Miniature Photography
with a webcam and Rasp Pi

LINUX PRO MAGAZINE

ISSUE 254 – JANUARY 2022

Phone Hacks

No more vendor updates? Put a free OS on your smartphone and keep it calling

EasyNAS
Turn your old computer into a network storage device

MergerFS
Join disks, volumes, and arrays into a single file system

Lutris
Play your favorite games on Linux with this unified gaming platform

10 MORE FABULOUS FREE TOOLS!

LINUX NEW MEDIA
The Pulse of Open Source

WWW.LINUXPROMAGAZINE.COM

VALUATIONS AND VALUES

Dear Reader,

Facebook whistle-blower Frances Haugen stirred up a lot of trouble for her former company with her recent appearance on Capital Hill. A vivid picture of Facebook's internal culture came to light in her recent testimony. One of the more disturbing revelations was that the angry emoji reportedly carried five times more weight in the Facebook algorithm than the like button [1]. Apparently, all the emojis had equal weight, but it was the angry face that tended to serve as a counterpoint to the Like button, which isn't technically an emoji. The result was that stuff that makes you angry was five times (later cut to four times) more likely to end up in your newsfeed than stuff you liked. Facebook was apparently aware of this but didn't take sufficient steps to fix it, which was an optical disaster. After years of saying their goal was to connect people, these revelations left the impression that their real goal was to maximize clicks, even if it led to division.

The situation was obviously really broken, but who or what was to blame? The problem with assigning a single culprit for such a fiasco is that you imply that the only issue is the implementation: If it weren't for a process snafu or a few bad actors, everything would have been fine. In fact, the system itself is the problem, and I'm not just talking about the kooky and often irrational Internet advertising economy (which certainly is a problem). The deeper problem is the business of business, or, more specifically, the business of how new companies get off the ground.

Facebook got off the ground sometime around 2003/2004 in Mark Zuckerberg's college dorm room, but for public investors, Facebook arrived on the scene with the Initial Public Offering (IPO) on May 18, 2012. The IPO was one of the largest in history for a technology company, with a total capitalization coming in at over $104 billion. At the time, commentators were wondering how the company would ever make enough money to justify the price. As *San Francisco Chronicle* technology editor James Temple wrote just after the IPO [2], "The problem is that the smart money on Wall Street simply doesn't think the company's [Facebook's] prospects justify the $105 billion that the offering price implied. And no wonder. That values the company at 108 times 2011 earnings, requiring almost ridiculous financial growth to make sense. By way of comparison, Google trades at less than 19 times earnings."

Price to earnings (P/E) ratios are best discussed elsewhere, but suffice it to say, average P/E ratios are much lower than Facebook's was at the time. Startups often have higher P/E ratios, because the whole point is that people are investing in potential, but you also need to figure in the basic uncertainty about Facebook's revenue model. As I write this column, the electric truck company Rivian just had a successful IPO with *no* earnings so far, which give it a P/E ratio of infinity, but everyone knows what a truck is and how to sell a truck. At the time of the Facebook IPO, the world was just beginning to see what a Facebook ad was, and no one had any way to predict whether Facebook would ever sell enough to justify a $104 billion valuation. Back-of-the-envelope calculations showed that they would have to capture a very large percentage of the global Internet ad sales market to achieve the earnings necessary to justify what the stockholders paid for their shares.

When emojis arrived on the scene in 2017, Facebook was still methodically working their way back from the oblivion imposed by their shareholders' obsession. Fast forward to today, and the company actually succeeded. They really did capture a very large percentage of the global Internet ad sales market, and their P/E ratio is currently down to around 32. In terms of execution, the company should be commended for succeeding in clawing their way to stability after the ridiculous early expectations, but the point is, they didn't get there by playing nice. From the date of the initial public offering, it was clear that Facebook would have to grow at an explosive, exponential rate in order to justify their share price, and they weren't in a position to lead with benevolence.

To be fair, they are trying to fix it now – they say they have already fixed it, but it is a little hard to tell, since there is no real oversight. There are lots of reasons to be angry with Facebook, but if you want to get angry about the anger, just remember that it was our stark and soulless corporate finance system that defined their culture. Their shareholders gave them an impossible task back in 2012, and like any good gamers, they made it the object of their single-minded focus. Their rocket to the stars ran on anger because that was the cheapest and most abundant fuel, and they needed a *whole* lot of lift.

Joe Casad,
Editor in Chief

Info

[1] "Facebook Formula Gave Anger Five Times more Weight than Likes, Documents Show": *https://thehill.com/policy/technology/578548-facebook-formula-gave-anger-five-times-weight-of-likes-documents-show*

[2] "Facebook IPO Underscores Shutting Out the Masses": *https://www.sfgate.com/business/article/Facebook-IPO-underscores-shutting-out-the-masses-3575283.php*

LINUX MAGAZINE
JANUARY 2022

ON THE COVER

NEWS

COVER STORIES

REVIEW

IN-DEPTH

Eventually phone manufactures just give up on supporting old hardware. If you're not ready to abandon that hardware yourself, you might find a better alternative with LineageOS — a free Android-based system that supports more than 300 phones, including many legacy models that are no longer supported by the vendor. We also explore PostmarketOS, a community-based Linux distribution that runs on several Android devices.

IN-DEPTH

MakerSpace

LINUXVOICE

Ubuntu 21.10 and EndeavourOS 2021.08.27

Two Terrific Distros on a Double-Sided DVD!

Ubuntu 21.10
64-bit

Ubuntu 21.10 (Impish Indri) is the latest release of one of the most popular distributions on the desktop. It will be supported until July 2022, when it will be replaced by release 22.04, the next Long-Term Support (LTS) release.

Ubuntu reserves major new features for its LTS releases, so 21.10 does not introduce any radical changes. The few that it does have include support for the upcoming Intel Alder Lake-S and AMD Aldebaran chips, Microsoft Surface laptops and tablets, and the beginnings of support for the Apple M1 ARM-based system on a chip. The release also benefits by including the Linux 5.12 kernel, which was released in April 2021. However, most of the changes are upgrades. On the desktop, these upgrades include the Gnome 40 desktop environment, as well as a collection of application images for containers, plus the latest versions of GCC, OpenLDAP, and OpenJDK for developers, and the latest versions of Pulse Audio, Apache, OpenStack, and LibreOffice for other users.

Ubuntu aims the release primarily at developers, especially those building cloud applications. However, the bug fixes and security from updated applications make Impish Indri a release that users of all interests and skill levels will want to install.

EndeavourOS
64-bit

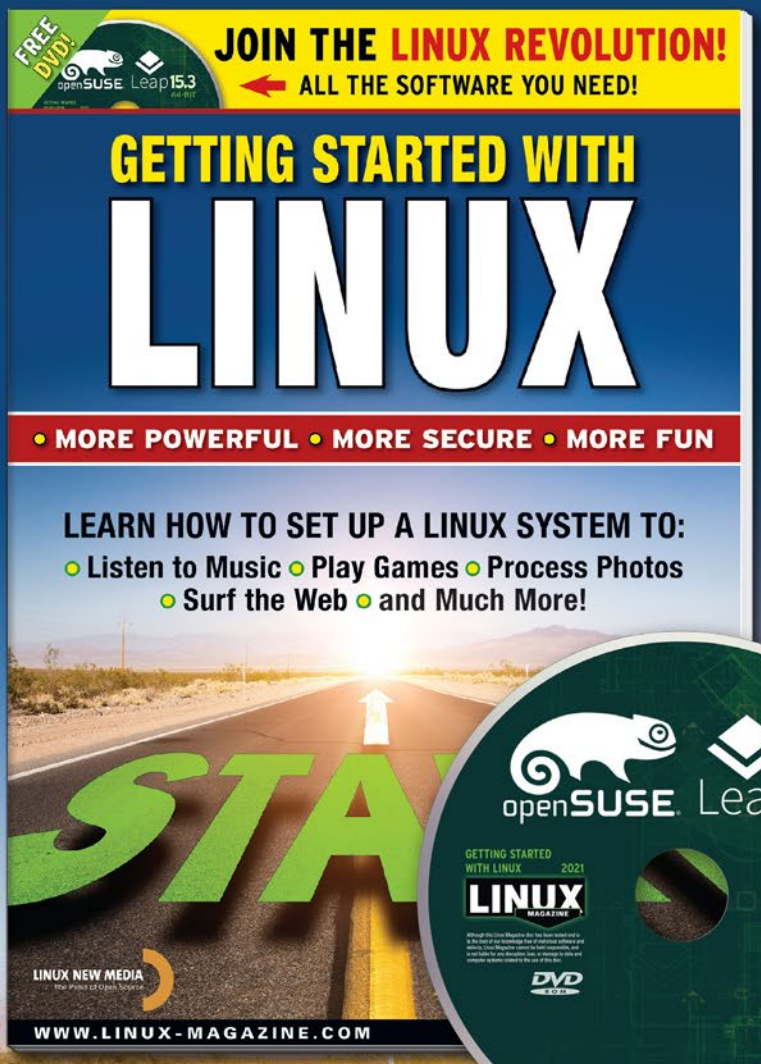
EndeavourOS is one of the major Arch Linux derivatives. Like Arch itself, Endeavour is a rolling release, with new packages being added to the repositories as soon as they are tested, instead of waiting for a new release. Its main difference from Arch is the Calamares installer, which provides an easy graphic install. In addition, Endeavour also includes a detailed Welcome App, which provides links to help new users get up and running as painlessly as possible.

Make no mistake, though: EndeavourOS is still very much tied to Arch. Although its main purpose is to make Arch more accessible, it still shares much of Arch's philosophy. It includes official and unofficial variants that support a variety of desktop environments, but EndeavourOS describes itself as "terminal-centric" with a successful install leaving new users at the command line. Similarly, like Arch, Endeavour does not include the usual curated list of applications found in most distributions. Instead, it assumes users enjoy exploring and learning and challenges them to "create your own destiny with an incredible journey." Endeavor adds, "EndeavourOS isn't an expressway to a predefined destination in computing, it is all about the journey towards your own destination and on top of that, you'll be meeting new and friendly faces during that exploration." To help on that journey, Endeavour has an extensive knowledge base of its own, as well as Arch's knowledge base.

Endeavour is intended for users who want to try Arch and will find the challenge enjoyable. If that sounds like you, then you've come to the right distribution.

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

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

NEWS

Updates on technologies, trends, and tools

THIS MONTH'S NEWS

System76 Developing a New Desktop Environment

System76 is never one to settle. Consider what they've done to the Gnome desktop environment with their COSMIC treatment. Effectively, the company has made Gnome their own. But that wasn't enough, so the developers have set out to create a brand new DE, from scratch, using Rust.

Why? Michael Murphy, Pop!_OS maintainer for System76, said, "There are things we'd like to do that we can't simply achieve through extensions in Gnome. Extensions in general feel like a hack. And what we want to do with our desktop differs from Gnome, so it's not like the option to merge pop-shell and COSMIC into Gnome Shell would be a welcome thing."

Although the new desktop will not be a fork of Gnome, the developers do plan on reusing some of the current Gnome tooling, such as Mutter, KWin, and Wlroots. On this issue, Murphy said, "We're already using gtk-rs for all of our stuff. My assumption is that it's likely to see some components in GTK for the foreseeable future." Murphy continues, "The shell itself though is lower level than a traditional desktop GUI toolkit. It'll use primitives from the window manager it builds upon. If a mature Rust GUI turns up, then it could be used in the future of course. I'd generally like to use the best tools where possible."

The System76 desktop will also be distribution-agnostic, so it won't require Pop!_OS to run. Additionally, the developers plan on sticking to (when possible) the standards set by FreeDesktop.

Don't expect this new desktop environment to appear any time soon. Because this is a brand new project, it'll take some time. Keep a close watch on the System76 blog for more updates (*https://blog.system76.com/*).

Hetzner Opens New Location in the USA

Hetzner has been well known in both Germany and Finland as a powerful cloud solution that benefits users with low latency and a price/performance ratio that doesn't put it beyond the reach of individuals and small businesses. With a user-friendly interface that can be spun up in seconds, developers can get to work quickly with virtualized Linux instances.

The new US location in Ashburn, Va., is in the heart of one of the most important data center capitals in the world, named Data Center Alley. The new Ashburn facility is available for all Hetzner cloud servers and offers the full array of Hetzner cloud features leveraging AMD EPYC 2nd Gen processors.

