web" in this month's Linux Voice.



Image © Olexandr Moroz, 123RF.com

## LINUXVOICE

### Doghouse – Economics 70

*Jon "maddog" Hall*

An affordable open source POS/ERP system has many potential benefits for small businesses.

### The Rise of the Small Internet 72

*Lee Phillips*

The danger and irritations of the modern web have unleashed a movement dedicated to creating a safer and simpler alternative.

### Font Manager 80

*Christoph Langner*

Find the font you're looking for and compare font options side by side.

### FOSSPicks 84

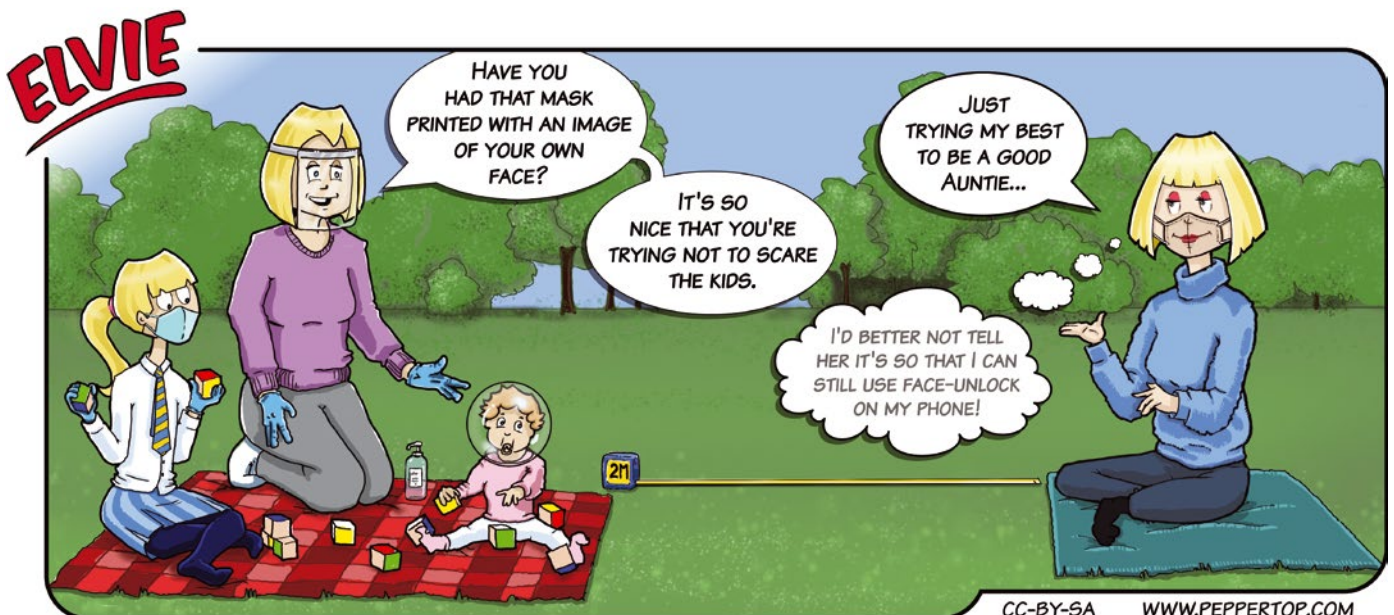
*Graham Morrison*

This month Graham reviews PeaZip, LibreSprite, NeoChat, Beaker, Giada, Thrive, Kurve, and much more!

### Tutorial – ugrep 90

*Karsten Günther*

Searching for text in files or data streams is a common and important function. Ugrep tackles this task quickly, efficiently, and even interactively if needed.



CC-BY-SA WWW.PEPPERTOP.COM

# MADDOG'S DOGHOUSE

An affordable open source POS/ERP system has many potential benefits for small businesses. 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®.

## Small Business Software

**M**y entire professional career has been coupled with what we now call open source or free software. Whether it be software written by users and donated to user groups like DECUS or SHARE, or the Unix system (not really free in any sense), which eventually led to BSD and GNU/Linux, I always appreciated access to the source code so I could get my solutions to work. It is somewhat gratifying that companies that once wailed about free software being communistic are now embracing it and hoping to have "unity" with the FOSSH community (while still making money for their stockholders).

Another thing I've seen happening over more than 25 years is that the number of FOSSH developers has increased from 130,000 (the first time I tallied it up) to millions of developers with hundreds of thousands of projects. Even large commercial companies are using FOSSH to develop products, which creates many good products at a lower cost with even greater profits for their stockholders.

In the past year, there has been a pandemic (you may have noticed), and many small restaurants, bars, hotels, and stores have closed. While we do not yet know how many small businesses will go under due to the pandemic, we can only hope that many are able to continue on and that eventually many new small businesses will emerge. One thing we do know is that successful small businesses are good for the economy.

All of this has me thinking about a particular open source tool that can help make small businesses successful: an inexpensive, flexible POS/ERP system to help owners manage their companies.

Many of you have seen these point of sale (POS) terminals at one of these businesses. It is typically composed of an LCD screen, a receipt printer, a cash drawer (to hold cash and credit card receipts), a scanner, keyboard, mouse, and (perhaps) a USB scale to weigh goods and produce (in the case of a retail store). Often there is some type of electronic payment system also attached.

There may be two, three, or more POS terminals in a store, along with handheld tablets to act as wireless remote terminals.

What is typically not seen is a back-end system called an enterprise resource planning (ERP) system. This system usually provides things such as an inventory control system, customer management system (CMS), accounting, sales, and other business-oriented software. The ERP system may also calculate taxes of different types. The data is typically held in a database as part of a system where modules are put into the backplane.

There are closed source POS/ERP systems from vendors such as NCR and MICROS (a company purchased by Oracle). These systems are typically fairly expensive, and it can also be difficult for the small business to obtain the small changes and extensions that may be necessary (or at least desirable) for their business.

The other problem with closed source systems is that they are relatively hard to migrate from one to another because of a lack of transparency on how they store their data. This creates vendor lock-in. This lack of transparency can also make it expensive to input data from other systems without either buying an expensive module or hiring expensive software tailoring.

There are also open source POS/ERP systems [1] available that have different capabilities and charges. While some of these prices can be mitigated by using gratis modules or hosting the system locally, the charges are typically modest and well worthwhile. What is most important is that they are FOSS software, and most can use off-the-shelf hardware that is available in the open marketplace. Often there are partners associated with these software systems that can give backup support to the customers.

What this software can do is allow a FOSS person to put together a world class solution that they could market to the small business person. The small business person (particularly new ones) will need help in installing, implementing, and populating the system with needed data. There will be a fair amount of upfront training for the business owner and ongoing support. The FOSS person could sell this system to small business owners and maintain an ongoing relationship in creating new functionality.

Purchasing hardware for these systems in quantity would create a full scale POS/ERP system with new hardware at a fraction of the cost that a closed system would cost for used hardware. Relationships with the ERP system companies can supply the training and backup support needed to help customers with more advanced needs.

Now is the time to start learning how these systems work and to start planning for a business in providing them to customers, so that these customers can have successful businesses that employ more people. ■■■

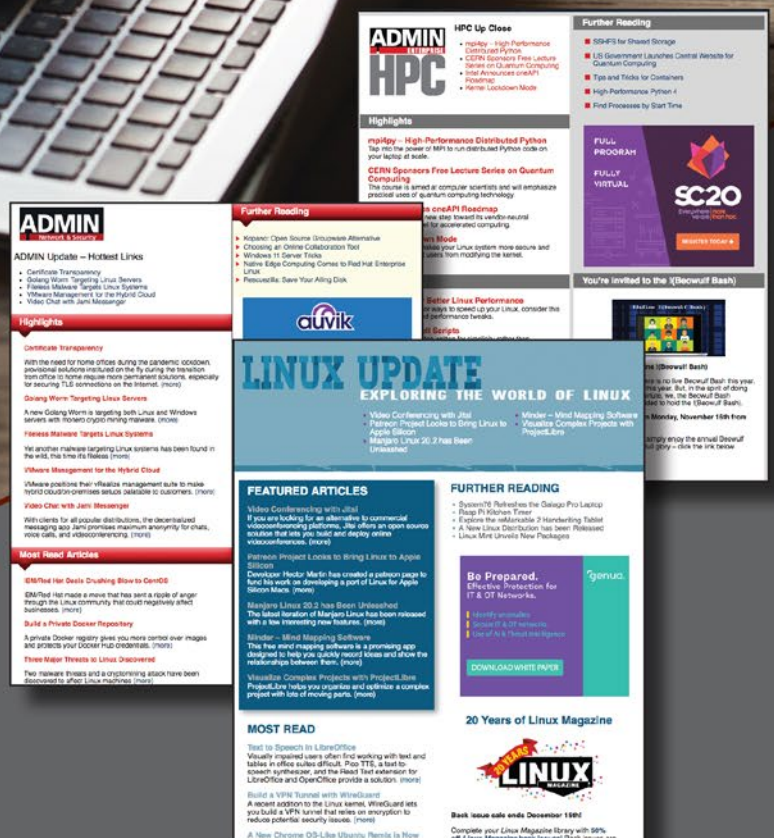
### Info

[1] Open source ERP systems:

<https://opensource.com/tools/enterprise-resource-planning>



# IT Highlights at a Glance



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)



# Gopher, Gemini, and the rise of the small Internet Beyond the Web

The danger and irritations of the modern web have unleashed a movement dedicated to creating a safer and simpler alternative. The old Gopher network and the new Gemini protocol have emerged as building blocks for this new “small Internet.”

BY LEE PHILLIPS

Anyone who has used the World Wide Web (WWW) lately knows that something bad is happening to it. It does not resemble the WWW of the early years, with enthusiastic amateurs freely sharing ideas and information. These things still exist, and the web is still an indispensable medium connecting the world. But the web experience is now encumbered with advertising, invasions of privacy in the form of pervasive tracking, enormous file sizes, CPU straining JavaScript, the danger of exploits, and door slams asking you to subscribe to a newsletter before viewing a site.

This unpleasant environment has led to a backlash. There are now some communities of developers and computer users who still desire a connected information system, but who seek a refuge from the noise, danger, and increasingly resource-hungry WWW. They feel that web technology does too much, and that since it makes various forms of abuse too easy, no lasting reform is possible.

The solution is to use or create a separate protocol that is simply not capable of supporting the technologies that enable advertising networks, user fingerprinting, or the myriad of other things that exploit users rather than helping them. This small movement has approached the problem from two directions that in practice are often merged: the revival of the Gopher protocol and the creation of a new protocol called Gemini.

## A Little History

Looking back at computer networks before the emergence of the Internet in the late 1980s, and even the Internet itself in its early years, there were only a few applications of interest to most users: email, logging on to timesharing systems, and the transfer of documents through the file transfer protocol (FTP).

FTP began as a command-line program, with graphical clients eventually arriving for every

platform. To get a file, you had to know its specific address on the FTP server where it resided. Eventually, it got a bit easier to find things with the arrival in 1990 of Archie, the first public search engine on the Internet, which indexed FTP files.

A command in the terminal would download the file to your machine. You could navigate through directories if they were exposed on the server, but you could not navigate outside the server, discover content, or interact with the files before downloading them. FTP was essentially a networked version of the `cp` command. It was, for years, the only way to share content, aside from passing around tapes and floppy disks; FTP was the way the first versions of the Linux kernel were distributed [1].

Shortly after the Internet's predecessors evolved into the modern global network, other means of sharing content emerged that offered more than FTP's spare file copying. The three most significant participants in this information landscape were the Wide Area Information Server (WAIS), Gopher, and the WWW.

WAIS was a sophisticated software system [2] that translated the user's natural language query into search commands for databases. From a networked computer, you could search the contents of databases all over the world and retrieve a variety of documents. There were textual and graphical clients for major operating systems, with varying degrees of polish.

WAIS was developed by Thinking Machines Corporation [3], a visionary company that created the first massively parallel supercomputers. WAIS is completely dead now. Even if a client could be found that ran under the current version of any OS, there is no content and no servers to interact with.

Gopher, on the other hand, is still being used. As Cameron Kaiser points out [4], “Gopher takes the strict hierarchical nature of a file tree or FTP and

turns it into a friendlier format that still gives the fast and predictable responses that they would get by simply browsing their hard drive.”

And that serves to explain the nature of Gopher: Like FTP, it was a way to display directories of files that could be downloaded, or in the case of text files, displayed directly by the client.

Gopher clients went beyond simply downloading files, because they had some notion of file type, and could respond appropriately, in a limited way. While FTP servers only presented lists of files and directories local to that server, Gophermasters could create directories that appeared as pages of links and text. The links could lead to other servers, creating the first easily accessible global network of information. These directories were not hypertext documents, however; they contained isolated links and descriptions of content.

Around 1990-91, the first WWW browsers appeared on computers all over the world, and in another year or two the web had surpassed FTP, WAIS, and Gopher as the medium for finding information and entertainment through the Internet.

At that time, I was working in a government laboratory, enjoying the luxury of the kind of fast Internet connection that most of the developed world takes for granted now. I had several FTP clients and at least one Gopher and WAIS client on my laboratory computer. The reasons for the rapid ascendancy of the web are not mysterious to me, because I distinctly remember watching it happen in real time.