With more than 100,000 servers, Hetzner is one of the largest web hosting and data center providers in Europe. In addition to their well-established cloud products, Hetzner's portfolio includes dedicated root servers, web hosting packages, and colocation services. With a combination of reliable tech solutions, attractive pricing, expert support, and an ever-expanding array of services, Hetzner Online has been able to strengthen its market position both nationally and internationally.

Read more about the products Hetzner has to offer (*https://www.hetzner.com/cloud*).

KDE Plasma 5.24 Introduces Fingerprint Reader Support

KDE Developer Nate Graham announced last week (*https://pointieststick.com/2021/10/22/this-week-in-kde-fingerprint-reader-and-nvidia-gbm-support/*) that KDE Plasma would be receiving fingerprint reader support in the upcoming 5.24 release. The added support has been a work in progress for some time, but Devin Lin (the primary developer on the feature) finally merged it into 5.24 (*https://invent.kde.org/plasma/plasma-desktop/-/merge_requests/149*).

As of now, the fingerprint reader support will allow you to enroll and unenroll fingerprints. Any enrolled fingerprint can then be used to unlock the screen, provide authentication for an app, and authenticate for sudo usage.

The developers have created a user-friendly GUI for onboarding fingerprints, which can be found in System Settings but will (obviously) require either a built-in or external fingerprint reader to use. The one caveat is finding an external fingerprint reader that is fully supported by Linux. Most all laptop fingerprint readers should, however, work as expected.

On top of fingerprint reader support, KDE 5.24 will also include Wayland support for DRM leasing (on VR headsets), as well as support for the NVidia driver's GBM back end, new screenshot features in Spectacle, a default blurred background effect for the Overview, and plenty of bug fixes and performance improvements.

The 5.24 release is scheduled for February 2022.

Ubuntu 21.10 Released and Finally Includes Gnome 40

Gnome 40 has been out for some time. For those that have experienced it, you know the new horizontal workflow is a game changer. With a much-improved ability to interact with and manage workspaces, the open source desktop makes daily usage considerably easier and more efficient.

Finally, this new workflow arrives in Ubuntu with the 21.10 iteration.

If you've already been working with Gnome 40 on a different distribution, this news is old. But for those who prefer to stick with Canonical's version of Linux, this upgrade is big. Along with the new horizontal workflow, you'll also see a new Yaru Light theme, which is now the default. The new theme has light gray header bars to better align with how developers wanted their apps to look all along.

Another interesting and important feature is that Firefox is now installed as a Snap package. This means users should receive updates faster than with previous iterations because Mozilla will now have more control over how quickly the updates can be pushed out.

You'll also find Linux kernel 5.13, which adds more security (such as the addition of the runtime memory error detector KFENCE), and even more device support.

And, finally, the NVidia closed source driver supports Wayland. This is a big to-do for anyone who uses NVidia GPUs and wants to experience the dramatically improved performance Wayland brings compared to the traditional X server.

Download your copy of Ubuntu 21.10 (*https://ubuntu.com/download/desktop/thank-you?version=21.10&architecture=amd64*) and experience what's new. And for more information on Impish Indri, check out the official release notes (*https://discourse.ubuntu.com/t/impish-indri-release-notes/21951*).

MORE ONLINE

Linux Magazine
www.linux-magazine.com

ADMIN HPC
http://www.admin-magazine.com/HPC/

Processor Affinity for OpenMP and MPI
• Jeff Layton
Processor affinity with serial, OpenMP, and MPI applications.

ADMIN Online
http://www.admin-magazine.com/

Automate Complex IT Infrastructures with StackStorm
• Holger Reibold
StackStorm is an open source, event-based platform for runbook automation.

Manage Cluster State with Ceph Dashboard
• Martin Loschwitz
The Ceph dashboard offers a visual overview of cluster health and handles baseline maintenance tasks; with some manual work, an alerting function can also be added.

U-Move Backs Up, Restores, and Migrates Active Directory Environments
• Christian Knermann
U-Tools Software's U-Move promises significantly simplified backups and restores of Microsoft's directory service in the event of a disaster, during migrations, and when setting up test environments.

Hive Ransomware Hitting Linux and FreeBSD Systems

Slovak security firm, ESET, has discovered versions of the Hive ransomware for both Linux and FreeBSD systems. However, the encryptors that have been developed for these systems are still in development and are quite buggy. In fact, according to ESET researchers, both encryptors completely fail when the malware payload is executed with an explicit path. And in comparison to the Windows version of Hive, the Linux/FreeBSD iteration only includes one command-line parameter (`-no-wipe`). When executed without root permission, the Linux variation of Hive fails to trigger the encryption because it isn't capable of injecting the ransom note into the device's root file system.

Hive is a ransomware group that has already affected more than 30 organizations but only counts as their victims those who have refused to pay the ransom to get their data back. According to Fabian Wosar, "The reason why most ransomware groups implemented a Linux-based version of their ransomware is to target ESXi specifically." ESXi is VMware's bare-metal hypervisor.

Because of the continued rise in targeting Linux systems with ransomware, it has become even more important that admins keep their systems up to date and make use of tools such as Rootkit Hunter.

Read the original Twitter thread from ESET research on the issue at *https://twitter.com/ESETresearch/status/1454100591261667329*.

SUSE Reaches Beyond the Edge with SUSE Linux Enterprise Micro 5.1

SUSE has offered a lightweight version of SUSE Linux Enterprise (SLE) for some time now. This version of their enterprise OS is purpose-built for containerization and virtualization. But as of version 5.1, it adds a third use case: edge.

Three of the exciting new edge-centric features are secure device onboarding, live patching, and the ability to enable the modernizing of workloads with support for IBM Z and LinuxOne.

SLE Micro is built to scale, which means enterprise users can incorporate the platform into their digital transformation, even when deployed on the edge. These deployments can help with the migration from monoliths to micro-services at any pace.

Of this new release, Thomas Di Giacomo, SUSE chief technology and product officer, says, "SLE Micro is rapidly becoming a critical foundation of customers' digital transformation, as evidenced by a large U.S.-based systems integrator choosing SLE Micro to modernize their embedded systems with a seven-figure investment." Giacomo adds, "They want to support container workloads on an immutable infrastructure that is easy to maintain and update, enabling them to reduce maintenance costs and modernize their systems infrastructure. This win, within six months of SLE Micro's introduction, underscores the enterprise readiness of SLE Micro, which is the result of leveraging decades of enterprise-hardened technology components of the SUSE Linux Enterprise family."

The benefits of SLE Micro include:

- Decreased deployment time and fewer manual processes with improved onboarding security through secure device onboarding of appliances and devices.
- Reduced costly downtime per device with live patching of the kernel.
- Capability for the gradual modernization of applications toward a microservice-based architecture.

Find out more about SUSE Micro at *http://www.suse.com/products/micro*.

Zack's Kernel News

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

By Zack Brown

Author

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

Wonky Typecasting

Tinkering with strange forces, Steven Rostedt felt that a particular struct in the kernel should be tucked away into a private file, and the parts of the kernel using that struct should only be given a pointer to it. However, some (but not all) compilers started to complain about those in-kernel users trying to dereference the pointer (i.e., to see what it pointed to, which Steven had rendered hidden and inaccessible).

However, it was a little more complicated than that. The code that dereferenced the pointer wasn't targeting this specific pointer; it would simply dereference any pointer that was handed to it. But in this case, the struct's pointer was a secret hidden struct and was thus different from the types of pointers that code was usually handed, so … boom.

To fix this, Steven went through a big pile of kernel code and changed a lot of pointer declarations so that the dereferencing code wouldn't complain anymore. He didn't change it to be more accurate; he just changed it so the dereferencing process itself wouldn't care what was at the end of its request.

Specifically, he changed all occurrences of

```
typeof(*p) *_p
```

to

```
typeof(p) _p
```

On the mailing list, Steven said he "Also had to update a lot of the function pointer initialization in the networking code, as a function address must be passed as an argument in RCU_INIT_POINTER() and not just the function name."

Simple, straightforward, and unobjectionable in any way, right?

Mathieu Desnoyers pointed out that Steven's change "removes validation that @p is indeed a pointer, so a user might mistakenly try to use rcu_dereference() on an integer, and get away with it. I'm not sure we want to loosen this check. I wonder if there might be another way to achieve the same check without requiring the structure to be declared."

So Mathieu wasn't totally opposed, but he did notice the swirling madness gathering around Steven's cauldron.

Steven replied:

"Is that really an issue? Because you would be assigning it to an integer.

```
x = rcu_dereference_raw(y);
```

"And that just makes 'x' a copy of 'y' and not really a reference to it, thus if you don't have a pointer, it's just a fancy READ_ONCE(y)."

But Mathieu quoted from the `Documentation/RCU/arrayRCU.rst` file, which said explicitly:

"It might be tempting to consider use of RCU to instead protect the index into an array, however, this use case is not supported. The problem with RCU-protected indexes into arrays is that compilers can play way too many optimization games with integers, which means that the rules governing handling of these indexes are far more trouble than they are worth. If RCU-protected indexes into arrays prove to be particularly valuable (which they have not thus far), explicit cooperation from the compiler will be required to permit them to be safely used."

Paul E. McKenney applauded Mathieu's documentation research abilities and quipped, "On the other hand, I am starting to believe that explicit cooperation from compilers might actually be forthcoming in my lifetime, so there might well be that …."

At this point, Linus Torvalds came into the conversation with a disgruntled critique. He said:

*"This is a sign of why we did it the way we did with that odd 'typeof(*p)*' thing in the first place.*

"The thing is, in any normal C, the function name should just stand in for the pointer to the function, so having to

add a '&' to get the function pointer is somehow odd.

"So I think you should just expose your type to anybody who uses a pointer to it."

Steven shrugged, put the lid back on his cauldron, and said:

"I'll go punt and just expose the structure. It's not a big deal, but I like abstraction of structures when they can be, just to keep from the temptation of tweaking them directly, and causing updates later to be more difficult.

"Too bad that the failure here is not RCU or the macros, but what I would call a bug in a specific compiler."

Linus said, "I'm admittedly surprised that something like this would be a 'different compiler versions' issue. But 'typeof()' isn't exactly standard C, so the fact that some version of gcc did something slightly different is annoying but I guess not _that_ surprising."

However! If anyone expected the story to have a happy ending, they were soon disabused, at least for now.

Jan Engelhardt unlidded his own vaporous cauldron and ladled out some horrific insanity that he felt might solve at least part of the per-compiler issues dereferencing Steven's nasty pointer. He said:

```
#define static_cast(type, expr) ((
  struct { type x; }){(expr)}.x)
typeof(p) p1 = (typeof(p) __force)
  static_cast(void *, READ_ONCE(p));
```

"Let the name not fool you; it's absolutely _not_ the same as C++'s static_cast, but still: it does emit a warning when you do pass an integer, which is better than no warning at all in that case."

Steven, donning his blue hat with yellow stars once again, replied, "Are you suggesting I should continue this exercise?"

To which Jan with mischievous encouragement replied, "Why not?"

Steven, rolling up the sleeves of his robes, said, "I may try it, because exposing the structure I want to hide, is pulling out a lot of other crap with it."

This was why he'd been trying to hide that struct in the first place. The struct, itself, contained pointers to other structs and unions that might also have to be exposed, not to mention various locks and types. It was like Chihiro in *Spirited Away* pulling that last little cork out of the Stink Spirit, and Steven feared all the pollution ever dumped into an entire river might come pouring out.

Linus said, "One option is just 'don't do rcu_access of a pointer that you're not supposed to touch in a file that isn't supposed to touch it'."

He went on:

"IOW, why are you doing that

```
pid_list = rcu_dereference_sched(
  tr->function_pids);
```

"in a place that isn't supposed to look at the pid_list in the first place?

"Yeah, yeah, I see how you just pass it to trace_ignore_this_task() as an argument, but maybe the real fix is to just pass that trace_array pointer instead?

"IOW, if you want to keep that structure private, maybe you really just shouldn't have non-private users of it randomly doing RCU lookups of it?"

To which Steven replied, "The problem is, the RCU isn't for touching it, it is for knowing it exists."

Steven went on to explain some of the deeper, darker recesses of his potion's efficacy. The whole issue stemmed from the fact that he was trying to do a bigger change, and "this was the best 'incremental' approach I had, as the code is currently all just open coded."

It's a longstanding practice for the top developers to try to break their giant dragons into readable minnows so their patches each do something small and clear – Linus once pointed to Alexander Viro as an exemplar of this style of patch submission. Ever since then, it's been every developer's common practice to at least make the attempt.

But in this case, Steven said, "the logic to synchronize updates is left to the user not the pid list itself." And therefore, a whole bunch of data allocation, swapping values, and other weird sorcery taking place that deep in the code, Steven said, was not actually related to what he was trying to do – it just happened to be there standing in his way. He wanted to avoid it as gracefully as possible, which meant rearranging the guts of that beast.

Steven summed up his situation by saying, "I don't believe there's anything wrong with returning a pointer of one type, and then typecasting it to a pointer of another type. Is there? As long as whoever uses the returned type does nothing with it."

And that was the point – he didn't actually want to do anything at that place in the code. He just wanted something that wasn't supposed to happen anyway – those annoying compiler errors – to stop happening.

But Linus, at this point, put his foot down. He said:

"Just stop doing this.

"Dammit, just include the header file that defines the type in the places that you use the thing.

"Because, yes, there is a LOT wrong with just randomly casting pointers that you think have the 'wrong type'. You're basically taking it on yourself to lie to the compiler, and intentionally breaking the type system, because you have some completely bogus reason to hide a type.

"We don't hide types in the kernel for no good reason.

"You are literally talking about making things worse, for a reason that hasn't even been explained, and isn't valid in the first place. Nothing else in the kernel has had a problem just declaring the damn type.

"If there was some clean and simple solution to the compiler warning problem, that would be one thing. But when you think you need to change core RCU macros, or lie to the compiler about the type system, at that point it's not some clean and simple fix any more. At that point you're literally making things worse than just exposing the type."

At which point Steven returned to his original pre-Jan-incitement concession, saying, "Fine, I'll just create a separate header file with all that is needed and add it to the include. At least that way, it doesn't muck up the rest of the header file."

And Jan, at this point siding perhaps with Linus or perhaps with the mere concept of seeing how much smoke could pour out of as many cauldrons at once, pointed to the actual C++ reference, regarding Steven's remark that there couldn't be anything wrong with returning a pointer of one type and then casting it to the pointer of another, especially if the user of that pointer just threw it away.

Jan said that what Steven wanted was actually illegal according to the standard. The reference material said, "If a

pointer to object is converted to a pointer to void and back, its value compares equal to the original pointer." And, "No other guarantees are offered."

So, at this point, Linus looked at the ground, kicked with the toe of his shoe into the dirt a little, hands behind his back, and said:

"Well, we happily end up casting pointers to 'unsigned long' and back, and doing bit games on the low bits of a pointer value.

"So it's not like the kernel deeply cares about theoretical portability.

"But I do discourage casting when not required, just because as much static type checking we can possibly have is good when we can do it."

At this point, the conversation flickered and died, with Steven saying, finally, "I solved this by creating a separate header for the nasty structure, but it's still public for all references."

The thing I love about this whole conversation is just the sheer weirdness of what Steven was trying to do, and the strange way it bit him in the ass. He wanted to hide some highly messy code from the rest of the kernel, as a way to simplify the lives of other maintainers, so they wouldn't feel any urge to tinker in that code while trying to fix anything nearby.

However, by doing this in a way that seemed to make sense to him, Steven triggered something strange and unexpected in the way a whole bunch of compilers were implemented differently from each other. And as Linus said, it's not like the kernel is a standards freak. Linus breaks standards whenever, in his judgment, the standard is wrong. But that's hardly a clear bit of guidance. And not Steven or anyone else can be faulted for going around the rules, or trying to, when something seems to need it.

So Steven was willing to abandon the attempt to hide the crazy pieces until Jan dared him to continue, but in fact it was just too nightmarish. It was probably somewhat fun (I would guess) for Steven to make the attempt and learn things he hadn't known before. But it was also a lot of work for nothing, since Linus finally wanted something simpler, even if that meant exposing the ugly thing that other developers might get confused by later. ■■■

Linux on a smartphone with PostmarketOS

Open Call

The mobile Linux distribution PostmarketOS is a fork of Alpine Linux that supports around a dozen user interfaces. *By Ferdinand Thommes*

Mobile Linux operating systems have experienced an enormous upswing in recent years due to the development of Linux smartphones like the PinePhone and the Librem 5. Around two dozen Linux smartphones exist today, and you can find the 15 most important operating systems on the ISO of a multi-distro demo known as Megi's multi-boot image [1]. Fragmentation in the mobile Linux operating system niche is high, but there is also some cross-fertilization in the scene.

The incentive is the desire to counter the duopoly of Android and iOS with something that open source enthusiasts can use without hesitation. The goal is also about sustainability. Frugal users believe that mobile devices need to be supported for ten or more years, instead of dropping out of the update process after two years, as is usually the case with Android, and ending up as hazardous waste. *Mainlining* [2] is an important keyword here – the term refers to the process of replacing the supplied kernel with a kernel that corresponds as closely as possible to the mainline kernel at kernel.org.

At the center of this development is the open source project PostmarketOS (PMOS) [3], founded in 2017 as a fork of the minimalist Alpine Linux distribution [4]. PMOS supports several user interfaces (Figure 1), including KDE Plasma Mobile, Unity 8 (the UI now known as Lomiri), Purism's GTK-based Wayland Phosh interface developed for Librem 5, the tiling compositor Sway, the Suckless-based Sxmo, Gnome, Mate, and Xfce4. Devices like the PinePhone or the Librem 5 run a mainline kernel out of the box, but the intent is for all Android devices with PMOS to operate with a mainline kernel in the long run, if possible.

PMOS is best known for supporting the PinePhone and Librem 5 phones, but the PMOS community supports 14 devices at various stages of usability. These devices include the Nokia N900, Wileyfox Swift, OnePlus 6, BQ Aquaris X5, and Pine64's PineTab tablet. In addition, around 300 devices are capable of booting from a PMOS image; around 90 of them already have

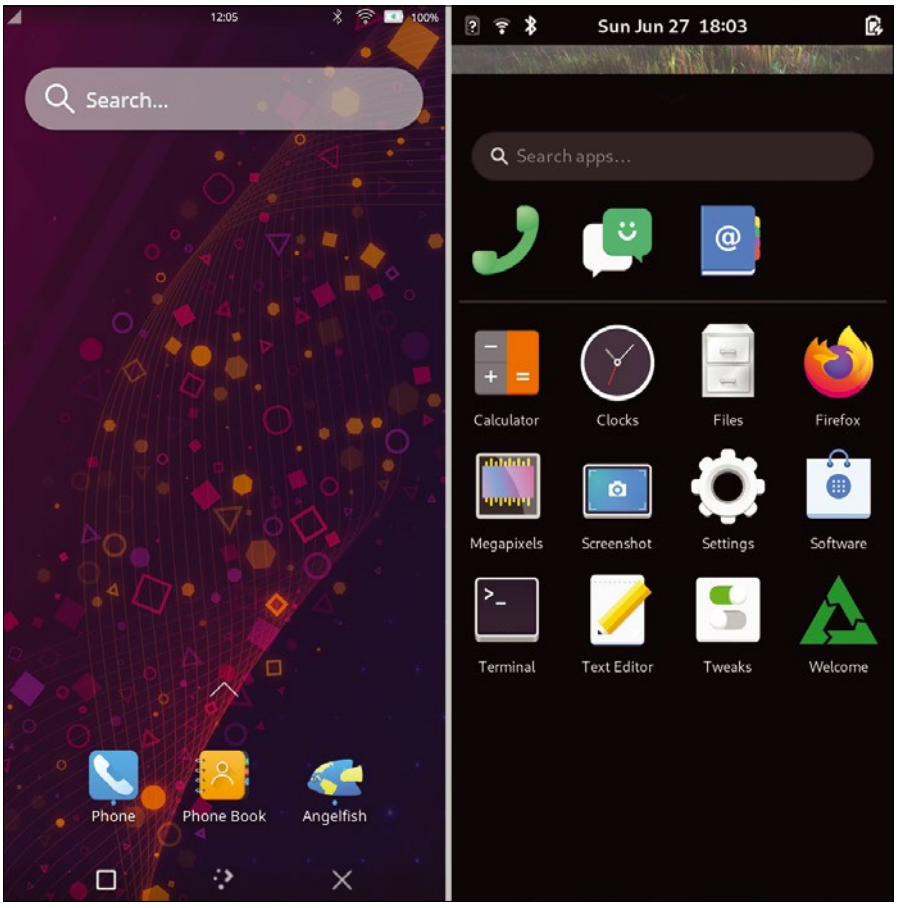
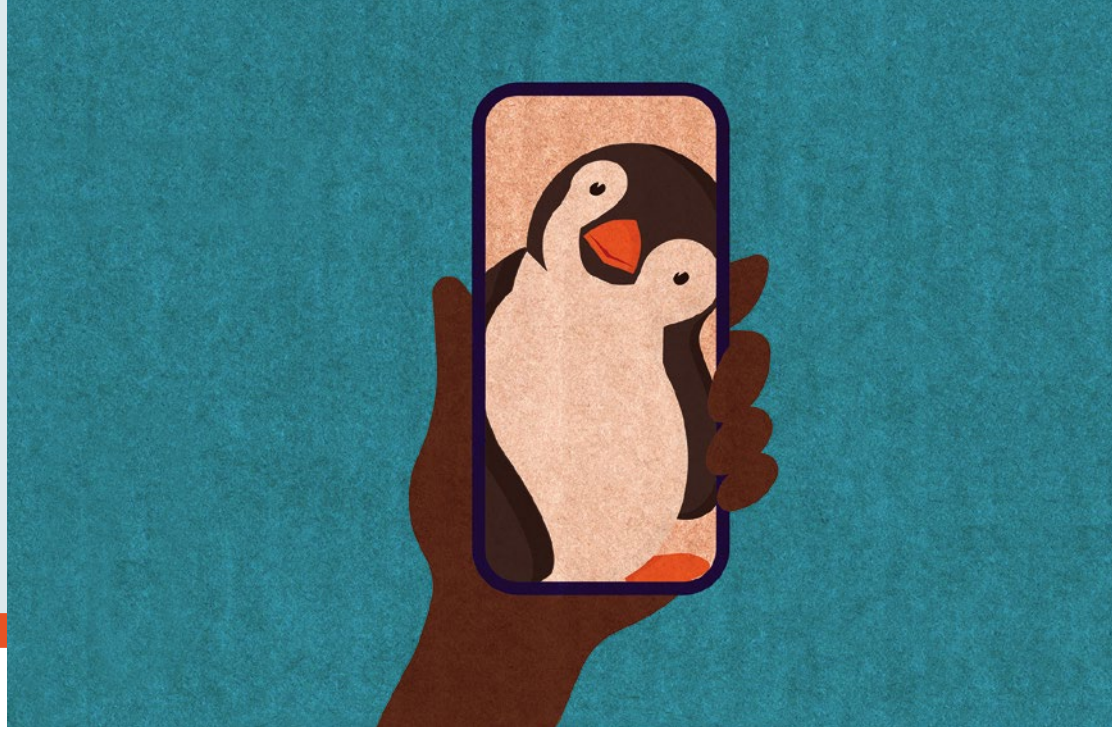


Figure 1: Plasma Mobile (left) is one of the oldest mobile environments for Linux smartphones. The Phosh interface (right) was developed by Purism for the Librem 5.



mainline support. As soon as maintainers can be found for these devices, they will be promoted to the rank of community projects. The developers maintain a list of all supported devices in the project's wiki [5].

Recently, PMOS developers have also been working on making the distribution available for smartwatches. To achieve this smartwatch support, they are integrating the user interface of AsteroidOS [6], which was developed specifically for smartwatches and is based on Qt 5 and QML. Here, too, the main aim is to take the fight to proprietary devices for the wrist, which often have even shorter-lived software support than smartphones, by offering an operating system with long-term support. AsteroidOS currently supports a total of 16 smartwatches [7].