As soon as people saw hypertext, it captured their imaginations. That is the magic of the web: to click on a word in a document and be taken somewhere else for further enlightenment. And these were not merely text documents; they could be styled to look like magazines. When inline images and other media appeared, it sealed the deal. In the lab, we started making our own pages to share information with other departments. This was the original purpose of the web, for scientists to share research with each other, and in our lab it was a reality.

Everyone I knew with a computer, including laypeople who had never heard of FTP or WAIS, was soon browsing the web. The WWW had replaced, more or less, the existing protocols and became the standard way to share information.

In 1992-93, as the WWW was just beginning to surpass WAIS and Gopher in network traffic, the WWW's inventor, Tim Berners-Lee, wrote a note explaining the differences among the three systems [5]. Berners-Lee pointed out the advantages of hypertext over the other methods of distributing information and emphasized an additional crucial virtue of the WWW – it could be used as an interface to Gopher or WAIS and was

therefore a superset containing the other two protocols. Thus the web rendered Gopher and WAIS obsolete in a technical sense in addition to replacing them as a preference on the part of users. Fast-forward to today, and even Tim Berners-Lee thinks the web has gone too far. Berners-Lee is one of many experts who have spoken out against the perverse incentives and lack of privacy that characterize today's web [6], and he has even launched an initiative to try to fix it [7].

The first murmurings of a revival of old protocols as reaction to the excesses of the web mentioned Gopher [8] as a plausible candidate for a refuge from the storm. (See the box entitled “URLs and Protocols.”) Some advocates for the Gopher revival speak of a “community outside the chaos of the modern Internet.”

This is a relatively easy way to build an alternative Internet community, because the Gopher universe never completely disappeared. There are clients that still work, or can be updated to work, on current operating systems.

Now Gopher is growing again, specifically because it is not the web. The number of Gopher servers is again increasing, a development that would have seemed highly unlikely as recently as five years ago. Because it does not support any kind of scripting, much of the abuse of the modern web can not occur. The lack of embedded media removes the possibility of most forms of advertising or other irritations. And its very retro nature gives the use of Gopher an inherent cool quality.

## Enter Gemini

Project Gemini [9] has no interest in replacing either the web or Gopher but seeks to live alongside them as another option for disseminating and consuming content. It carefully positions itself between Gopher and the WWW in its complexity and capabilities. The creators of Gemini realized that Gopher lacks certain features that are desirable for dissemination of even simple documents over the web, such as encryption and the inline inclusion of images along with

### URLs and Protocols

The familiar URL, like *https://google.com*, contains several parts. The part before the colon is the protocol, in this case secure HTTP. The new protocols that go beyond the web borrow this URL structure. Gopher addresses look like *gopher://example.com*, Gemini locations like *gemini://example.com*, and FTP sites like *ftp://example.com*. You can configure your browser to open each protocol with a specific application.

text. But they deliberately excluded from the specification [10] web features such as scripting and complex document models that are more likely to enable abuse of the information consumer. To this end the protocol is designed to be difficult to extend, so that its protections against abuse will endure. One example of Gemini's emphasis on privacy is that connections are required to be encrypted. Another is that the protocol has no provision for User-Agent or Referer headers, and it is specifically designed not to be extensible to include them.

It has several advantages over the older Gopher protocol, chiefly in the use of non-ASCII character sets, using MIME types to handle non-text content, redirects, and virtual hosting; the requirement to use TLS encryption; and links within documents.

The type of link implemented by Gemini is not freely mixed in with the document as in HTML; instead it sits on a separate line. The Gemini designers see this as an advantage, as it helps in the organization and outlining of documents. We'll take a closer look at the native Gemini file format in Listing 1, when we try out one of the Gemini browsers.

This article will not go into setting up content servers, but those interested in serving Gemini content will find some free, hosted avenues for doing so at the Gemini FAQ.

The Gemini space is still small, but growing rapidly. For historical interest, the project exposes a list of the first 50 known Gemini hosts at (naturally) a Gemini address. Up-to-date information, including an automatically generated list of Gemini hosts, is available at the Gemini Universal Search (GUS) search engine.

Still, there are, according to GUS, about 200,000 Gemini pages served from 422 domains, compared with over a billion pages on the World Wide Web. Consequently, Gemini is not a place to find information or do research, nor even to browse interesting content in depth. Today it is still a place to explore a new idea in information sharing and perhaps to enjoy wandering through a small but enthusiastic networked community while knowing that your privacy and security are being respected and without the irritations of the modern web. Gemini fills a niche, is well executed, and has been

thoughtfully designed. Its limitations are very much deliberate choices. I believe it will continue to grow and stands a reasonable chance of becoming a real second place to find and share information on the Internet.

The new Gemini protocol, along with the revival of Gopher, and the attitudes and community that go with them, are sometimes called the "small Internet," in contrast to the WWW, which is obviously anything but small. This nascent movement also embraces the venerable Unix Finger protocol. The `finger` command in Unix machines returns a small amount of text indicating the status of a user or a machine. Despite its simplicity, it has been used by creative individuals as a minimalist communications medium.

Part of the appeal of Gemini is the same thing that makes Gopher attractive to some people: The very fact that it is still small and new makes browsing the available content seem like exploring a new universe, somewhat like the feeling of exploring the web in the early years.

## Linux Gopher Clients

VF-1's motto [11] is "High speed, low drag." It is a basic terminal client, similar in operation to the command-line FTP clients of olden times. VF-1 is written in Python and can be installed with:

```
pip3 install VF-1
```

After the user types `vf1` in a terminal, a brief welcome and a simple prompt appear. If the user guesses correctly and types `help`, he or she will be rewarded with a list of the available commands (Figure 1). One of these is `veronica`, which submits a search term to the Gopher search engine of that name. Navigating to results is accomplished by typing the number displayed next to the file name and hitting return, after which the file contents are dumped to the screen. Alternatively, the user can view the file in the `less` pager or save it to disk.

If you know where you're going, the `go` command will take you right there. In Figure 2, I've used this command to navigate to a Gopher mirror of the *Hacker News* front page. Gophermasters often adorn their directory pages with ASCII art to personalize them in the absence of inline images.

The next step up in convenience from a pure command-line program like VF-1 is a curses-style interface. These programs are still lean and fast, and they run in the terminal, but through random access to the screen area they allow a richer mode of interaction and display.

Phetch is a Gopher client in this mode written by Chris West. Binaries and source are available on GitHub [12]. After typing `phetch` in the terminal, the program presents the startup screen shown in Figure 3. Navigation is accomplished through the keyboard, with links and text being distinguished by

```
Welcome to VF-1!
Enjoy your flight through Gopherspace...
VF-1> help

Documented commands (type help <topic>):
=====
add      exit    handler  links   previous  save    tls    veronica
back     fold   help     ls      quit      search  tour
bookmarks forward history mark    reload    set     up
cat      go     less     next   root      shell   url

VF-1> █
```

Figure 1: A list of the available commands for the VF-1 Gopher client.



color. As this is Gopher, text is all there is, so the terminal remains a reasonable environment for consuming the content, as shown in Figure 4. Phetch does what it was designed to do with an intuitive, efficient interface. I found it a pleasant way to explore the Gopherspace and encountered no bugs or rough edges of consequence.

There appear to be no existing graphical Gopher clients for Linux that are not broken or too ridden with bugs to be useful.

FORG [13] is a graphical Gopher client written in Python. The GUI uses tkinter. I was able to start FORG by executing the main Python file after downloading the repository from GitHub, but first I had to use pip to install a dependency, `Pmw`. When I tried to use it, I found that FORG displayed its configured home page with no problem, but any link I followed resulted in a blank page and error messages written to the console. I hope that this program gets these issues resolved, because it has some valuable features.

Another GUI client too buggy to use in my testing on Linux, but apparently popular on macOS, is called Little Gopher Client.

Finally, some web browsers, including text-mode browsers such as Lynx and ELinks, have

support for the Gopher protocol, so it's likely you can browse Gopherspace without downloading any new software. There is also a growing number of Gopher extensions for browsers that don't come with support out of the box.

### Linux Gemini Clients

Jaakko Keränen's Lagrange [14] is a highly polished GUI Gemini client. One minor warning: On high-resolution screens, Lagrange's user interface will be unusably small. Fortunately, there is a preference item under **Window** called UI scale factor that works perfectly, although the program needs to be restarted before it takes effect.

There was only one other tiny glitch: I have my Caps Lock key mapped as a second control key, which works in every other application, but is not recognized in Lagrange. However, there is an easy-to-use preference item for remapping the keyboard shortcuts, and with this it only took a few seconds to fix this glitch for most shortcuts. Regrettably, certain others can not be remapped; so I still need to use the pointing device to enter an address, for example.

Lagrange embeds images linked from a page into the display, rather than opening a new win-

indow. An important part of the Gemini protocol is that the clients will not make network requests that are not manually initiated by the user, so normally the reader will click on each image of interest. However, Lagrange offers a preference setting that loads images when they are scrolled into view, reproducing an experience more like a conventional

```
VF-1> go gopher://hngopher.com/

hngopher.com:70
Hacker News
- The Underground Hacker News Mirror
(updated hourly)

This is a mirror of the popular tech news aggregator Hacker News
(https://news.ycombinator.com). All stories and discussions have
been converted to plain-text ASCII. Have fun exploring the site!

[1] Live Feed/
(last update 2020-12-22 17:04:07 UTC)
[2] Archive/
[3] Guestbook/
[4] Information/

We're not counting, but you're visitor number 25387
VF-1>
```

Figure 2: The *Hacker News* Gopher mirror in VF-1.

```
phetch

~ * ~

* 1. search gopher
  2. gopherpedia
  3. gopher lawn
  4. welcome to gopherspace

~ * ~

5. show help      (ctrl-h)
6. show history   (ctrl-a)
7. show bookmarks (ctrl-b)
```

Figure 3: The startup screen for phetch, a Gopher client that runs in the terminal.

web browser. With this setting selected, images are inlined when scrolling with the arrow key or the spacebar, but the user can use the scrollbar to scroll without triggering image loading. This is another example of Lagrange's pleasant and well thought out user experience. Lagrange can also embed an audio player in the displayed page.

Lagrange can display a navigable outline of the page in a sidebar, but it requires the mouse to jump between page sections. The two sidebars can also display other information, such as bookmarks, history, and TLS client certificates. This mature browser also lets you search for text on the page. It is quite configurable through its preferences dialog, allowing the user to choose or reject smooth scrolling and set many aspects of the appearance and user interface. It has tabs as well.

Lagrange has keyboard navigation that works like link hinting in Vimperator and similar systems, but it is even easier to use. It caches recent pages, so going back and forth is instantaneous.

In Figure 5, you can see (both) results displayed in Lagrange from a search for "cornea" using GUS. The rendering shows most of the common elements that a user may encounter on a Gemini page: regular text, two different levels of headings, links both within the server and to pages on other servers, and a quotation. It is up to the client to decide how to display these various elements. Lagrange distinguishes server-local and remote links with a triangle or a globe, shows link destinations when hovering, shows you what links you have visited by coloring them differently, displays link hints when hitting *F* that replace the triangle

or globe symbols, and has special formatting for quotations.

Let's take a look now at the markup language used by Gemini. The source for the page in Figure 5 is shown in Listing 1. Readers familiar with Markdown will recognize the inspiration for the syntax, but this markup format, called "Gemtext," has some important differences from Markdown and from HTML.

The most unusual aspect to Gemtext markup is that it is based on single lines, with each line possessing one type. Paragraphs of normal text exist as single long lines, wrapped by the client for display. The longer lines are wrapped here to fit the format of the page. The other major line types are signalled by the characters at the start of the line. Headers are indicated by *#* characters, links by *=>*, and quotations by *>*. By design, links are on their own lines and can not be intermixed with text, as mentioned above. Gemtext therefore is deliberately designed to not be a full hypertext format.

The line-oriented specification has two purposes: to create a well-structured, organized document, and to make it as easy as possible to write parsers for the format, making it more likely that clients will proliferate. It is possible to write a Gemini client in under 100 lines of code in a high-level language like Python.

Amfora [15] is a browser exclusively for Gemini, like Lagrange, but running in the terminal rather than in a GUI. It is distributed as a simple binary for Linux, all BSDs, macOS, and Windows, as well as, of course, source code. Since Amfora is navigated solely through the keyboard, some form of



Figure 4: A page from the Gopherpedia, the Gopher interface to Wikipedia, shown in pfetch.

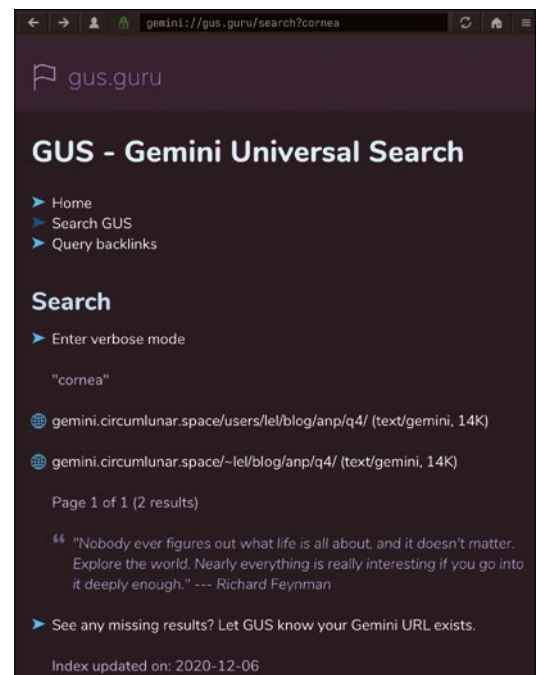


Figure 5: The results from searching "cornea" in the Gemini Universal Search, as shown in the Lagrange Gemini browser.

link hinting is required. As shown in Figure 6, Amfora supplies numerals beside each link in the document display. To follow a single-digit hint, the user need merely type the digit; to disambiguate multiple-digit hints, hitting the spacebar takes the user to the address area at the bottom. This system of navigation is quick and efficient. Amfora does not display images inline, as might be expected. Selecting an image will give the user a choice to download it or display it in the default external application. Amfora has tabs, handled neatly in the text interface, as shown in Figure 7. I could find no way to search within a page, however, which is a serious omission.

In my trial of Amfora, I found that it had no problem following links and quickly navigating backward and forward, but that there was an issue in handling forms, as queries to the GUS search engine became stuck and never returned results. This was not a problem with the server, as

identical queries at the same time from Lagrange returned results almost instantly. However, the problem only exists when trying to use the dialog box that pops up when querying GUS through its search link. If you enter Amfora's command area by hitting the spacebar, you can enter search terms there, and pressing the return key sends them directly to GUS as URL parameters and quickly displays the returned results.

Finally, if you would like to browse Geminispace content without installing one of the small Internet browsers, there is a portal for the WWW [16] that works in a normal web client.

### Combined Clients

Several clients exist that can handle both Gopher and Gemini (and sometimes Finger, as well).

One of these is called Bombadillo [17], a terminal program that can handle all three protocols, plus local files. Bombadillo can be compiled from source on any Unix-like operating system if you have a Go compiler installed, and there are pre-compiled binaries available for several processor architectures on Linux and for macOS.

### Listing 1: A Gemtext Example

```
# GUS - Gemini Universal Search

=> / Home
=> /search Search GUS
=> /backlinks Query backlinks

## Search
=> /v/search/1?cornea Enter verbose mode

"cornea"

=> gemini://gemini.circumlunar.space/
users/1el/blog/anp/q4/ gemini.
circumlunar.space/users/1el/blog/anp/q4/
(text/gemini, 14K)

=> gemini://gemini.circumlunar.space/
~1el/blog/anp/q4/ gemini.circumlunar.
space/~1el/blog/anp/q4/
(text/gemini, 14K)

Page 1 of 1 (2 results)

> "Nobody ever figures out what life is
all about, and it doesn't matter. Explore
the world. Nearly everything is really
interesting if you go into it deeply
enough." --- Richard Feynman

=> /add-seed See any missing results?
Let GUS know your Gemini URL exists.

Index updated on: 2020-12-06
```

```
# Project Gemini

## Overview

Gemini is a new internet protocol which:

• Is heavier than gopher
• Is lighter than the web
• Will not replace either
• Strives for maximum power to weight ratio
• Takes user privacy very seriously

## Resources

[1] Gemini documentation
[2] Gemini software
[3] Known Gemini servers
[4] Gemini mailing list
[5] Gemini client torture test

## Web proxies

[6] Gemini-to-web proxy service
[7] Another Gemini-to-web proxy service

## Search engines

[8] Gemini Universal Search engine
[9] Houston search engine

## Geminispace aggregators

[10] CAPCOM
[11] Spacewalk

## Gemini mirrors of web resources

[12] A list of mirrored services

## Free Gemini hosting

gemini://gemini.circumlunar.space/
```

**Figure 6:** Information about Project Gemini, displayed in the Amfora Gemini browser.

```
# New Tab

You've opened a new tab. Use the bar at
the bottom to browse around. You can
start typing in it by pressing the space
key.
```

**Figure 7:** The Amfora browser, which runs in the terminal, offers some interesting features, including the tabs shown here.



Figure 8 shows how Bombadillo renders the GUS search results page that we saw Lagrange handling above. It uses numerals for navigation, using the same system as Amfora for following links with either one digit or more. Unlike Amfora, Bombadillo does not color links, and, as Figure 8 shows, does not break lines correctly, splitting words at arbitrary places. Also unlike Amfora, Bombadillo handled the query to the search engine with no problem, using a one-line command area at the bottom of the window to accept the search terms. Like the other clients reviewed here, Bombadillo has back and forward navigation and a way to save the user's bookmarks. Operation is quick and painless.

Another graphical client supporting Gemini, Gopher, and Finger is called Castor [18]. It is written in Rust using the GTK GUI library.

### Have Fun!

The new phenomenon of the "small Internet" is still not a place to go for serious research. But it is, today, an interesting place to explore. After the cacophony of the modern web, browsing through

Gopher or Gemini land feels like a quiet stroll on a remote beach, where you might just find the occasional beautiful shell lying on the sand.

There are good reasons to expect Gemini, in particular, to evolve into a viable alternative to the modern web. For now, I intend to keep a couple of small Internet browsers installed and visit the beach on occasion. ■■■

### Info

- [1] "25 Years of Linux" by Brian Proffitt, Red Hat Blog, 2016: <https://www.redhat.com/en/blog/25-years-linux>
- [2] WAIS: <https://www.ou.edu/research/electron/internet/wais-faq.htm>
- [3] "The Rise and Fall of Thinking Machines" by Gary A. Taubes, *Inc.com*, 2020: <https://www.inc.com/magazine/19950915/2622.html>
- [4] "Why Is Gopher Still Relevant?" by Cameron Kaiser: <https://gopher.floodgap.com/overbite/relevance.html>
- [5] "W3 vs. WAIS and Gopher" by Tim Berners-Lee, W3.org, 1992: <https://www.w3.org/History/1992/WWW/FAQ/WAISandGopher.html>
- [6] "30 Years On, What's Next #ForTheWeb": <https://webfoundation.org/2019/03/web-birthday-30/>
- [7] Contract for the Web: <https://contractfortheweb.org/>
- [8] "The Web May Have Won, but Gopher Tunnels On" by Nate Anderson, *Ars Technica*, 2009: <https://arstechnica.com/tech-policy/2009/11/the-web-may-have-won-but-gopher-tunnels-on/>
- [9] Project Gemini: <https://gemini.circumlunar.space/>
- [10] Gemini protocol specification: <https://gemini.circumlunar.space/docs/specification.html>
- [11] VF-1 command line Gopher client: <https://github.com/solderpunk/VF-1>
- [12] phetch: <https://github.com/xvxx/phetch/releases/tag/v1.0.0>
- [13] FORG on GitHub: <https://github.com/Tom4hawk/FORG>
- [14] Lagrange Gemini client: <https://git.skyjake.fi/skyjake/lagrange>
- [15] Amfora terminal browser for Gemini: <https://github.com/makeworld-the-better-one/amfora>
- [16] Gemini portal: <https://portal.mozz.us/gemini/gemini.circumlunar.space/>
- [17] Bombadillo: <https://bombadillo.colorfield.space/>
- [18] Castor browser for Gemini, Gopher, and Finger: <https://sr.ht/julienxx/Castor/>

### The Author

Dr. Lee Phillips is a theoretical physicist and writer who has worked on projects for the Navy, NASA, and DOE on laser fusion, fluid flow, plasma physics, and scientific computation. He has written numerous popular science and computing articles, as well as technical publications, and he is engaged in science education and outreach.

```
(( ( Bombadillo )) )  gemini://gus.guru:1965/search?cornea
# GUS - Gemini Universal Search

[1]  Home
[2]  Search GUS
[3]  Query backlinks

## Search
[4]  Enter verbose mode

"cornea"

[5]  gemini.circumlunar.space/users/lel/blog/anp/q4/ (text/gemini, 14K)
[6]  gemini.circumlunar.space/~lel/blog/anp/q4/ (text/gemini, 14K)

Page 1 of 1 (2 results)

> "The ships hung in the sky in much the same way that bricks don't." --- Douglas Adams

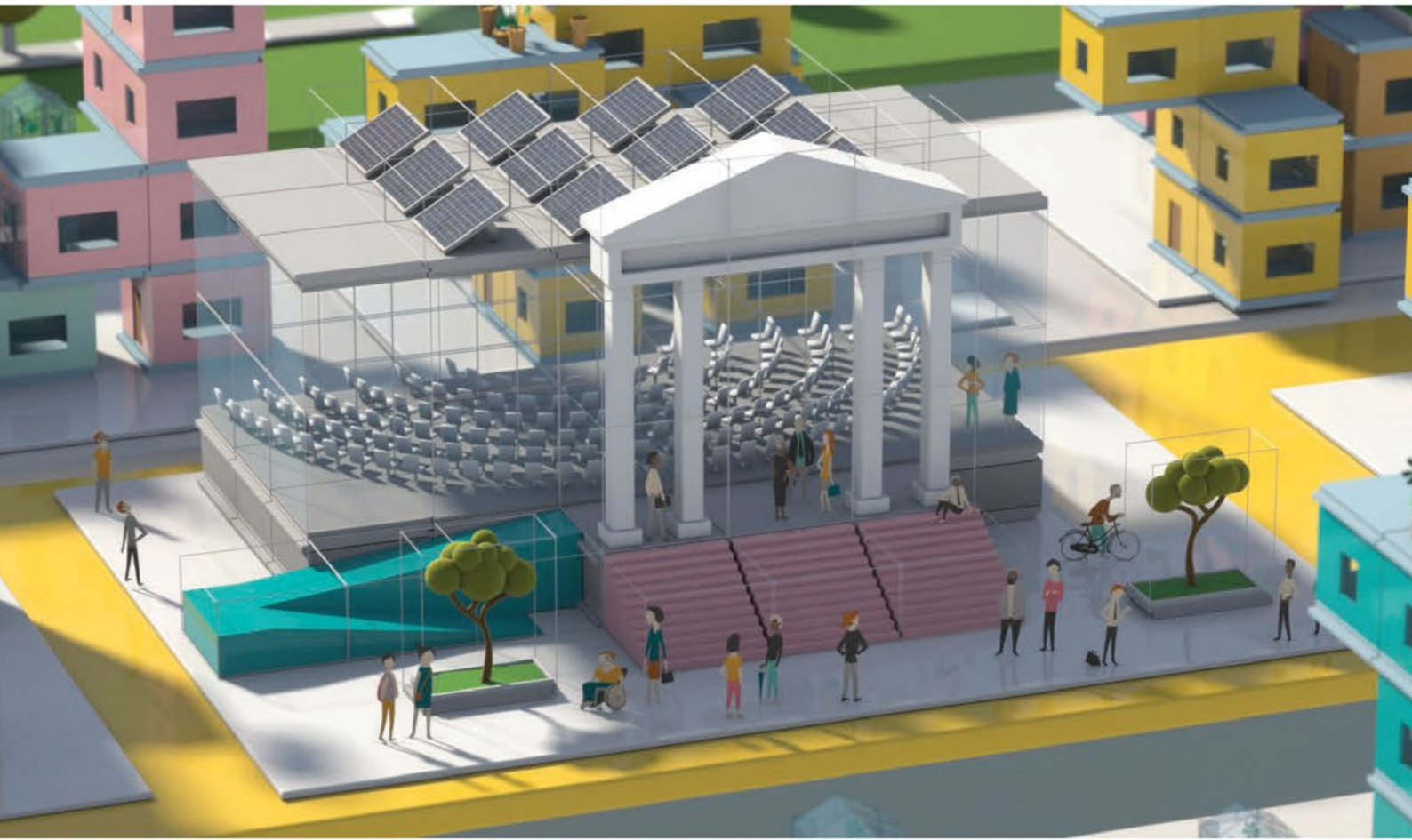
[7]  See any missing results? Let GUS know your Gemini URL exists.

Index updated on: 2020-12-06
```

Figure 8: Bombadillo's rendering of a GUS search results page.

# Public Money

# Public Code



## Modernising Public Infrastructure with Free Software



Free Software Foundation Europe

**Learn More:** <https://publiccode.eu/>

# Simplify font selection with Font Manager

## Font Roundup

Font Manager makes it simpler to find the specific font you're looking for and to compare font options side by side. **BY CHRISTOPH LANGNER**

There are a few things that you can rely on to accumulate on a computer like fluff under a living room couch, and fonts are one of them. If you do any kind of design work, you may be dealing with a huge number of different fonts. It's easy to lose track of the fonts installed on your system, especially since not all programs display a preview of the font in the selection dialog. If you're looking for a specific font but can't remember the name, or if you just want to test different fonts, you would normally have to try out the individual fonts one by one. Using the open source Font Manager [1] program can greatly simplify working with your font collection.

### Installation

The current version 0.8.0 of Font Manager comes with numerous innovations, which I will discuss in the course of this article. The best approach is to install the latest version if possible. The developers offer pre-built package sources for Fedora and Ubuntu (Listing 1), as well as a cross-distribution Flatpak. Arch Linux users will find the program in the Arch User Repository. The project provides up-to-date information on the install on its GitHub page, as well as a build guide for users who want to compile the program themselves.