Quick Install

For less tech-savvy contemporaries, installing an Android ROM like LineageOS on an Android cell phone is still an error-prone endeavor, even after years of development. The new swarm of Linux smartphones simplifies this process considerably. First, you transfer the image to an SD card with the Linux `dd` command or a tool like USBImager. After removing the back of the PinePhone, you insert the card into the slot above the SIM card and boot from there. This is more or less the same as testing the Live version of a distribution for the desktop from a USB stick. The aforementioned multi-boot ISO, which combines 15 distributions for testing on an SD card, is a good option.

Once you have stored the multi-boot ISO on the SD card and inserted it into the PinePhone or other supported device, you will see a selection menu after startup that lists the available user interfaces, including five PMOS variants with Plasma Desktop, Plasma Mobile, Phosh, Gnome, and Sxmo (Figure 2). You can navigate through the options with the volume rocker and invoke the selected interface with the Start button. At the very top of the selection, you will possibly find a previously installed distribution in the fixed internal eMMC memory. At the bottom, you can activate a private mode that disables wireless support, the camera, and the microphone for the test.

This ISO offers the fastest starting point for getting to know PMOS in its entirety on the PinePhone or other supported smartphone. If you prefer to test the different variants individually or have already decided on an interface, you can find detailed instructions and reports on the PMOS wiki describing what already works with the interface and what will take some time [8].

The distributions gathered on the multi-boot ISO are meant for testing, not for installation. The images of the individual distributions offer some advantages. One benefit is that you can create a more up-to-date image of your desired interface, because you design the image yourself using `pm-bootstrap` [9], which is part of PMOS. And this method automatically expands the root filesystem to the entire available space on the SD card. In addition, you can store a public key

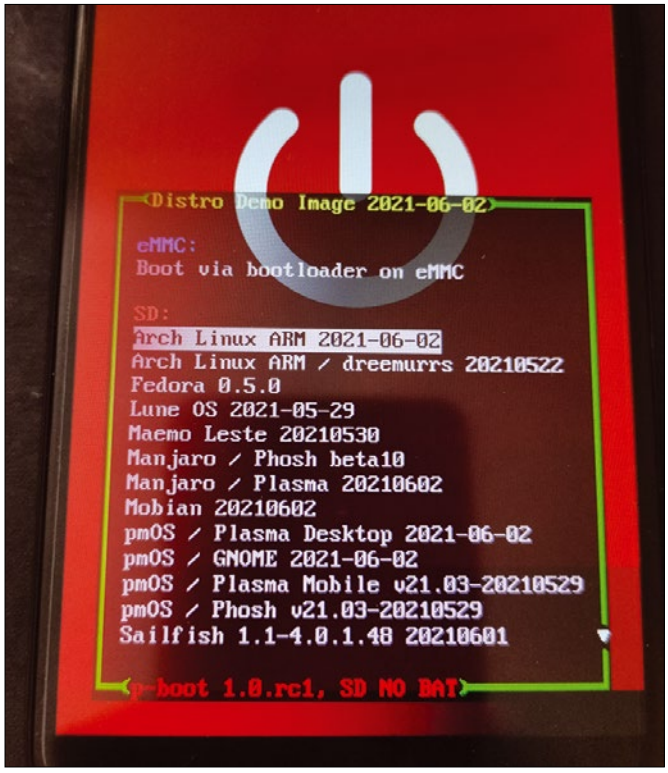


Figure 2: Megi's multi-boot ISO is a useful option for getting to know PostmarketOS, with four different environments in addition to distributions like Mobian, Arch, Fedora, and Sailfish.

for SSH directly on the system if it is on the Linux PC on which `pmbootstrap` is running.

If you are experimenting with PMOS, you need to be aware that the project is entering uncharted territory and that the distribution is still under development; a first stable version was released in early July 2021. However, in combination with the inexpensive PinePhone, it makes for an ideal experimental field to familiarize yourself with the new Linux phone niche. If you keep that in mind, there is nothing standing in the way of an installation on the PinePhone's internal eMMC memory card. You can install directly via `pmbootstrap` or transfer PMOS from a booted SD card to the eMMC with Jumpdrive [10].

Listing 1: Installing PMOS on an SD Card

```
### Install Python-Pip3 on the PC
### Debian:
$ sudo apt install git python3-pip
### Fedora:
$ sudo dnf install git python3-pip
### Install Pmbootstrap:
$ pip3 install --user pmbootstrap
### Create an environment:
$ pmbootstrap init
### Release Channel: stable or edge
### Choose a vendor: pine64
### Codename: pinephone
### Define timezone, name etc.
### Copy SSH key if needed
### Start installation. Before doing so run
### fdisk -l to check if mmcblk0
### is the right device
$ pmbootstrap install --sdcard=/dev/mmcblk0
###
```

PinePhone

PinePhone's development stagnated for quite some time with the suspend feature, or rather, the inability to wake up from it early enough when a call comes in. The problem has now been fixed, so you can use hibernation mode and answer calls on the first ring. This also ensures an acceptable runtime.

In addition, the camera was an unusable component for a long time, since it lacked an app capable of taking pictures. PMOS developer Martijn Braam, who is enthusiastic about photography, remedied this with his Megapixels app. Developed for PMOS, it now enjoys widespread use on other mobile distributions. Braam described the journey in a series of four blog entries [11]. He also wrote an implementation of Gnome Tweaks for PMOS, which goes by the name of postmarketOS Tweaks [12].

Listing 1 summarizes the commands you use to put PMOS on an SD card, which you then boot on the PinePhone. The PIN you are prompted for at startup time and the passwords for user and root are 1111 for Megi's multi-boot ISO. For a

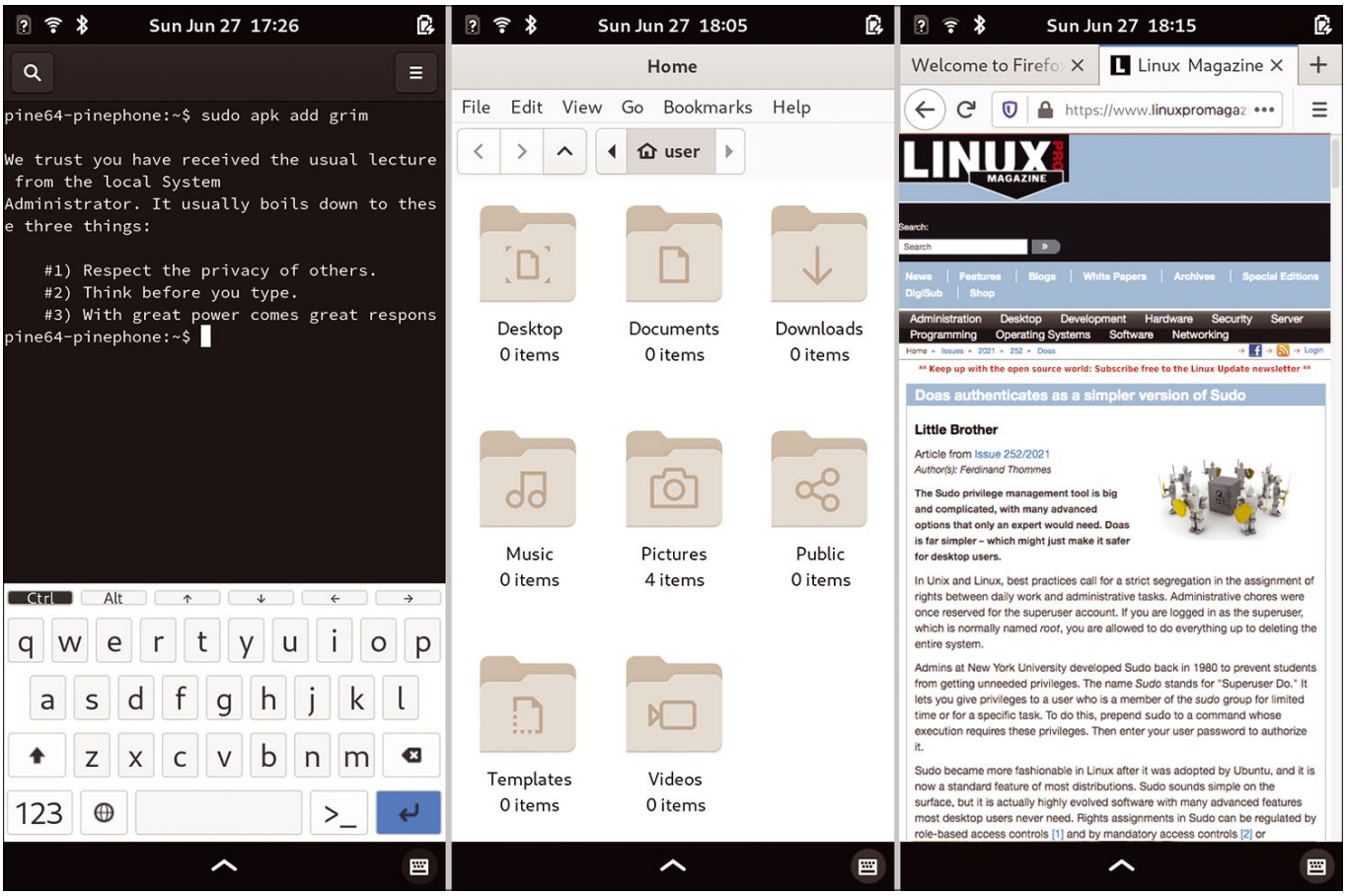


Figure 3: As a fork of Alpine Linux, PMOS uses Alpine's package manager apk (left). A mobile-ready version of the Gnome Files app is used as a file manager (center). Not all mobile distributions have adapted Firefox as well as PMOS with Phosh (right).

user-designed image, you can set these details during the bootstrap process.

You can install additional packages using the Alpine package manager, `apk`. For example, to install Braam's photo app, use `sudo apk add megapixels`. Alpine does not use systemd, but rather RC as an init program. Enable SSH by typing `sudo rc-service sshd start`. To enable SSH at every reboot, run the `sudo rc-update add sshd` command. As usual, you will find the IP address of the PinePhone needed for SSH with the `ifconfig` or `ip a` commands.

We found the Librem 5 interface Phosh to be the most likely to show PMOS in its best light. The version from Megi's ISO shipped with kernel 5.13-rc4. In addition to the telephony app, the chat module Chatty, and the contacts app, and the Firefox browser, you'll find some familiar Gnome programs, such as Files, Software, Settings, Terminal, and Tweaks (Figure 3). A Welcome menu explains the various gestures.

We post-installed Gnome Screenshot, Gnome Maps, and htop in our lab, and all of them work. However, due to the PinePhone's low-powered hardware, things are quite slow. And not all apps fit the form factor perfectly as of yet, which means that you have to rotate the device through 90 degrees to access the buttons for some actions.

Conclusions and Outlook

All told, PostmarketOS's current features will not be enough for many wanting to use the PinePhone as their sole daily companion. This hesitancy is partly due to the limited hardware and partly due to the early stage of development for PMOS itself.

Many users are longing for a Linux PC in their pockets, and the fact that it is taking so long is a disappointment for many enthusiasts. But you have to remember that the community is just now laying the foundations for – hopefully sustainable – long-term development. ■■■

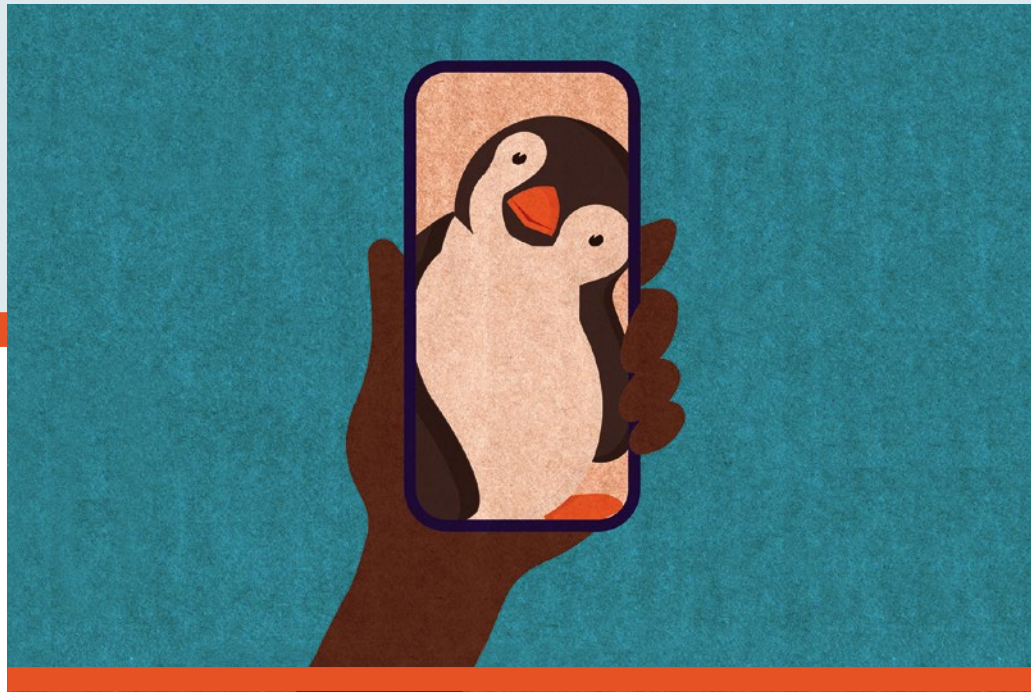
Info

[1] Megi's image: *https://xnux.eu/p-boot-demo/*
[2] Mainlining: *https://wiki.postmarketos.org/wiki/Mainlining*
[3] PostmarketOS: *https://en.wikipedia.org/wiki/PostmarketOS*
[4] Alpine Linux: *https://www.alpinelinux.org/*
[5] PMOS supported devices: *https://wiki.postmarketos.org/wiki/Devices*
[6] AsteroidOS: *https://asteroidos.org/*
[7] AsteroidOS supported smartwatches: *https://asteroidos.org/install/*
[8] PMOS interfaces: *https://wiki.postmarketos.org/wiki/User-Interfaces*
[9] pmbootstrap: *https://wiki.postmarketos.org/wiki/Installing_pmbootstrap*
[10] Jumpdrive: *https://github.com/dreemurrs-embedded/Jumpdrive*
[11] Megapixels: *https://blog.brixit.nl/pinephone-camera-pt4/*
[12] postmarketOS Tweaks: *https://gitlab.com/postmarketOS/postmarketos-tweaks*

Fresh Breath of Life

Breathe new life into your old smartphone: The TWRP recovery image and the free Android offshoot LineageOS let you install the latest security updates every week.

By Peter Kreußel



Every mobile device needs its own Android build because of numerous drivers that are not available in the source code. The need to maintain every version of Android for every mobile device means that many manufacturers eventually stop supporting updates. Often, smartphones or tablets that still work perfectly can no longer be used without worry because the manufacturer has simply ceased to offer bug fixes and security updates.

The fact that many drivers for smartphone hardware are not available as source code makes it difficult to provide Android upgrades. The LineageOS project [1], the successor to the CyanogenMod project, which was discontinued in 2016, proves that it is not impossible to keep these devices up-to-date. Unpaid volunteers at LineageOS do the work that many manufacturers do not want to do: They combine current Android releases with the required device-specific drivers.

Dewy

The LineageOS project (Figure 1) provides Android systems with a fresh patch status every month for around 300 devices. The builds are released weekly, unless there is a problem during the build. The *Devices* page on the LineageOS Wiki [2] provides the details of whether a LineageOS build is available for your smartphone or tablet.

Practically all Android devices support a recovery mode that starts a small, specialized Android system to restore the main system in case of damage. Out of the box, recovery mode only installs the state intended by the manufacturer. Therefore, the first step towards installing an alternative OS such as LineageOS is to flash a manufacturer-independent recovery system. I prefer to use TWRP [3] as a recovery tool because, unlike the recovery system supplied by the LineageOS project, it creates backups of the existing system (Figure 2). TWRP (pronounced "twerp"), is a custom recovery tool for Android phones. TWRP replaces the recovery system provided by the hardare manufacturer, which then allows you to install an independent system such as LineageOS.

Unless you unlock the bootloader, you can't install another recovery system or a full system on the Android smartphone

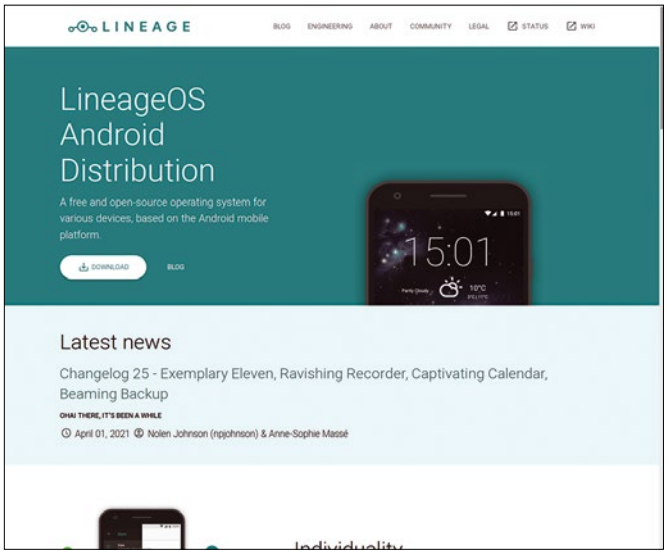


Figure 1: The LineageOS team continues work of the CyanogenMod project, which was discontinued in 2016. LineageOS provides over 300, often older, devices with fresh Android updates every month.

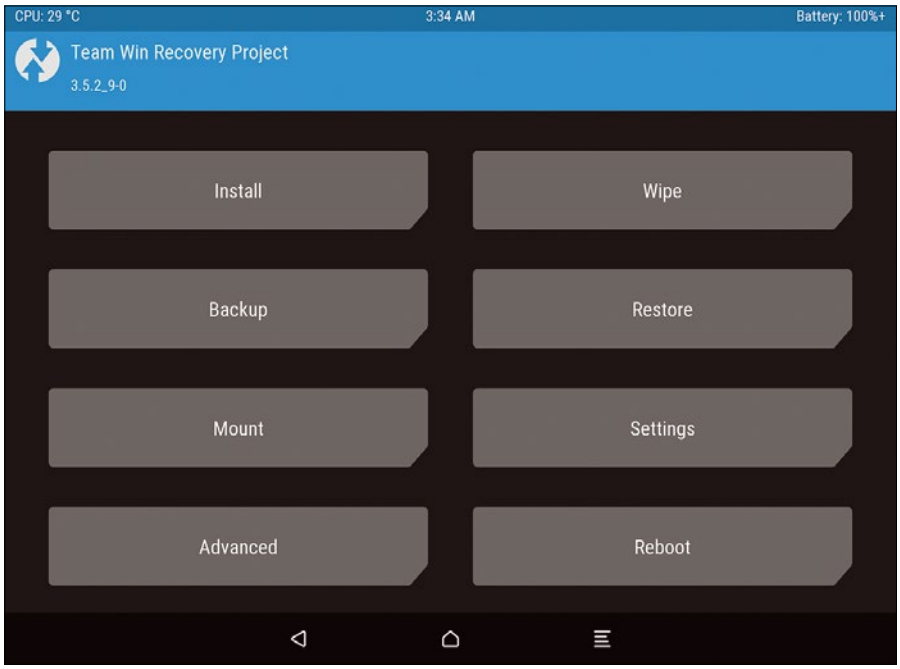


Figure 2: The TWRP recovery system offers a graphical interface that is friendly enough for non-professionals.

later (Figure 3). As of Android 6, you can enable the *OEM unlock* or *Bootloader unlock* item in the Android settings below *Developer options* | *OEM unlock* to unlock the bootloader.

If you do not see the *Developer options* menu item under *System* | *Advanced*, tap the *Build number* displayed at the end of the heading seven times in the *About the phone* settings category. You also need to enable the *USB debugging* option for flashing via USB.

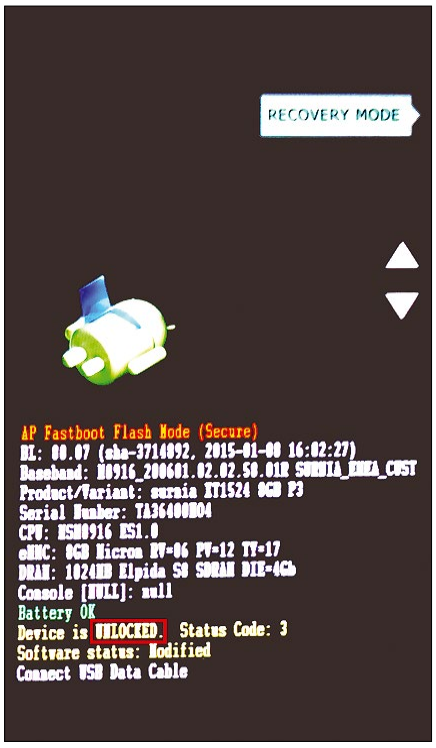


Figure 3: The Android Fastboot bootloader reports "Device is UNLOCKED."

However, this procedure does not remove locks set up separately by the manufacturer to prevent the installation of other software and rooting of the smartphone. Some manufacturers require you to register the device to unlock it, and then – after telling you that the warranty is now void – they hand over a code. Others refuse to unlock the device altogether. However, it is probably not possible, in the EU at least, to make the hardware warranty dependent on you not installing your own software. The LineageOS Wiki page for your device [2] also describes how to unlock the bootloader.

Once you have freed the bootloader, it is time to download the necessary software: First, you need the TWRP recovery image [4] for your device. On your Linux computer, install the Android tools (the Ubuntu packages are *android-tools-adb* and *android-tools-fastboot*, otherwise it's just *android-tools*) containing the Fastboot program as a counterpart to the bootloader on the Android device. Working from the PC, the tools load the recovery image into the partition assigned by the manufacturer's recovery routine. Finally, you need an image of the actual LineageOS system, which the recovery routine finally installs on the system partition after a backup.

Right Choice

Android builds – the final LineageOS as well as TWRP – only work on the device they were built for using the Android open source project and numerous proprietary drivers. To make sure you're installing the right image, you need to know the manufacturer's code name for the device. This will help you identify the right images on the LineageOS and TWRP download pages. The app Device ID [5] gives you the name on a running system (Figure 4).

Variants of a basic device type can exist in some cases; they are stated in the *Device name* line in the *About phone* tab in Android settings. Normally a build should support all variants, but it doesn't hurt to check the LineageOS wiki page for the device matching the codename to see if your model shows up under *Supported models*.

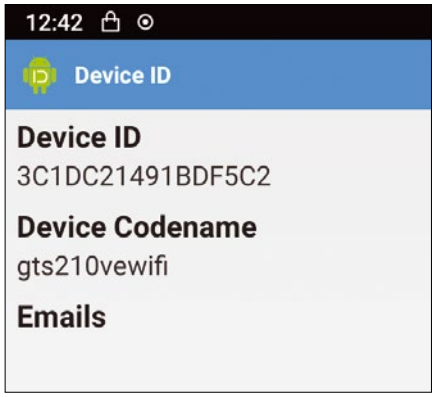


Figure 4: The Android Device ID app shows the code name of the device; you can then use this code name to find the right image to flash.

Once all the software components are in place, boot the Android device to the unlocked bootloader. Most Android devices (with the exception of Samsung, see the box "Installing TWRP on Samsung Devices") rely on Android's own Fastboot bootloader. The LineageOS wiki explains how to access the bootloader for each device. Usually, you can reach the bootloader by turning off the device and then turning it back on while holding down the Volume Down key.

Fastboot fields the data from the PC `fastboot` command-line program and writes it to partitions on the mobile device. The prerequisite is a USB connection between the PC and smartphone.

Installing TWRP on Samsung Devices

Smartphones and tablets by Samsung do not use the Fastboot bootloader, which is otherwise common for Android. Samsung has developed its own Windows tool named Odin for flashing firmware. Fortunately, there is a free Linux port named Heimdall [6]; the popular Linux distributions have Heimdall packages.

Volume Up+Home+Power boots a powered-down device to Download mode, which expects data for flashing via USB. If `heimdall print-pit` on the Linux console prints a series of messages and the device finally boots the normal system, then the communication via USB cable between PC and smartphone is working in Download mode.