When launched, Font Manager comes up with an easy to navigate window (Figure 1). At the top of the right column, the program shows you all the fonts installed on your system. You can use the down arrow to unfold the supported font styles (standard, italic, bold, etc.). Below that, Font Manager displays a preview of the currently selected font in different sizes, in what is known as a waterfall.

In the left column, you can filter the selection by different criteria. Like with all modern Gnome applications, the most important switches and dialogs are found in the window's titlebar. You can access the settings via the wrench

icon. From the drop-down dialog next to it, you can export all the settings in *User Data* and load them again later as needed.

Using the tabs in the preview area, Font Manager displays a *Characters* view with all the characters supported by the font, in addition to the *Waterfall* view. The *Properties* tab provides further details, such as copyright information or the size of the font file. The *License* tab displays the entire license text for this purpose and provides a link to further explanations. The icon with the three stacked dots lets you switch between the familiar waterfall, a freely editable preview, and a Lorem ipsum text, which is dummy text in pseudo-Latin used as a placeholder in layouts.

Running the Font Manager in *Manage* mode lets the user disable and enable fonts or install new ones. If you press the button in the top left corner, the program also gives you the option of displaying all the fonts in a kind of map or list mode via *Browse*. *Compare* mode lets you specifically select some fonts and then displays them in detail one below the other to help you find the most appropriate font for a document (Figure 2).

### Font Management

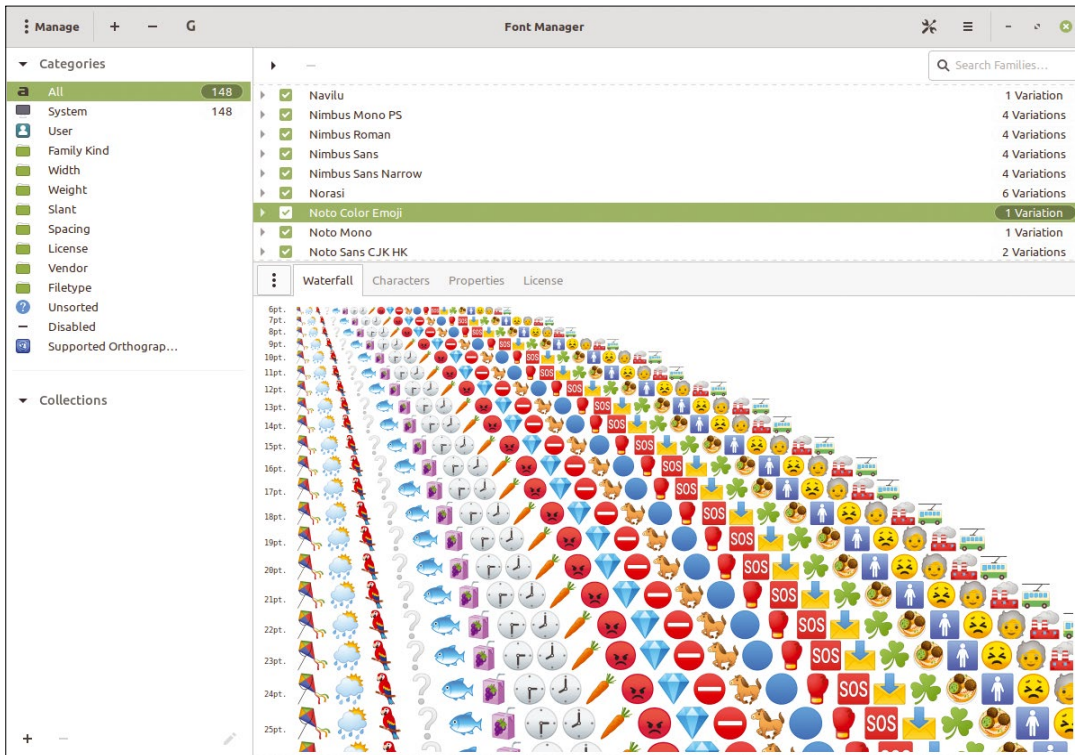
When installing fonts from the package sources, you may automatically download numerous font variations for specific languages. For example, the *noto-fonts* package in Arch Linux contains numerous language-specific font styles, including variants

#### Listing 1: Installation

```
### Install current version on Fedora:
$ dnf copr enable jerrycasiano/FontManager
$ dnf install font-manager

### Install current version on Ubuntu:
$ sudo add-apt-repository ppa:font-manager/staging
$ sudo apt update
$ sudo apt install font-manager
```



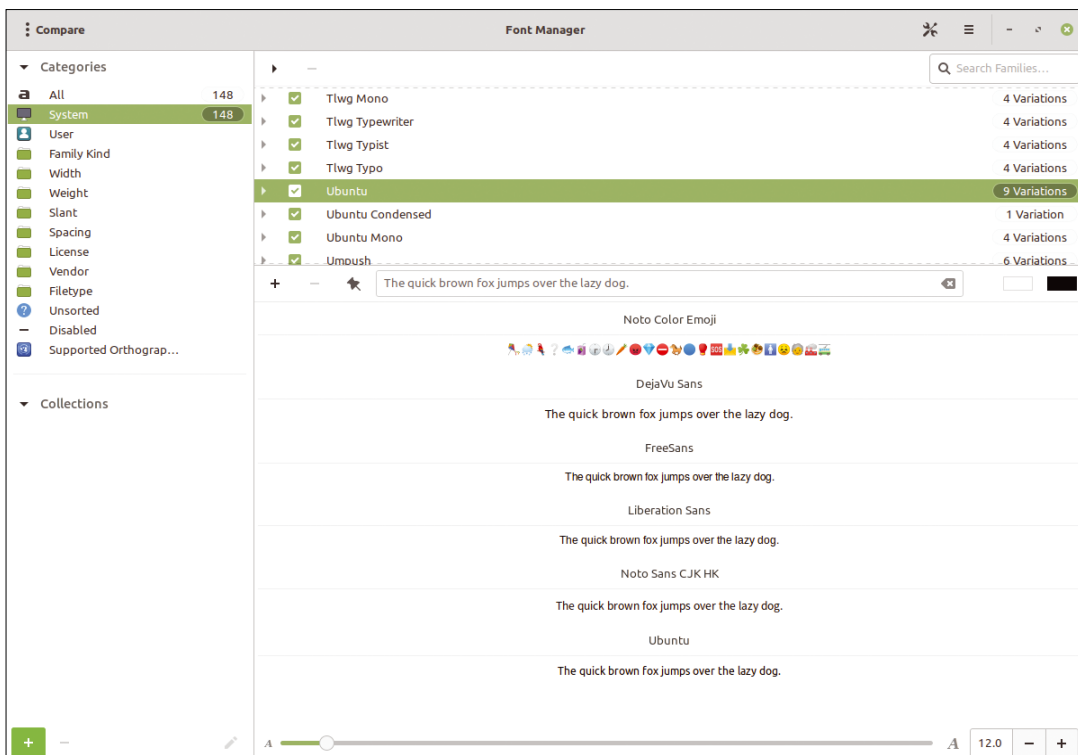


**Figure 1:** In Font Manager, unneeded fonts can be disabled at the push of a button without deleting them completely from the system.

for common European languages, as well as for the West African Fula language and for Yi, a language spoken in China and Vietnam.

This flood of fonts unnecessarily complicates the selection of a font in programs such as LibreOffice Writer – even if you need to compose

documents in multiple languages, you’re unlikely to need all the available font variants. By clicking on the blue-colored check mark, unused fonts can be disabled for the current user. The program then colors the corresponding font gray and crosses out the entry, but does



**Figure 2:** In Compare mode, you can identify the best font for the current project in no time at all.

### Fonts and Yet More Fonts

Linux organizes fonts on the filesystem in `/usr/share/fonts/` as a system global repository for all users and in `~/.local/share/fonts/` (formerly `~/.fonts/`) specifically for each user. Font Manager displays all fonts, but only installs new fonts to `~/.local/share/fonts/` in the home directory of the active account. From there, the fonts can also be deleted from the program. However, the application ignores fonts that still exist in `~/.fonts/`. If in doubt, it is recommended to move all font files to `~/.local/share/fonts/`.

not delete the font. After restarting LibreOffice, the font is no longer displayed there.

If you only occasionally need some of the fonts, you can manage them by grouping them in collections. To do so, switch from *Categories* to *Collections* below the left column and press the plus button to create a new entry – for example, “Funny” for funny fonts suitable for designing an invitation to a children’s birthday party. Then switch back to the *Categories* view and drag and drop the fonts you want into the appropriate collection. As soon as you “grab” a font, Font Manager automatically switches to the *Collections* view. You can now disable all of the “Funny” fonts

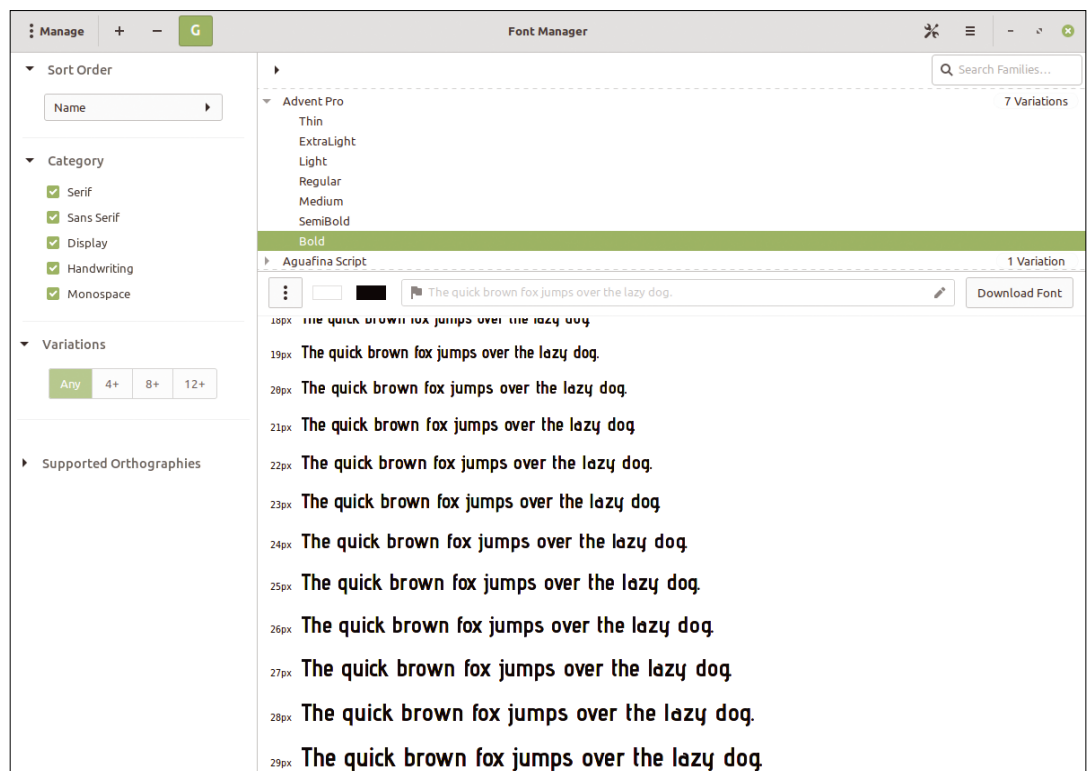
with a single click and re-enable them for the system just as easily whenever required.

You also can add new fonts to the system in Font Manager by pressing the plus button in the titlebar. Then use a file manager to select the desired font files from the hard disk. Optionally, the program automatically extracts fonts stored in archives (such as in ZIP or GZ format). Font Manager automatically transfers the fonts to `~/.local/share/fonts/` making them available in all applications (see the “Fonts and Yet More Fonts” box). In Font Manager, the fonts installed in this way appear in the *User* category. A system global setup for all users is currently not supported in Font Manager. The minus button can be used to remove manually installed fonts.

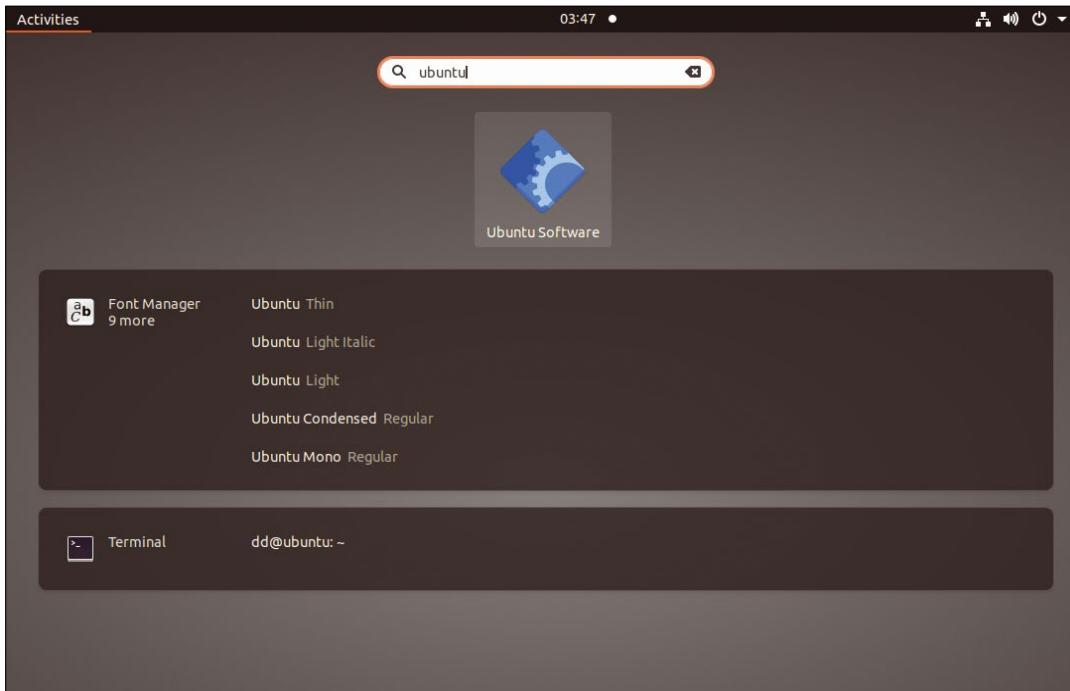
### Latest Features

Font Manager v0.8.0 now integrates the Google Fonts library [2]. Most of the fonts in this collection are under the SIL Open Font License; some use the Apache license instead, while others use the Ubuntu Font License. But, as a rule, you can use the fonts freely. You can access the Google Fonts by pressing the button with the typical Google “G” in the titlebar (Figure 3). Font Manager then displays all the fonts registered with Google in a dialog and lets you download them to the system individually or as a font family.