After turning the device off again, you can switch back into Download mode using the key combination shown above. The command from Listing 1 installs the TWRP recovery image [4] for the model. Then immediately turn off the device by either removing the battery or pressing Volume Down+Power for eight to 10 seconds. Once the screen goes blank, release the keys immediately. On the powered-off device, press Home+Volume Up+Power to start recovery mode with TWRP. You will now be able to flash Lineage via the recovery option just as you would with other Android devices.

After plugging in the USB cable, first run `adb devices` to test whether this connection with the bootloader launched as described and the console program works (Figure 5).

The command in Listing 2 installs the TWRP recovery system on the recovery partition. The percentage values displayed at the command line during the transfer often turn out to be not very meaningful. The main thing is that the operation completes without an error message.

The factory-installed recovery system is now overwritten. The option to use it to return to the factory state of the smartphone is no longer available. Instead, you can now back up and restore the actual state of the system with TWRP.

Use the volume keys to select which system the currently active bootloader will start. The options *START* (normal system) and *RECOVERY MODE* are important. You can start the recovery that you need next right away or make sure the normal system is working to reassure yourself. In this case, turn off the device again and go back to the bootloader.

Data Manager

TWRP recovery comes with a clear graphical interface (Figure 2). First you need the *Backup* function to save the previous system, which has not yet been replaced by LineageOS. Later on, when you install the weekly LineageOS updates, it always makes sense to back up a version that you know will work up front. The backup only takes a few minutes on a newer device with a sufficiently fast SD card.

The backup screen opens, showing you the partitions *System*, *Data (excl. storage)*, and *Boot* preselected for the backup. These settings are correct for a later restore of an entire system. In *Select Storage*, you now need to select the desired storage medium. The storage medium is usually not *Internal Storage* but an SD card.

Many users move the large backups to the PC immediately after they have been created. You will find the data on the storage medium below `TWRP/BACKUPS/<DEVICE ID>/` in a directory with a name that starts with the date and time and contains a system name such as `lineage`. To restore a backup, restore this folder to the same location. The backups will then appear as a list in the TWRP *Restore* function. After selecting a backup, all you do to restore it is drag a slider to the right. If a file selector appears instead of the list of existing backups, use *Select Storage* to switch between *Internal Storage* and *SD Card*.

A file manager is available on the TWRP home screen after you select *Advanced* (Figure 6). You can access the home screen at any time via the button at the bottom of your screen. Using the file manager is not immediately intuitive: For example, it is important to know that the contents of the SD card can be found in `/data/media/0/`, including the aforementioned `TWRP/` folder. To delete a backup subfolder, open it and tap *Select Current Folder* bottom right. The file manager then displays the *Delete* option (line top right in Figure 6).

Tabula Rasa

Anyone installing a system of a different type, be it a change from the default Android to Lineage or a newer LineageOS version, now needs to trigger a factory reset (*Wipe* in TWRP, the *Factory Reset*, is preselected (Figure 7)). All the installed apps are lost but not the data, such as music files or photos. It goes without saying that a backup of this data is recommended, in addition to the system backup with TWRP.

The Titanium Backup [7] smartphone program gives you an option for backing up the installed apps, including settings before *wiping*, but it requires the device to be rooted up front (Figure 8). In principle, you can use this option to transfer apps purchased in the Play Store to a smartphone without using Google's services because Titanium saves the associated APK installation packages, which can then simply be copied. In practice, the apps usually check whether they are running on a device registered to the buyer and refuse to work or cannot be installed at all. The MicroG framework [8], an open source replica of Google's services, promises a remedy.

The MicroG services also need to contact the Google server for license management but will conceal your per-

Listing 1: Flashing Samsung Devices

```
$ heimdall flash
  --RECOVERY twrp-version_number-Devicecode.img --no-reboot
```

Listing 2: Flashing the Image

```
$ fastboot flash recovery image_file
```

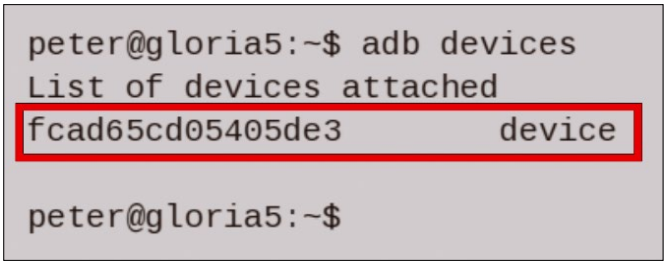


Figure 5: If the list returned by adb devices contains an entry, the USB connection between the mobile device bootloader and PC is working.

sonal data, according to their developers. Their location services work offline, for example. Most apps start without problems in the environment, but some do not [9].

Although MicroG continuously transmits the device IP to Google, many users might be more trusting of an open source app – for example, when it says that it does not read and transmit the GPS coordinates if the location services are disabled. If you insist on a completely Google-free smartphone, you have no choice but to restrict yourself to the alternative F-Droid app store (Figure 9). There are far fewer apps available there but still around 4,500.

MicroG cannot be post-installed on LineageOS because it has to spoof signatures to Google – LineageOS does not allow that at the system level. However, there is the LineageOS fork LineageOS for MicroG [10], which integrates MicroG into an otherwise unmodified LineageOS system. /e/OS [11] goes a bit further, with its own app store and many free apps from the Play Store.

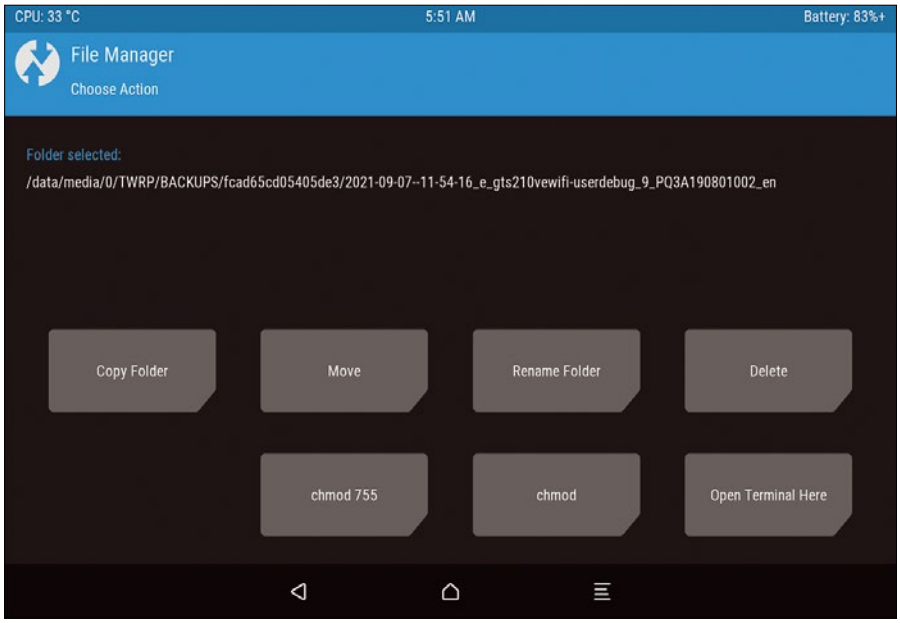


Figure 6: TWRP integrates a file manager that you can use to delete older backups or other data as needed to free up space.

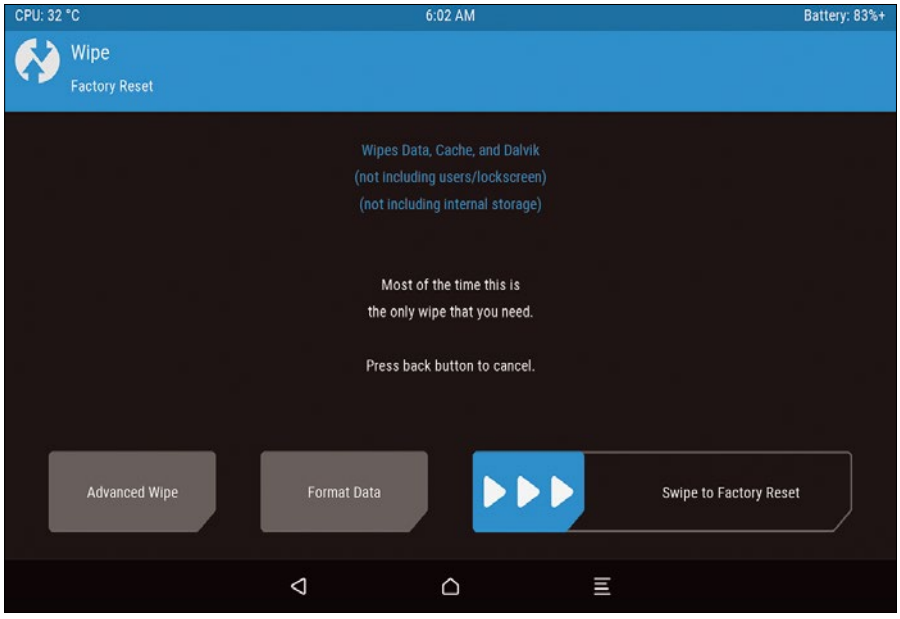


Figure 7: The TWRP Wipe menu item performs a factory reset after dragging the slider – almost always the only reset operation you need, as the program itself correctly notes.

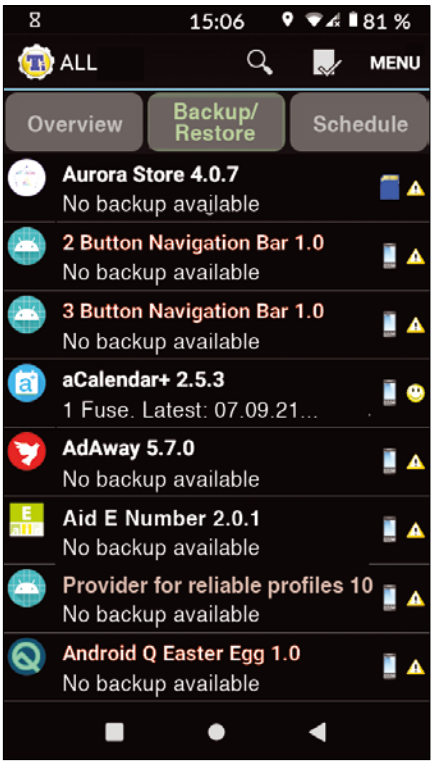


Figure 8: The Titanium Backup app backs up all installed apps, including their settings, on rooted devices and quickly restores them after a factory reset.

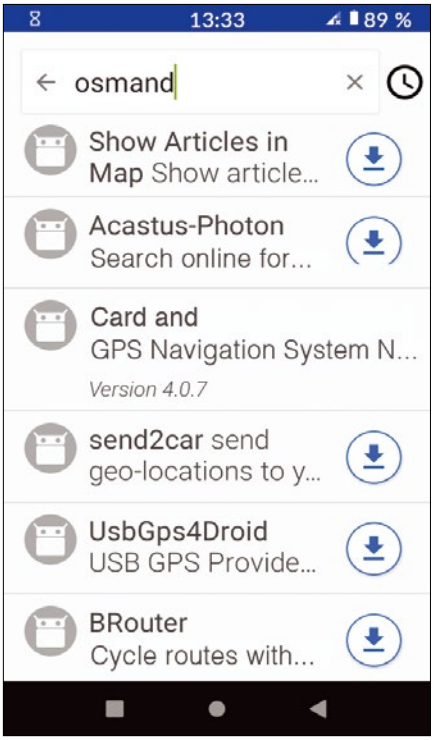


Figure 9: Many authors of free Android software exclusively publish their apps in the alternative F-Droid store, or they use F-Droid in addition to the Google Play Store.

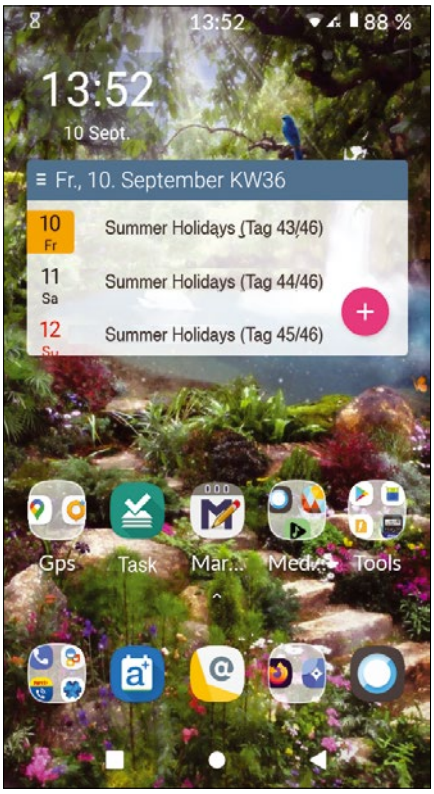


Figure 10: Trebuchet, the default homescreen app on LineageOS, combines good configurability with acceptable resource consumption.