The latest version of Font Manager also tries to integrate more closely with the Gnome desktop. The program adds itself to the search function. If you press the Super key (also known as the Windows



**Figure 3:** Font Manager integrates locally stored fonts into the system and also offers a feature for setting up fonts from the Google Fonts library.



**Figure 4:** The Font Manager developers have taken great care to integrate the program into the Gnome desktop, including in the search function.

key) and then enter one of the installed fonts for the search, the system lists the matches (Figure 4). Pressing the Enter key then opens the font in Font Manager. If you do not need this feature, you can disable Font Manager integration in the Gnome settings below *Search | Font Manager*.

## Conclusions

Most desktop environments in the Linux universe at best support one-click font installation, but lack features to handle the flood of fonts and font variations. This is where Font Manager comes in

handy, especially on the Gnome desktop. The program makes it easy to browse through the fonts that are already installed. It also supports user-specific deactivation of fonts installed by the admin. This means that you can have fonts available for special occasions without them being too distracting in everyday life. ■■■

## Info

- [1] Font Manager: <https://fontmanager.github.io>
- [2] Google Fonts: <https://fonts.google.com>





# FOSSPicks

Sparkling gems and new releases from the world of Free and Open Source Software



Not content with reliving his youth through retro gaming, Graham recently used OpenSCAD to design and build a bike ramp to practice bike jumps in his garden! **BY GRAHAM MORRISON**

Pixel editor

## LibreSprite

Modern retro gaming, along with its associated aesthetic, is hugely popular in the video game world. From pixelated platformers to isometric role playing games, they're often some of the best new games you can play in any year. There are some AAA examples, such as the Ori games from Microsoft Studios, but the majority of retro-themed games are built with small teams and modest budgets. It's perfectly feasible for a developer and a designer to create an entire title

between themselves, such as with the wonderful Fez or Cave Story. In such cases, open source software helps to keep costs down and functionality high. And in the realm of modern retro gaming, one of the best pieces of software is called Aseprite.

Back in the olden days of 1980s gaming, the physical computer or games console would contain special hardware for handling a small group of pixels in unison. These were known as sprites, and they could typically move around and scale without taxing the 1.023

MHz CPU, or without the developer needing to worry about overlap and redrawing the background. As a result, they were often used to move and animate the main character and enemies, as well as interactive parts of the background. The hardware limitations no longer exist, but the sprite development and gameplay mechanic are still a big part of game design, leaving artists with the tricky problem of drawing meaningful characters in a 32x32 grid in the modern era without getting hold of an Amiga.

Aseprite is one of the best sprite editors built to solve this problem; a brilliant retro design tool that allows artists to draw in huge squares with a fixed palette and animate from one frame to the next. Aseprite was once open source, but moved to a proprietary license in 2016 and remains successful. Vitally, however, Aseprite was forked from its last open source commit, resulting in the creation of the LibreSprite project, which also continues to this day. It features a very similar retro-themed user interface that focuses on a tabbed canvas with an initially empty

pixel matrix. On the left is the palette with a color selector, while a floating window shows a 1:1 view of your artwork. Drawing is as simple as click and drag across the blank canvas. Layers can also be added, enabled, and disabled from a lower panel, as can frames when creating an animation.

Animations are built by making slight changes in each frame, and there's an onion skin option to show you previous and next frames while you're drawing. There's a tiles texture mode to help with repeating patterns, such as grass or water, as well as with shading, filled contours, and polygons. When finished, your work can be saved as industry standard PNG, ASE, and even GIF files. It works in a very similar way to the ancient Deluxe Paint 2, running on the Amiga, albeit with a larger color palette and smaller default image size. Also just like Deluxe Paint, it's a lot of fun to play with regardless of whether you have any drawing talent or not.

**Project Website**  
<https://github.com/LibreSprite>



- 1. Palettes:** Restricted color sets can be loaded, edited, and used for drawing.
- 2. Tabs:** Work on more than one sprite at a time.
- 3. Tools:** Rotate, fill, select colors, and cut and paste from the tools palette.
- 4. Pixel perfect:** Select this option to remove the artifacts you usually see when drawing lines.
- 5. Preview:** See how your designs look at native screen size.
- 6. Frames:** Animate your sprite across as many frames as you need, complete with onion skin drawing.
- 7. Layers:** Add layers with opacity and blended color modes.

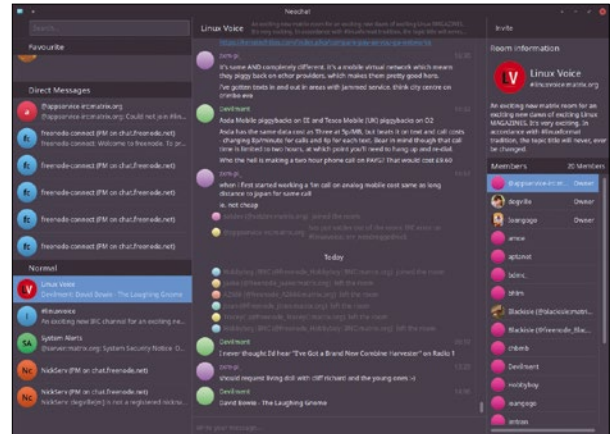
## Chat client

# NeoChat

Recent months have seen huge demand for alternative messaging platforms. From Mozilla switching to Matrix, to Facebook's WhatsApp privacy settings update pushing users to Telegram and Signal. It's the perfect time for open source platforms to capitalize on renewed interest in decentralized messaging and to help solve the biggest problem they've always had – the chicken and egg problem of adoption. But this lack of adoption isn't always because people don't know the alternatives exist. It can also be because the alternatives don't look too good. Signal is a great example because, while it's undoubtedly one of the best open source messaging solutions, switching to it is a tough ask for the typical

WhatsApp user used to seamless device and media integration. Matrix is another platform that can be hard to get your head around without a beautiful, clean client to make it worth the effort. But NeoChat might finally make the effort worthwhile.

NeoChat is a messaging client designed specifically to access the federated Matrix messaging network, and the best thing about it is its elegant interface. After logging in with your Matrix account, you soon forget you're accessing a disparate conglomerate of servers and contacts, because the experience feels just as good as some proprietary services. You can search for groups/rooms you find interesting, thread conversations, and upload media. There's even an integrated image editor for making



**NeoChat is a beautiful portal to the Matrix network and will even work on a smartphone running Plasma Mobile.**

small changes before you share a file, and room management functions help you remove problematic users from your own groups. The whole application is encased within a scalable UI that will stretch to include three columns on a wide screen, including a sidebar showing group members, and diminish to a single column chat view on narrow screens. Remarkably, this even includes smartphones if you're lucky enough to have one running Plasma Mobile.

**Project Website**

<https://github.com/KDE/neochat>

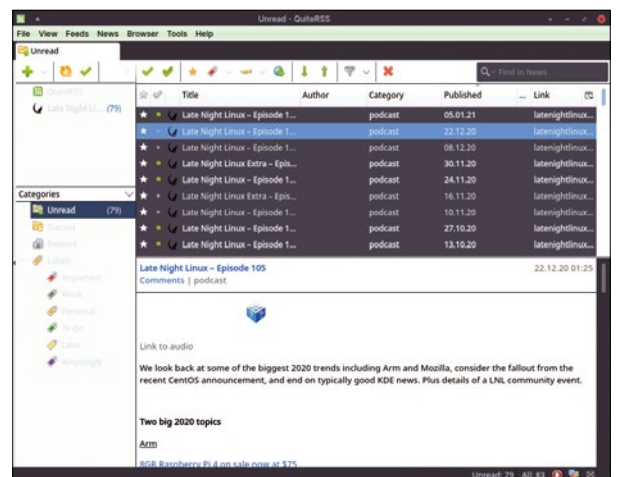
## RSS reader

# QuiteRSS

With lots of people currently considering the kind of digital footprint they're leaving on the Internet and what kind of services they're happy to share their personal data with, there's never been a better time to reinvestigate RSS. RSS is an ancient technology: A simple XML file that updates when new posts are made to a website or podcast. Timestamps enable clients to scan for changes whenever they choose, and brief snippets of whatever a post is about allow users to decide whether or not to visit or listen based on an update. There are no cookies and few ways for marketing agencies to subvert the one-way protocol to retrieve unsolicited amounts of data. All the user needs is a client. However, because RSS has

become increasingly rare on the web, such clients are becoming rare. Which is why QuiteRSS is such a pleasant surprise.

The "quite" in QuiteRSS represents the idea that the application is "quite fast." And it is, thanks to a cut down Qt-based UI and sparse design. It's an application that can feel a little out of date, but it's still being maintained and has some unique features that make it a great choice for beginners to the format. It will automatically paste a compatible RSS URL into the new subscription requester, for instance, and there's a huge amount of control over which stories you want to see and how they're displayed. Right click a feed and choose *Properties*, for instance, and you can set specific data limits for data retrieval,



**The category list is automatically populated with posts from various RSS feeds according to their categories.**

whether a feed gets its own tab, which columns are shown, and what icon to use. This allows you to use the same application for a huge variety of sources, relying on the category filters and tabs to automatically make best sense of whatever glut of information you're subscribed to.

**Project Website**

<https://github.com/QuiteRSS/quiterss>

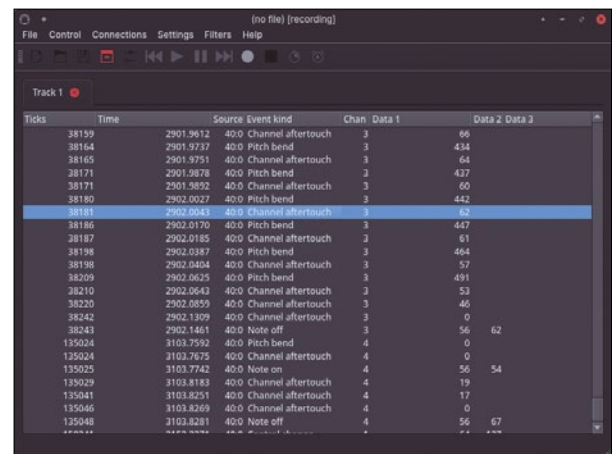
## MIDI monitor

# Drumstick MIDI Monitor

The MIDI protocol, commonly used to connect synthesizers, drum machines, and audio hardware to each other and a computer, is now 40 years old and still used in almost every artifact of music technology. This is remarkable when you consider how much technology has changed over that time: 1200 baud modems to broadband Internet, FM radio to 5G mobile data, and analog synthesizers to digital synthesizers and back to analog. Also, computers became smartphones and tablets, and MIDI became encased within USB. But MIDI still carries the same note, controller, volume, and pitch information that it did in 1985, and even the recently ratified MIDI 2.0 specification maintains this long term compatibility. This is great

for compatibility, because it means your Roland Jupiter-6 from 1982 still works with your USB MIDI interface, but it's not so great for legibility. MIDI has no namespaces, no useful YAML or JSON formatting to let you focus only on the parts that interest you, and no easy way to debug connections and see exactly what is going where. This is where the Drumstick MIDI Monitor can help.

Drumstick MIDI Monitor feels modern, thanks to its Qt UI, and connects to whatever virtual or physical MIDI interfaces you have connected via ALSA. On most distributions, this means you won't need to do any configuration. Anything connected will simply show up. When you start generating data, such as pressing a key or moving a slider, the application will



Track 1	Time	Source	Event kind	Chan	Data 1	Data 2	Data 3
38159	2901.9612	40:0	Channel aftertouch	3	66		
38164	2901.9737	40:0	Pitch bend	3	434		
38165	2901.9751	40:0	Channel aftertouch	3	64		
38171	2901.9678	40:0	Pitch bend	3	437		
38171	2901.9892	40:0	Channel aftertouch	3	60		
38180	2902.0027	40:0	Pitch bend	3	442		
38191	2902.0044	40:0	Channel aftertouch	3	64		
38196	2902.0170	40:0	Pitch bend	3	447		
38197	2902.0185	40:0	Channel aftertouch	3	61		
38198	2902.0387	40:0	Pitch bend	3	464		
38198	2902.0404	40:0	Channel aftertouch	3	57		
38209	2902.0625	40:0	Pitch bend	3	491		
38210	2902.0643	40:0	Channel aftertouch	3	53		
38220	2902.0859	40:0	Channel aftertouch	3	46		
38242	2902.1309	40:0	Channel aftertouch	3	0		
38243	2902.1461	40:0	Note off	3	56	62	
135024	3103.7552	40:0	Pitch bend	4	0		
135024	3103.7675	40:0	Channel aftertouch	4	0		
135025	3103.7742	40:0	Note on	4	56	54	
135029	3103.8183	40:0	Channel aftertouch	4	19		
135041	3103.8251	40:0	Channel aftertouch	4	17		
135046	3103.8269	40:0	Channel aftertouch	4	0		
135048	3103.8281	40:0	Note off	4	56	67	

Even system exclusive data can be captured with Drumstick, allowing you to archive and restore patches from esoteric old MIDI hardware.

list each and every message, complete with its contents. Note data will show in the on and off velocity, for example, while control values will show both the least and most significant bit of any data. The best feature, though, enables you to record the stream of MIDI data and either save it to a text file for analysis, or a MIDI file for playback – whether from the application directly or your favorite piece of music software. If you do anything with MIDI, this and the virtual MIDI keyboard by the same developer are essential tools.

## Project Website

<https://kmidimon.sourceforge.io>

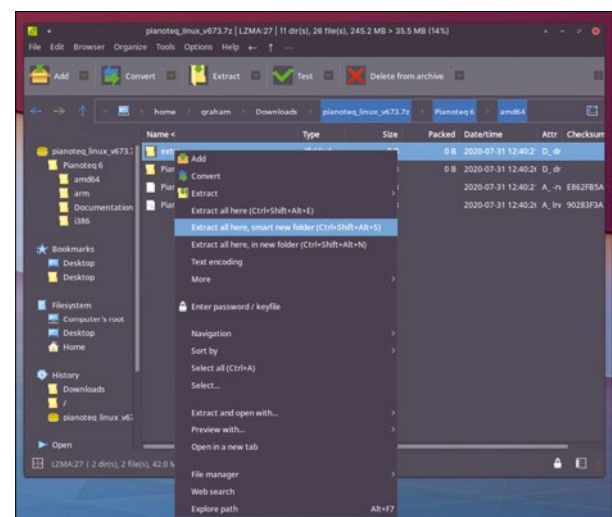
## Archiver

# PeaZip

There are so many different archiving tools for Linux including the venerable tar, gzip, and xz, and the plucky upstart, zip. Each typically has its own syntax for compressing files and folders into a single file, and conversely, its own syntax for decompressing that single file into one or more of its constituent original parts. Some even separate these functions into separate commands entirely. Of course, it's common to find graphical tools such as Xarchiver that encapsulate all these functions into a single executable. What isn't so common is finding a tool that does this across operating systems like PeaZip promises. PeaZip is a graphical desktop archive tool that can extract the contents of over 200 archive types, including all of

the above, and many more, on both Linux and Windows.

The Linux version is built using the GTK2 engine, and it's functionally identical to the Microsoft Windows version, which is itself compatible with Microsoft Windows 10/8/7/Vista, XP, Wine, and even ReactOS. The application functions much like a file explorer, with filesystem navigation in the pane on the left and files on the right. Double-clicking on any supported archive will present its contents after a few moments delay, where you can continue to navigate through the archive as you might a directory. Single files and directories can be extracted either manually or with drag and drop. There are some nicely integrated functions, such as a web search, password decryption, and



PeaZip is ideal if you need to support nontechnical people remotely across both Windows and Linux.

advanced filters, and the breadcrumb path management and history views will help new users. But the best thing about the application is still that it works the same across platforms, which is especially important on Windows where the operating system is littered with commercial proprietary equivalents offering fewer features.

## Project Website

<https://peazip.github.io/>



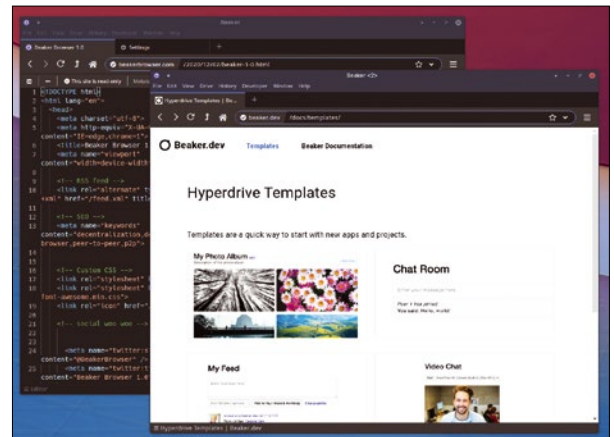
## Peer-to-peer web browser

# Beaker

There's no denying that the World Wide Web is a wonderful thing. Anyone can set up their own website and make it available online to anyone else. However, there's no doubt that the process is now more convoluted, and risky, than it was in the early days where you could serve pages from `httpd` running on your own system. This is maybe why corporations like Facebook have been so effective at becoming the (closed) World Wide Web for local community groups and institutions, and it's something that this new browser could potentially help. Beaker is a web browser with a unique, and currently experimental, emphasis on peer-to-peer hosting of personal sites that could offer an alternative to a hosted community.

The application has just hit its 1.0 milestone.

When first launched, there's very little to distinguish Beaker from any other Electron app. Thanks to its embedded Chromium engine, you can immediately start browsing the HTTPS web just as you can with Chromium. The menus are a little sparse, but there is support for bookmarks, tabs, and ad-block filter lists. There's also a developer menu with a terminal, and even more mystically, a protocol called Hypercore (`hyper://`). This is where the peer-to-peer part comes in: Beaker lets you create your own web pages and turn them into something called "Drives." These can be immediately hosted to other browsers capable of using the Hypercore Protocol – no servers required. Beaker's functionality



Beaker is a Chromium-based web browser that also allows you to create and host your own websites with other peers across the Hypercore Protocol.

is already mature, including the editor and testing tools, as well as the API for integrating the publishing mechanism to your own workflow, good documentation, and a decent range of templates for getting started. The drive component lets you only share a link with specific people, making it perfect for personal photo albums, for example. It's an interesting idea that could easily develop into an alternative web platform if just a few more people were using it.

## Project Website

<https://beakerbrowser.com/>

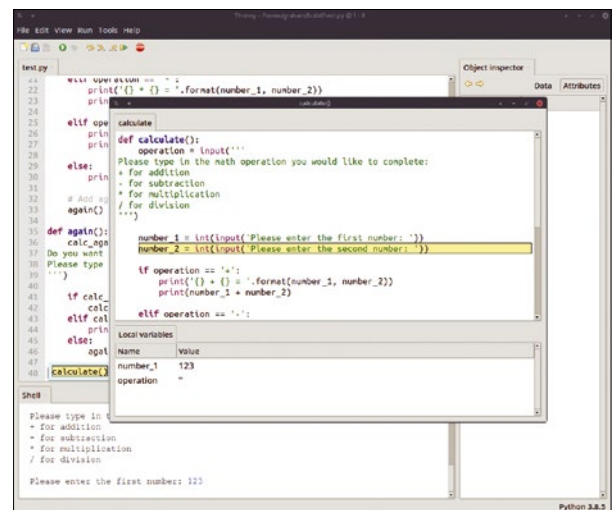
## Python IDE

# Thonny

We often look at IDEs for programming languages on these pages. But the ones we look at are usually most beneficial to experienced programmers. Thonny is the opposite. It's a small, low-resource, and self-contained IDE that helps immensely when learning Python. There's even a mode designed specifically for the Raspberry Pi desktop. But the best thing about it is that it bundles Python 3.x with the main application. That means the beginner doesn't have to struggle with installing the right version of Python and making sure it takes precedence in their path before they've even typed a line of code. And when you're ready to write code, the user interface is initially as simple as it can be. The main

panel taken by the text editor, an object inspector on the right, and an interactive interpreter along the bottom are all empty.

To get started, start writing code in the editor. There's easy to see syntax highlighting, which will even flag simple errors, and simple code completion when you press the industry standard `Ctrl` and `space` keys, although we did notice it didn't offer `elif` for some reason. With the integrated debugger, you can also watch how your variables change through the running of your code, and the right panel can be used to introspect functions and objects during execution. This will even open a new window showing only the code for that function so that you can easily see its namespace and scope. It's much easier to use than Python on



Thonny is a simple Python IDE that does a great job of making Python programming easier to manage.

the command line, and it doesn't require any specific onboarding of its own, which is probably why it's so popular on the Raspberry Pi. If you're looking for an environment to use when teaching people how to code in Python, Thonny is a great place to start.

## Project Website

<https://thonny.org/>

## Sample player

## Giada

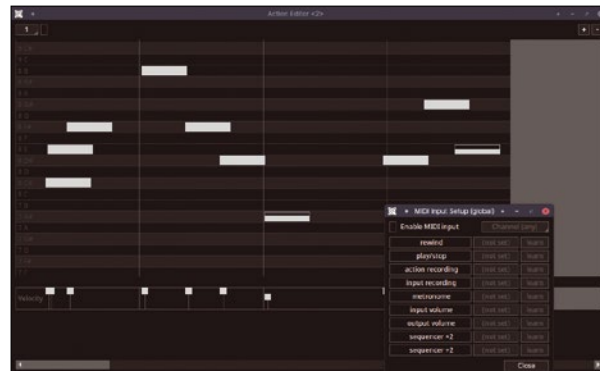
When it comes to music and audio software, we have several complex and comprehensive applications for sound generation and audio production, but there isn't much choice when your requirements are simple. One such simple task is triggering specific audio recordings (samples) at specific times. You could press play at the right time in Audacity, or you could set up a sequence of notes in an application like Ardour or Rosegarden. There aren't, however, many tools designed specifically to do this, although there are sampler plugins that get close. This is almost exactly the use case that hardware samplers were designed to address, and it's also the use case that the recently rebooted Giada has been designed to address.

Giada is a powerful audio file player that makes playing audio files and samples easy. At its simplest, you drag an audio file from your favorite file manager into one of the empty columns in the GUI, and almost instantly, an entry appears with options for

play, record, mute, and solo, alongside buttons for loop configuration and effects. Click play, and it plays. It's just as quick to drop in an entire directory of recordings, and you're free to drop them into any of the empty columns you see in a new project. Click play next to any of them, and they'll play. Click on more than one at a time, and they'll play simultaneously. Each recording is also accompanied by a status box that shows the playback position, and you can tweak the sound with the volume dial or even add live effects with the FX button.

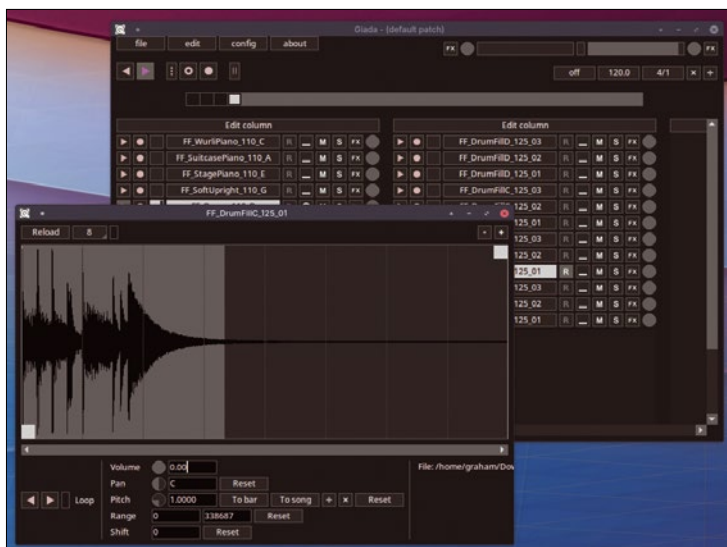
Giada is a powerful audio file player that makes playing audio files and samples easy.

For a podcast or live theater production, this is all the power you need. Drag and drop a selection of files, save them as a project, and trigger them when you need them. But Giada expands on this to take the whole idea of audio playback further.



Even with no BPM data encoded in the audio, Giada can play audio loops in time and at the same pitch automatically.

For example, there are various loop and playback modes for each audio file that can repeat drum or melodic loops as needed, while synchronized to an overall tempo and even transparently resampled or pitch adjusted when the tempo strays from that of the original recording. Even more cleverly, from a right-click menu on the audio file, you can configure it to be triggered by a specific key on your keyboard or MIDI note, and the audio will start either immediately, or at the beginning of a bar if you've configured it to do so. This is where Giada transforms from a handy audio file player into a music composition tool, because you can then sequence when audio files, samples, and loops are played using the record button. This is exactly what aspiring DJs need as they play full tracks alongside tempo-synced loops and effects. You can even edit the audio files in place with an excellent embedded sample editor; if you want to be more ambitious with sequencing, the action editor can be used to create an entire song, complete with piano roll, velocity, and volume automation for programming software and hardware synthesizers. It's a brilliant application that scales from a simple player to an almost fully fledged Ableton-like DAW, without bringing your system to its knees.