Install

After running the backup function, the next step is flashing. Click the *Install* button in the main menu to get the flashing started. This step installs the ZIP images [2] of the LineageOS distribution. You just need to tap on an archive file in the file manager on the *Install* screen.

In the next screen, the software offers you the option of flashing additional ZIP archives onto the new system (*Add more Zips*), for example, a root extension for LineageOS or the Google Play Store (see box entitled "LineageOS Add-Ons"). You can launch the actual flash process by dragging the *Swipe to confirm Flash* slider to the right.

LineageOS Add-Ons

Many power users root their system, as rooting is the only way to adblock, control cells with a tool such as Tasker [12], or tune the kernel settings for better performance and improved power consumption. However, current LineageOS releases no longer offer the option to root via a checkbox in the developer options. Instead, the project recommends installing Magisk [13]. Rename the downloaded `Magisk-VERSION.apk` file to `Magisk-VERSION.zip`, and TWRP's Install function will recognize it as an installation image that you select with *Add more Zips* before flashing the main system. On a rooted system, install the original `Magisk-v2x.x.apk` package, which refreshes the app without launching the rescue system.

LineageOS does not ship Google Play Store or other company services and apps by default. If you want to use the Play Store or paid apps, you need to install OpenGApps [14] before starting the new system for the first time. Apps are also available as ZIP files, which you can add to the flash process in TWRP by selecting *Add more Zips*. On the OpenGApps page, select the correct variant based on three properties: the processor platform (see the LineageOS wiki page for your device: *Architecture*), the Android version (also in the LineageOS wiki), and the *Variant*, that is, the content scope of the OpenGApps package. The *stock* option sets up the full scope installed on Nexus phones. The *nano* option includes all items not available through the Play Store and is recommended for users who want to build their own systems.

Quickly Served Up

Sideloading lets you you transfer files directly from your computer without requiring you to manually copy the files on the SD card. (TWRP still needs enough storage space for the image on the SD card.) Sideloading also lets you install several ZIPs one after another.

For a sideload, with the smartphone connected to the PC and the TWRP system running, first call `adb devices` to check if the mobile device can be accessed. The adb tool is in the *android-tools-adb* (Ubuntu) or *android-tools* package. The adb call should return a line with a device identifier, followed by *Recovery*. Select *Advanced* | *ADB Sideload* on the TWRP homescreen and drag the appropriate slider to the right.

On the PC, run `adb sideload IMAGE`. Again, the percentage value displayed on the PC console is not very meaningful; it often remains at *0%* for a long time. In our lab, the transfer even supposedly aborted at 49 percent, but TWRP reported a complete installation and the new system booted flawlessly.

Now you have a clean Android system without bloatware, either with or without Google App Store, and with the resource-saving Trebuchet homescreen app (Figure 10), which is perfect for many users. You can select the home app in the Android settings below *Apps and Notifications* | *Default Apps* | *Start App*.

The Android web browser, a calendar, a picture viewer and camera app, a telephony app, email, a text message app, a simple audio player, a voice recorder, a calculator, and a file manager are pre-installed.

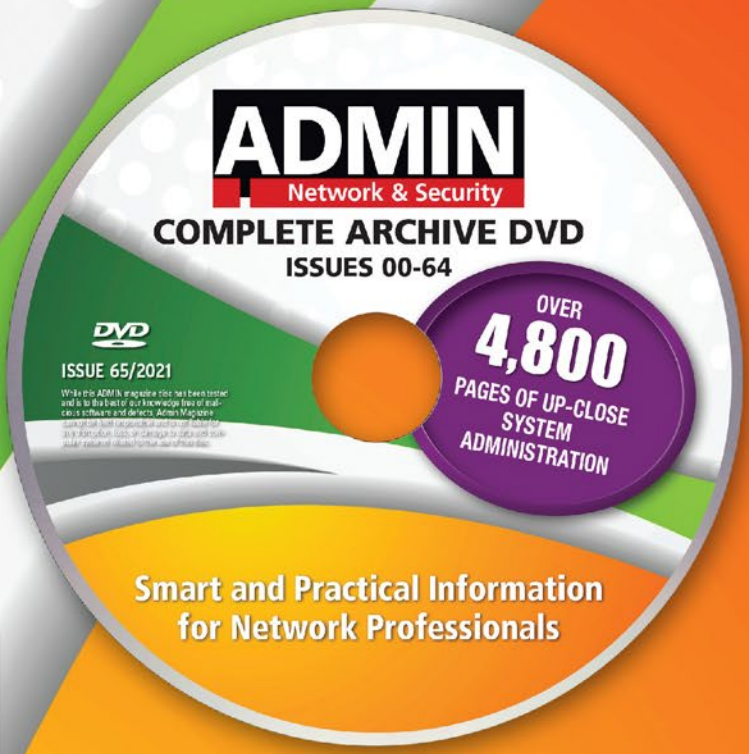
Conclusions

I recommend the LineageOS project as the first port of call for anyone who wants to protect an older smartphone or tablet that is no longer maintained and doesn't receive Google security patches. The LineageOS derivatives LineageOS for MicroG and /e/OS make it even easier to enjoy a Google-free smartphone without too many restrictions. ■■■

Info

[1] LineageOS: *https://lineageos.org*
[2] LineageOS builds: *https://wiki.lineageos.org/devices/*
[3] TWRP: *https://twrp.me*
[4] TWRP builds: *https://twrp.me/Devices/*
[5] Device ID app: *https://play.google.com/store/apps/details?id=com.redphx.deviceid*
[6] Heimdall: *https://glassechidna.com.au/heimdall/*
[7] Titanium Backup: *https://play.google.com/store/apps/details?id=com.keramidas.TitaniumBackup&hl=en_US&gl=US*
[8] MicroG: *https://microg.org*
[9] Apps having problems on MicroG: *https://github.com/microg/GmsCore/wiki/Problem-Apps*
[10] LineageOS with MicroG: *https://lineage.microg.org*
[11] /e/OS: *https://e.foundation/e-os/*
[12] Tasker: *https://tasker.joaoapps.com/*
[13] Root add-on Magisk: *https://magiskmanager.com/*
[14] OpenGApps: *https://opengapps.org*

12 Years of ADMIN on One DVD



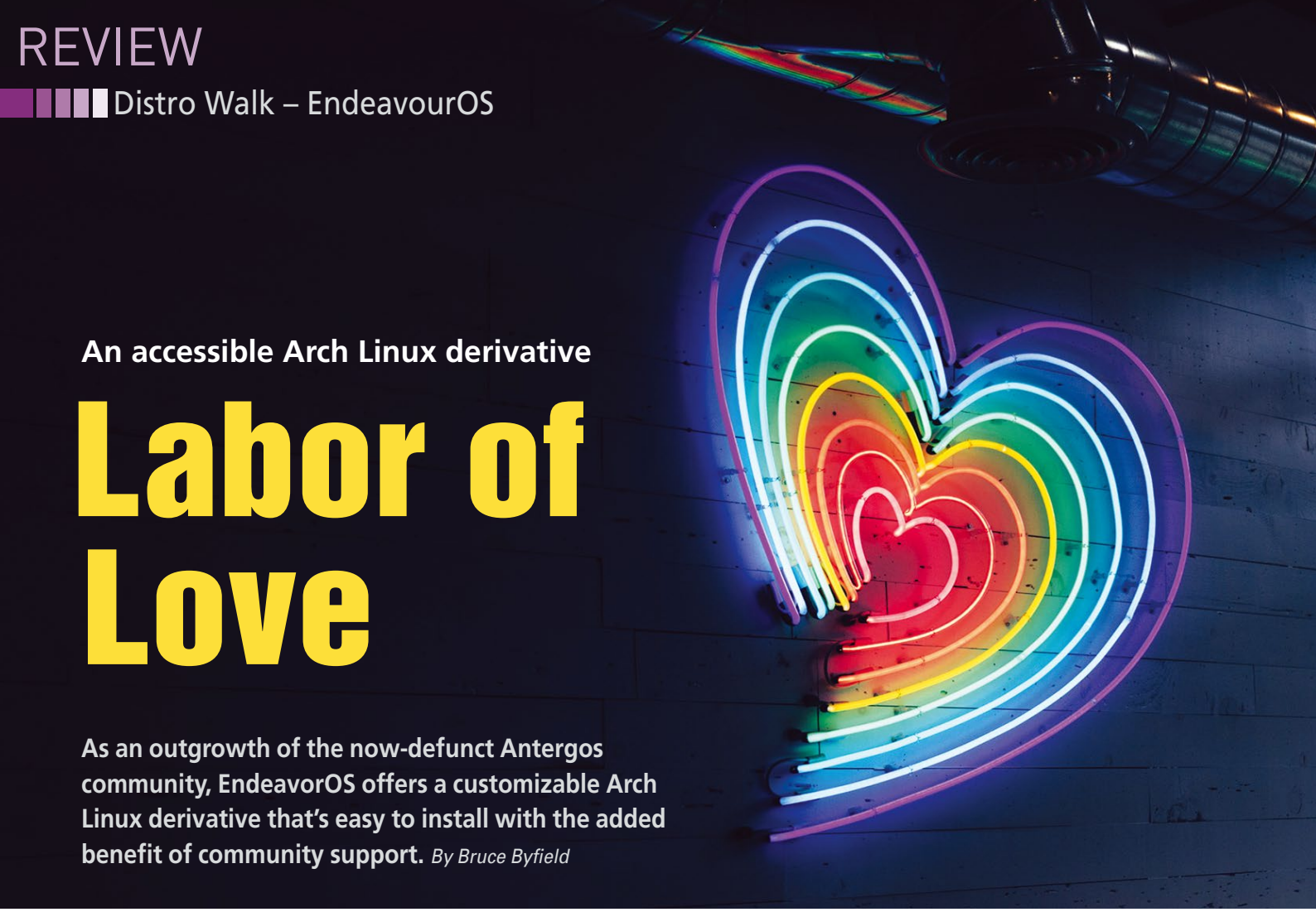
This searchable DVD gives you 64 issues of ADMIN, the #1 source for:

- DevOps
- ransomware
- edge computing
- automation
- orchestration
- and more from the front lines of IT

Clear off your bookshelf and complete your ADMIN library with this powerful DVD!

ORDER NOW!

shop.linuxnewmedia.com



An accessible Arch Linux derivative

Labor of Love

As an outgrowth of the now-defunct Antergos community, EndeavorOS offers a customizable Arch Linux derivative that's easy to install with the added benefit of community support. *By Bruce Byfield*

In the past few years, distributions based on Arch Linux have become more popular. Perhaps their popularity reflects the growing expertise of Linux users, but both Antergos and Manjaro have both been in the top five for page views on DistroWatch. In the past year, EndeavourOS [1], a distribution based in the Netherlands, has also appeared in this spot, holding steady in second or third on the list.

Figure 1: The Calamares installer is one of the major features of EndeavourOS. It produces a command-line installer with some of the tools needed for further customization.

EndeavourOS occupies a definite position among Arch derivatives. Where Manjaro offers an Ubuntu-like desktop experience, Endeavour's focus is on getting users to a successful command-line install with less difficulty than Arch Linux and then assisting with all the resources needed for customization. Similarly, where Arch seems arcanely advanced, EndeavourOS emphasizes the pleasure of learning and doing things your way. As the project's front page says, "If you like to roll up your sleeves and go on an adventure, then you might want to give us a try." Recently, I talked with Bryan Poerwoatmodjo, one of EndeavourOS's founders, about where the distribution has been and where it's going.

Linux Magazine (LM): How did you come to be involved with free software?

Bryan Poerwoatmodjo (BP): All the founders (Joe Kamprad, Manuel, Fernando Omiechuk Frozie, and I) had an early interest in Linux and FOSS. We'd met on the forum of Antergos, an Arch-based distro that offered more or less the same options EndeavourOS is offering now. Subsequently, we all became forum moderators. In particular, Joe, Manuel, and Fernando became some of the trusted faces on the forum for solving issues or sharing shortcut scripts on the forum.