Giada is incredibly fast and efficient, thanks to its UI and minimalist design, but it's also deceptively powerful.

## Project Website

<https://github.com/monocasual/giada>

## Evolution simulator

# Thrive

**G**odot, the open source games development engine, has been getting lots of deserved support and coverage in the last year. It's doing a brilliant job helping people get into games development without getting themselves locked into a proprietary ecosystem, much like Blender is doing for 3D designers. Thrive is a game built with Godot that proves this progress is having a hugely beneficial effect on games development, not just in helping developers create games at low cost, but in helping games to be developed that might not otherwise be easily created. Thrive is a game about the evolution of life on an alien planet not too dissimilar to Earth, and some gamers think it's an idea not too dissimilar to Spore – a cult game

published by Electronic Arts in 2008. The basic principle is that you help to evolve a single-celled organism into a race that can conquer the stars, in much the same way Civilization takes you from prehistory to the space age.

There are seven stages planned for the game, each corresponding to an evolutionary stage: Microbe, Multicellular, Aware, Awakening, Society, Industrial, and Space. As the game is under heavy development, only the earliest Microbe stage of the game is currently playable, but it does include tutorial sections, beautiful graphics, and already compelling gameplay. You start as a cell, which you navigate through the primordial alien soup. You absorb compounds as you cross them, trying to fill various



In Thrive, you take your life form from microbe to the stars – or you will be able to when development reaches that stage.

bars necessary for life. You can then upgrade your cell to become a more complex organism. It's a great way to learn about how these basic building blocks of life could combine, especially when you start to see the environment and accompanying species change as a result of your progress. And the same is true of the project itself, where you can even hopefully watch the game develop from these proto stages into a fully fledged title.

**Project Website**

<https://revolutionarygamesstudio.com/>

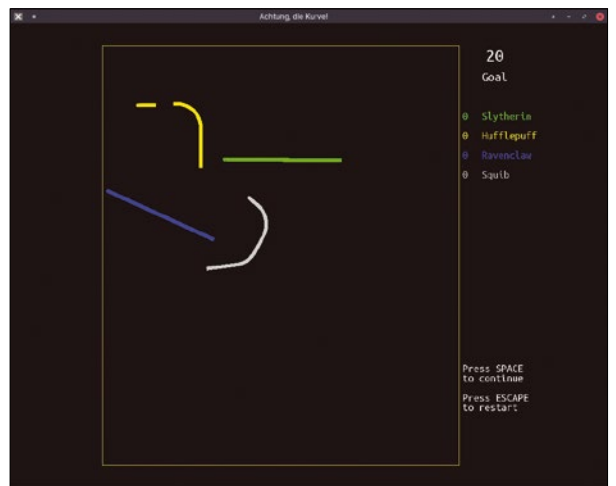
## Multiplayer game

# Kurve

**T**here are many brilliantly designed games that are very simple yet have great playability, and Kurve is one of the best examples. It's a modern remake of an old freeware DOS game from the mid-90s called "Achtung, die Kurve!," although many older players will recognize the gameplay as being similar to the light cycle level in the arcade game Tron, or perhaps even Snake on the Nokia 6110. In Kurve, you control the direction of a pixel that's drawing a continuous line by using only two keys, one for left and one for right. If at any point you cross a line or hit the edge of the gameplay area, you lose. But the brilliant element that makes it all so much fun is that six people can play at the same time – not online, or with different controllers,

but all via one keyboard in front of your computer. It means you can get up to six of your friends together in close proximity, each with fingers delicately poised on specific pairs of keys on your keyboard, and each with the ability to zoom about across the screen creating a line in whatever color that person's pixel avatar has been granted.

The ensuing game is predictably chaotic but a lot of fun. Every player starts in a random location and needs to wait for the other players to press their keys to signal their ready state. Their dots are then launched across the screen, which quickly turns into cartoon-style spaghetti. Each player's drawing will stop at random points to create gaps that give skilled players a way through,



**Kurve is one of the best things to play on your computer when you can finally get a few friends around it in person.**

and that means games last longer than in Tron, especially with fewer players. But it always ends the same way, as each player eventually succumbs to the mess of lines on the screen. All that's then left to do is enjoy your ranked moment on the leaderboard before pressing space to play again.

**Project Website**

<https://github.com/h-valdes/kurve>



# Search more efficiently with ugrep

## Tracked Down

Searching for text in files or data streams is a common and important function. Ugrep tackles this task quickly, efficiently, and even interactively if needed.

BY KARSTEN GÜNTHER

**G**rep is one of the oldest Unix commands. The abbreviation “grep” stands for *Global/Regular Expression/Print* or *Global search for a Regular Expression and Print out matched lines*. It picks up on the syntax of the original Unix editor, QED, which used `g/re/p` to search for patterns in text files. In addition to fixed search terms, it can also search for patterns with wildcard characters. The GNU variant of grep is normally installed on Linux. It extends the features of the original grep in some places, for example, allowing recursive searching in directories.

Another variant of grep, `agrep` (approximate grep) [1], extends text searching to include fuzzy searches. It also finds near misses as long as the differences are below a specified threshold, known as the word distance. This is calculated from the necessary permutations, deletions, and additions of letters that convert the search pattern into the actual data.

In addition, there are some variants of grep that also find search patterns in certain archive types, such as ZIP files. These programs are relatively slow, since they first need to unpack the archive. However, all grep variants used on Linux can also read data from pipes via the standard input channel and write the results to the standard output channel for searching in archives (Listing 1).

### ugrep

Ugrep can do all of this and more without explicitly unpacking the data streams. In addition, the program is known for its exceptionally fast processing speed. To speed up the search, it uses multiple threads if necessary.

On Debian and Arch Linux, setting up ugrep is easy. Debian has the tool in its repositories; with

Arch Linux, you can use the AUR. For all other distributions, you will have to install ugrep from the source code [2]. The commands required for this are shown in Listing 2.

Ugrep is programmed in C++, has been around for several years, and is available not only on Linux, but also on other operating systems. Search patterns specified as regular expressions can span consecutive lines, a thing that many other grep variants cannot do. By default, ugrep assumes Unicode as the encoding for the search data.

Ugrep supports archive types including CPIO, JAR, PAX, TAR, and ZIP, compressed with all common methods (BZIP, GZ, LZ, and XZ). In addition, you can use filters to prepare data in special formats in advance. For example, PDF documents can be converted to text with a filter, before ugrep performs the search.

Like all grep variants, the program is largely controlled by options. For most options, as usual, there is a short form (`-<0>`) and a long form (`--<0ption>`). Table 1 summarizes the most important options.

Besides all of this, the developer suggests a number of alias constructs for the `.bashrc` to ensure compatibility with GNU grep, for example (see Table 2). Some of these short forms rely on the `ug` command variant. In this form, ugrep reads in a configuration file (by default `$HOME/.ugrep`) which can contain special settings. This means that important presets can be applied implicitly without having to specify them at the command line every time.

Ugrep supports several search pattern variants, which you enable through appropriate options

### Listing 1: Archive Search

```
$ zcat archive.gz | grep <pattern>
```

### Listing 2: Installing ugrep

```
$ git clone https://github.com/Genivia/ugrep
$ cd ugrep && ./build.sh
$ sudo make install
```

(see the “Patterns” box). Besides simple and extended regular expressions like GNU grep, ugrep also supports Perl regexes and word patterns. In addition to these default patterns, which always define positive patterns, ugrep can also use negative patterns (exclusion patterns). They let you,

for example, ignore matches if they occur in comments. Files whose names match a certain pattern can also be excluded from the search. The `--not` option has a special effect: All patterns to the right of it are used by ugrep as exclusion patterns.

**Table 1: Important Options**

<code>-a</code>	Interpret data as text
<code>-c</code>	Match count
<code>-e &lt;pattern&gt;</code>	Search for specified pattern (can specify multiple patterns)
<code>-E</code>	Interpret search patterns as extended regular expressions (default)
<code>--encoding=&lt;encoding&gt;</code>	Set encoding for data
<code>-f &lt;file&gt;</code>	Load search pattern from specified file
<code>-F</code>	Interpret search pattern as string (special characters are considered as text)
<code>--filter=&lt;filter&gt;</code>	Pre-filter based on specified filter criteria
<code>-G</code>	Interpret search patterns as simple regular expressions
<code>-i</code>	Ignore case in pattern
<code>-N &lt;pattern&gt;</code>	Define negative search pattern
<code>--not</code>	Interpret all of the following search patterns as exclusion patterns
<code>-O &lt;extension&gt;</code>	Edit only files with the specified extension
<code>-P</code>	Interpret search patterns as Perl expressions
<code>--pager=&lt;pager&gt;</code>	Set pager for terminal output
<code>-Q[&lt;delay&gt;]</code>	Incremental search with optional delay
<code>-R</code>	Recursive search
<code>-w</code>	Word search
<code>-X</code>	Output in hexadecimal form
<code>-z</code>	Unpack compressed data streams in advance
<code>-Z&lt;Criteria&gt;</code>	Fuzzy search with set criteria for allowed deletions, insertions, or substitutions

**Table 2: Suggested Alias Constructs**

Alias	Function
<code>alias uq = 'ug -Q'</code>	Interactive, incremental search
<code>alias ux = 'ug -UX'</code>	Binary search
<code>alias uz = 'ug -z'</code>	Search in (compressed) archives
<code>alias ugit = 'ug -R --ignore-files'</code>	Grep for Git
<b>Compatibility with classic variants</b>	
<code>alias grep = 'ugrep -G'</code>	Search with simple regular expressions
<code>alias egrep = 'ugrep -E'</code>	Search with extended regular expressions
<code>alias fgrep = 'ugrep -F'</code>	Search without regular expressions
<code>alias pgrep = 'ugrep -P'</code>	Search with Perl regular expressions
<b>Search in compressed data</b>	
<code>alias zgrep = 'ugrep -zG'</code>	Archive search with simple regular expressions
<code>alias zegrep = 'ugrep -zE'</code>	Archive search with extended regular expressions
<code>alias zfgrep = 'ugrep -zF'</code>	Archive search for strings
<code>alias zpgrep = 'ugrep -zP'</code>	Archive search with Perl regular expressions

## Patterns

The term “pattern” usually appears in multiple contexts with different meanings in search programs like `ugrep`. Patterns in file names determine which files the program processes. The file content patterns are the actual search patterns for which it searches the processed files. With `ugrep`, these may also be across lines. `Ugrep` and some other search programs also support negative patterns. They are used to exclude files or not to display corresponding matches. In fact, `ugrep` takes this procedure quite far: In the program’s documentation, there is a separate section, *Search this but not that with -v, -e, -N, --not, -f, -L, -w, -x*, that deals with the finer points of this subject.

## Extensions

In many places `ugrep` extends the other, classic program versions. The new features for patterns in file names (“globbing”) are particularly interesting. For example, `**/` stands for any number – even zero – directories. At the end of a path definition, `/**` stands for any number of files. The special case `\\?` addresses zero characters or one. In the man page, the globbing section summarizes these features and also gives numerous examples.

Special environment variables let you additionally control the behavior of `ugrep`. `$GREP_PATH` simplifies access to so-called pattern files (i.e., files that define search patterns); the `-f` option enables this feature. Patterns in external files are a good way to keep complex search patterns permanently.

Some options, including `-Q`, can use an external editor that the key combination `Ctrl+Y` starts. If the `$GREP_EDIT` environment variable is set, `ugrep` uses the editor defined there; otherwise the one defined in `$EDITOR` is used.

The `$GREP_COLOR` and `$GREP_COLORS` environment variables let you specify when and how `ugrep` color highlights matches when using the `--color` option. The `GREP_COLORS` section in the man page describes this in more detail.

But the really outstanding extensions in `ugrep` are the incremental search feature and the user interface.

## User Interface

Grep programs are usually used interactively in command lines, scripts, or pipes; in many cases the results then act as input for further commands. This also works without any restrictions in `ugrep`. In addition, the developer has also paid great attention to extended interactive usability. For example, incremental searching is currently

an absolutely unique selling point of `ugrep`. The user interface used for this was modeled on editors such as Emacs and is normally reserved for GUI programs.

With this type of search, each additional letter specified further refines the search and reduces the number of matches. All lines that match the previous entries are then displayed. For this form of search, `ugrep` provides a special interface that you enable using the `-Q` option. As an argument of `-Q`, you can specify a small delay that `ugrep` waits for before evaluating the input.

The `Q>` prompt now appears in the upper left corner of the terminal. Everything you type is interpreted by `ugrep` as a search pattern; each additional keystroke refines the search. Typos can be corrected with the backspace key. In the example from Figure 1, we called `ugrep` with the `-ZQ` (fuzzy, interactive) options and searched for “alles” (“everything” in German). Due to the fuzzy search, `ugrep` also finds “alpes”, “alls”, “ales”, and so on.

This feature is so powerful that `ugrep` in this mode can sometimes even replace a pager for displaying output. For example, `man ugrep | ugrep -Q` displays the man page of `ugrep` and lets you define exactly which search term it should display. The output can also be shifted vertically with the arrow keys; `Esc` ends the mode again.

On top of that, this option can be combined with others. In case you need more than the ability to see just the line with the match, you can add two context lines before and after the match to the output using `-C2`. In this form, `ugrep` is extremely useful as an alias (`alias q2='ug -C2 -G '`), shell function, or script.

The ability to search archives is a similar case. Many modern documents are in complex formats

```

Z>alles
alpes          463/tcp
alpes          463/udp
h323hostcalls  1300/tcp
h323hostcalls  1300/udp
mt-scaleserver 2305/tcp
mt-scaleserver 2305/udp
call-sig-trans 2517/tcp
call-sig-trans 2517/udp
sonuscallsig   2569/tcp
sonuscallsig   2569/udp
allstorcn      2901/tcp
allstorcn      2901/udp
caller9        2906/tcp
caller9        2906/udp
stonefalls     2986/tcp
stonefalls     2986/udp
ewinstaller    4091/tcp
ewinstaller    4091/udp
parallel       4989/tcp
parallel       4989/udp
alesquery      5074/tcp
alesquery      5074/udp
h323callsigalt 11720/tcp
h323callsigalt 11720/udp
(END)

```

**Figure 1:** Ugrep enables interactive, fuzzy, and incremental searches.



like EPUB, ODF, etc. There, the options usually only act on metadata in the document containers – often ZIP archives. To search in the actual contents, you have to unpack these archives, which is done either by a filter (more on that later) or the `-z` option, often combined with `-r` for recursive.

Ugrep supports fuzzy searching with the `-Z` option, which may be followed by a number appended directly without spaces. The latter determines the degree of fuzziness, that is, the permissible number of errors (omitted, added, swapped characters). The default is 1. Larger values quickly lead to many additional hits, but this sometimes makes the results unusable.

However, the type of allowable errors can be specified: With a prefix of `+` or `-`, the specification refers only to additions or omissions, respectively. The tilde (`~`) groups several errors. `-Z~-2` means that up to two omissions or swaps are allowed. The `--sort=best` option sorts the output so that the files with the best matches appear first.

Ugrep uses some function keys for special tasks in interactive mode. For example, `F1` activates the online help (Figure 2) where ugrep displays the current keyboard shortcuts. You can enable additional options by calling them in this mode. For example, after pressing `F1`, the key combination `Alt+Shift+Z` activates fuzzy searching.



Figure 2: The ugrep help function conveniently comes with a built-in configuration mode.

## Shop the Shop

shop.linuxnewmedia.com

### Missed an issue?

You're in luck.

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

shop.linuxnewmedia.com

**GET IT NOW!**

SAVE TIME ON DELIVERY WITH OUR ALTERNATIVE PDF EDITIONS



Invoked with the `--save-config` option, the program creates the `$HOME/.ugrep` configuration file. If necessary, you can create another file using `--save-config=/path/<file>`. Similarly, `--config` reads configuration files. Calling `ugrep` as `ug` automatically parses the configuration.

Since configuration files are a powerful means of controlling `ugrep`, there is also the shorthand `---<file>` for loading. You can create configuration files with certain preset options with the following command:

```
$ ugrep -<option> [...] --save-config
```

The configuration files are well commented and can be easily customized with a text editor if needed.

## Filters

`Ugrep` tries to determine the type of an examined file based on the data it contains, the file name extension, and the signature (the “magic byte”). In this way, the search can be specially prepared for certain file types (i.e., filtered).

Here the filter extracts the text components from the data streams. These filters execute a command, a script, or a specific function, with pipes if necessary. They are prepended to the search process via the `--filter=<Filter>` or `--filter-magic-label=<Label>:<MagicByte>` option.

In the form `--filter=<filter>`, the `<filter>` consists of an expression of the form `<Ext>:<command line>`. `<Ext>` is a comma-separated list of file name extensions for which you want the filter to apply, such as `.doc,.docx,.xls`. The `*` character is a special case that acts on all files, especially those for which there are no other filters.

The `<command>` line must be constructed to read input via the standard input channel and write the results to the standard output channel. Typical commands include `cat` (pass everything) and `head` (pass the first lines of text), but tools like `exiftool` (extract and pass metadata) or `pdftotext` (extract text from PDFs) can also be included this way. Some commands, like `pdftotext`, require options to work correctly – in this case `pdftotext % -`. You then need to quote spaces in the command lines to protect them:

```
--filter='pdf:pdftotext % -'
```

The `--filter-magic-label=<Label>:<Magic>` option lets you extend the filtering mechanism to data streams that `ugrep` then classifies by reference to the magic byte. Details can be found in the man page.

Multiple filters can be specified as comma-separated lists. A combined definition for PDF and Office documents might look like the one shown in Listing 3.

## Conclusions

`Ugrep` belongs on every computer. It replaces and complements the standard commands quite excellently, and anyone who has to deal with text searches should familiarize themselves with it. The incremental search alone is so useful that it more than justifies the minimal training time. ■■■

## Info

[1] `agrep`: <https://linux.die.net/man/1/agrep>

[2] `ugrep`: <https://github.com/Genivia/ugrep>

## Listing 3: Combined Filter Definition

```
--filter="pdf:pdftotext % -,odt,doc,docx,rtf,xls,xlsx,ppt,pptx:soffice --headless --cat %"
```

■■■

# 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.



**#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 wade 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



**#240/November 2020**

## It's Alive

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

**On the DVD:** Linux Magazine 20th Anniversary Archive DVD



**#239/October 2020**

## Build an IRC Bot

IRC bots do the essential work of coordinating and forwarding chat messages on the Internet. This month we show you how to build your own custom bot – and we give you an inside look at how to work directly with IRC.

**On the DVD:** Debian 10.5 and Devuan 3.0



# 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.

## DrupalCon North America 2021

**Date:** April 12-16, 2021

**Location:** Virtual Event

**Website:** <https://events.drupal.org/drupalcon2021>

DrupalCon North America is where the people who make digital experiences possible come together to innovate. This week-long event is all virtual and all inclusive, with opportunities to learn, connect, and build with Drupal. Participate wherever you are, whether you're experienced or new to Drupal.

## DevOpsCon Hybrid Edition

**Date:** April 21-23, 2021

**Location:** Virtual Event

**Website:** <https://devopscon.io/london/>

DevOps is fundamentally changing the IT world and sets the path for successful business transformation. Join DevOpsCon to learn about the latest tools, technologies, and methodologies for building and maintaining secure, scalable, and resilient software systems.

## Events

CloudFest 2021	March 23-25	Virtual Event	<a href="https://www.cloudfest.com/">https://www.cloudfest.com/</a>
Kubernetes Community Days	April 8-9	Amsterdam, Netherlands	<a href="https://sessionize.com/kcdams2021/">https://sessionize.com/kcdams2021/</a>
NSDI '21	April 12-14	Virtual Event	<a href="https://www.usenix.org/conference/nsdi21">https://www.usenix.org/conference/nsdi21</a>
DrupalCon North America 2021	April 12-16	Virtual Event	<a href="https://events.drupal.org/drupalcon2021">https://events.drupal.org/drupalcon2021</a>
DevOpsCon London Hybrid Edition	April 20-23	London, UK and Online	<a href="https://devopscon.io/london/">https://devopscon.io/london/</a>
KubeCon + CloudNativeCon Europe	May 5-7	Virtual Event	<a href="https://bit.ly/kube-cloudnativecon">https://bit.ly/kube-cloudnativecon</a>
Linux Storage Filesystem & MM Summit	May 12-14	Palm Springs, California	<a href="https://events.linuxfoundation.org/lfsfmm/">https://events.linuxfoundation.org/lfsfmm/</a>
LISA21	June 1-3	Anaheim, California	<a href="https://www.usenix.org/conference/lisa21">https://www.usenix.org/conference/lisa21</a>
SYSTOR 2021 Hybrid	June 14-18	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>
ISC High Performance 2021 Digital	June 24-July 2	Virtual Event	<a href="https://www.isc-hpc.com/">https://www.isc-hpc.com/</a>
Cloud Expo Europe	July 7-8	London, United Kingdom	<a href="https://www.clouDEXPOEUROPE.COM/">https://www.clouDEXPOEUROPE.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>
KVM Forum	August 2-4	Vancouver, British Columbia	<a href="https://events.linuxfoundation.org/">https://events.linuxfoundation.org/</a>
Embedded Linux Conference North America	August 4-6	Vancouver, British Columbia	<a href="https://events.linuxfoundation.org/">https://events.linuxfoundation.org/</a>
Open Source Summit North America	August 4-6	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/conference/usenixsecurity21">https://www.usenix.org/conference/usenixsecurity21</a>
Embedded Linux Conference Europe	Sept 29-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).



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, Lori White

### Cover Image

©glebstock, 123RF.com

### Advertising

Brian Osborn, [bosborn@linuxnewmedia.com](mailto:bosborn@linuxnewmedia.com)  
phone +49 89 3090 5128

### Marketing Communications

Gwen Clark, [gclark@linuxnewmedia.com](mailto:gclark@linuxnewmedia.com)  
Linux New Media USA, LLC  
2721 W 6th St, Ste D  
Lawrence, KS 66049 USA

### Publisher

Brian Osborn

### Customer Service / Subscription

For USA and Canada:  
Email: [cs@linuxpromagazine.com](mailto:cs@linuxpromagazine.com)  
Phone: 1-866-247-2802  
(Toll Free from the US and Canada)

For all other countries:  
Email: [subs@linux-magazine.com](mailto:subs@linux-magazine.com)

[www.linuxpromagazine.com](http://www.linuxpromagazine.com) – North America

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

While every care has been taken in the content of the magazine, the publishers cannot be held responsible for the accuracy of the information contained within it or any consequences arising from the use of it. The use of the disc provided with the magazine or any material provided on it is at your own risk.

Copyright and Trademarks © 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, 2721 W 6th St, Ste D, Lawrence, KS, 66049, USA. Periodicals Postage paid at Lawrence, KS and additional mailing offices. Ride-Along Enclosed. POSTMASTER: Please send address changes to Linux Pro Magazine, 2721 W 6th St, Ste D, Lawrence, KS 66049, USA.

Published monthly in Europe as Linux Magazine (ISSN 1471-5678) by: Sparkhaus Media GmbH, Ziebländstr. 1, 80799 Munich, Germany.

## Authors

Bernhard Bablok	60
Erik Bärwaldt	64
Axel Beckert	30
Zack Brown	11
Bruce Byfield	6, 22, 34, 56
Joe Casad	3, 14
Mark Crutch	69
Karsten Günther	90
Jon "maddog" Hall	70
Frank Hofmann	30
Charly Kühnast	29
Christoph Langner	26, 80
Vincent Mealing	69
Graham Morrison	84
Lee Phillips	72
Rob Reilly	46
Mike Schilli	50
Tim Schürmann	14
Tomasz Szandała	38
Jack Wallen	8
Harald Zisler	14





Issue 246 / May 2021

# Optimizing Startup

Linux startup is faster than ever, but our expectations are also rising and our full-featured systems are evermore bulky. Next month we show you how to squeeze some precious cycles and seconds from the Linux startup process.

## Approximate

UK / Europe	Apr 03
USA / Canada	Apr 30
Australia	May 31

## 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>

Photo by christian-foude on unsplash



# Hone your skills with special editions!

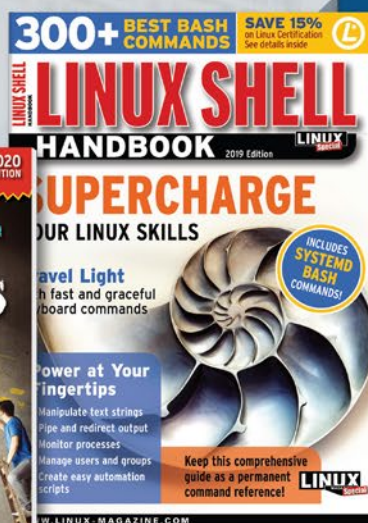
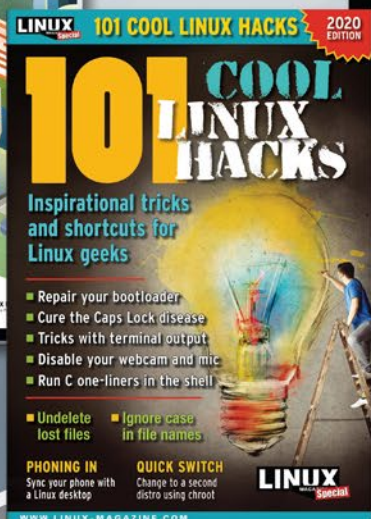
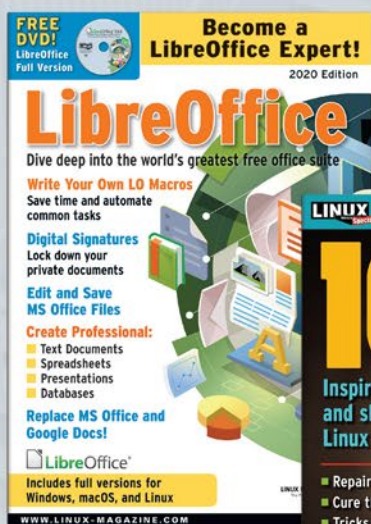
Get to know Shell, LibreOffice, Linux, and more from our Special Edition library.

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)





# 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.



Lead Image © enki, 123RF.com

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

**2020**  
**Archives**  
**Available**  
**Now!**