LM: What prompted the founding of EndeavourOS?

BP: It never was our intention to start a distro, but when Antergos announced their end in May 2019, the community was taken by surprise. Through their existence, Antergos not only built a distro with an easy GUI Arch installer (Figure 1), but they also were surrounded by a friendly community that went beyond technical support. It was a place where online friendships developed.

Photo by Jiroe on Unsplash

I opted for a new forum where we could carry on Antergos's community legacy, and Joe Kamprad was the first to join me in that quest. At that point, creating a distro wasn't the plan, but when Fernando Omiechuk Frozie joined, the whole plan changed. He had already developed an unofficial offline Antergos installer with Xfce as its main DE [development environment], and when we approached Manuel, who already created some handy scripts to either solve issues or just for convenience's sake, the project changed from preserving a community to the birth of a new distro that was developed with the existing Antergos community in mind.

LM: How does EndeavourOS interact with Arch Linux?

BP: Package-wise, there is no difference. All our packages and updates are directly coming from the Arch repositories, without any alterations. The only difference between an Arch installation and an EndeavourOS one, besides the use of the Calamares installer, is that we provide the user a jumpstart with some basics like a browser (Firefox) and working network, sound, and graphics for most systems. It is just ready enough for a user to start to customize the system to their liking, without having to deinstall a bunch of unwanted packages we think a user should need. This also means that the user has to be hands-on from the get-go to get features such as Bluetooth, printer drivers, and specific settings, like Gaming or NAS setups, working.

LM: What sort of user is Endeavour directed at?

BP: The majority of our users have been around the Linux block, so to speak. There are two types of users: The first one already knows what they need to customize a system to their specific needs, and the other one is willing to dive into the deep end to learn and discover the possibility an Arch-based distro can offer. [By contrast], a lot of Arch-derivatives either provide an out-of-the-box experience with a separate ISO for each DE, or they are offering a command-line installer. We provide the user a graphical installer that will install an almost vanilla system.

I have to say, occasionally, it confuses some new users, thinking we provide the same out-of-the-box experience that our fellow Arch derivatives provide. Luckily, our community is there to more than try and give such users a lot of support to help them further.

LM: How is EndeavourOS organized and governed?

BP: EndeavourOS is run by five admins: Joe, Manuel, Pudge, Alpix, and I. Fernando left the team due to a career opportunity, which left him with less time to stay involved with the project. Joe is in charge of the development of Calamares and the ISO, Manuel is the developer of our native apps, Pudge is in charge of EndeavourOS ARM, Alpix is our mirror admin, and I'm the community manager and general project leader.

There are also groups of moderators for the forum, Telegram group, and Reddit; translators for our announcements, the Welcome app, and very soon our websites; and a group of developers who are responsible for our community editions Sway, bspwm, and ARM. There are new community editions in the pipeline, and they will be released when ready.

For the Calamares installer, we are actively working together with the Calamares developers and the developers of several distros that are using the installer. Only together we can improve it. For our infrastructure, hardware purchases, and donations to other FOSS projects, we are purely funded by donations from our community. We do keep an eye on our available budget and discuss very often among the admins on how to spend the donations wisely. Unnecessary purchases or paying for too highly priced services are things we want to prevent. In the end, it is the community's trust we are spending, so our spendings are transparently displayed on our Open Collective page [2].

LM: Do you have any statistics about things like the number of downloads and regular contributors?

BP: The number of regular and one-time contributors are transparently displayed on our Open Collective page. At the moment, we are using a handful of mirrors, offered by community members or non-profit organizations, and also GitHub

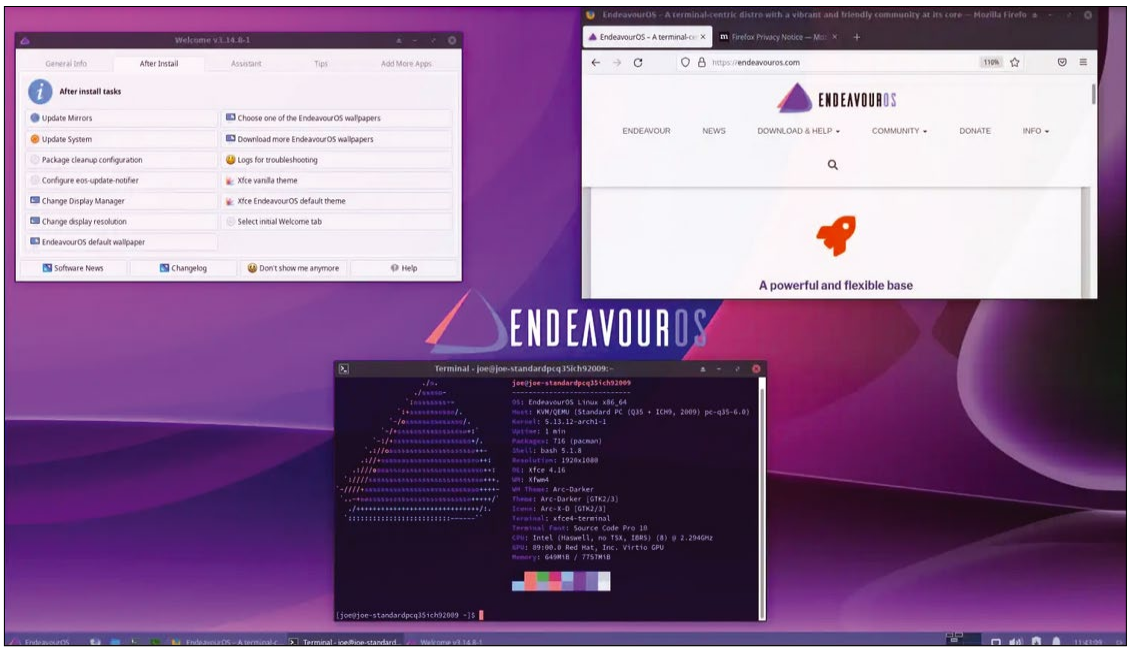


Figure 2: Endeavour's Xfce desktop environment, showing the handy Welcome app.

and GitLab to make our ISO publicly available. We do not ask those mirror admins for actual numbers of downloads; this may sound strange, but numbers of downloads and numbers of members aren't our priority. We are a group of enthusiasts who like to share our work with the community, just for the fun of it, and also for the interaction with the Linux and FOSS community.

Our experience so far is mainly very warm and positive. Prior to this distro, none of us had any experience in creating and running a project like this, and we are learning every day. This community is a beautiful example of how open, inventive, and friendly the FOSS community can be.

LM: Why is Endeavour described as "terminal centric"?

BP: It was actually born out of necessity. Because of the Calamares installer, some users expected an out-of-the-box experience after installation with a GUI software center, despite the info on our website. So after our second release, we decided to add this tagline, just to make the nature of the distro clear.

LM: What are some of EndeavourOS's unique features?

BP: The Welcome app (Figure 2) is the most distinctive app from us, likely because you will see it by default. Welcome is meant to provide a reasonable amount of information about EndeavourOS for users that are using it for the first time. Later it has grown to include many more features that we think are beneficial for the users.

Then there are other smaller apps (listed in no particular order):

- eos-apps-info: an easy-to-use documentation about EndeavourOS apps (with a GUI)
- akm: a small kernel manager (with a GUI)
- eos-log-tool: for troubleshooting potential problems (with a GUI)
- eos-pacdiff: a small script to make using `pacdiff` easier on all desktops
- nvidia-installer-dkms: for installing drivers for NVidia cards (this originated from Antergos)
- paccache-service-manager: a small GUI app to configure package cache management
- reflector-simple: a small GUI wrapper for Reflector, a mirror management tool
- UpdateInTerminal: a package updater that also supports the Arch User Repository; it has special NVidia support, which helps detect potential problems from upstream updates in the Linux kernel packages on a machine with an Nvidia graphics card

and many smaller scripts to make users' lives a bit easier. Some of the apps have a simple GUI, and others are purely terminal apps. We have used a GUI for cases where a GUI seems to be of help to users. The Welcome app has most of the above apps integrated and usable with a simple click.

LM: What future directions are planned?

BP: To be honest, the project is developing a bit organically, moving further on how the circumstances are directing us. EndeavourOS is a labor of love in the end. Having said that, we do have plans to enhance the ARM installation experience, perhaps by including the installation scripts on our ISO, and we do have plans to improve the way we are going to release the ISOs, but that is in a very early stage, and I can't reveal too much of it at this moment. ■■■

Info

[1] EndeavourOS: *https://endeavouros.com/*

[2] EndeavourOS Open Collective page: *https://opencollective.com/endeavouros*

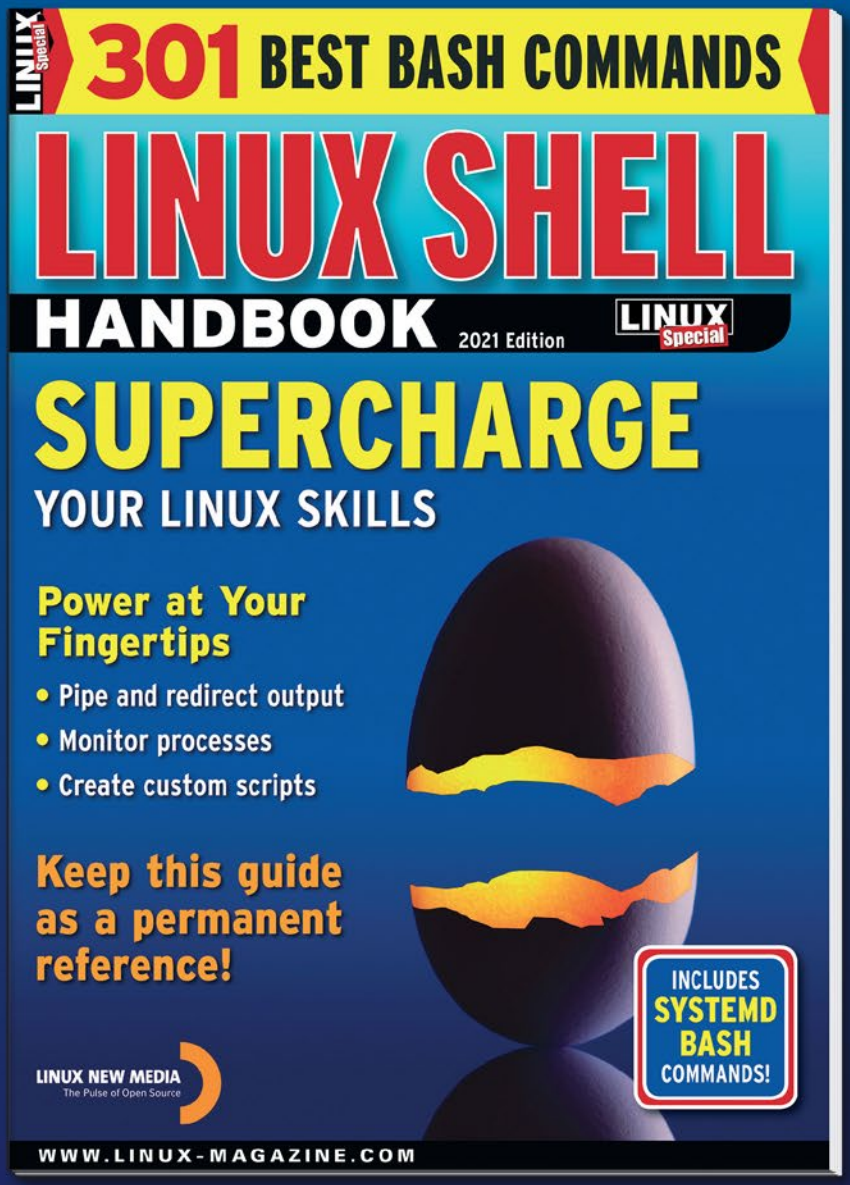
Author

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/*.

THINK LIKE THE EXPERTS

Linux Shell Handbook 2021 Edition

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



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

- Customizing Bash
- Regular Expressions
- Systemd
- Bash Scripting
- Networking Tools
- And much more!

ORDER ONLINE:

shop.linuxnewmedia.com/specials

Turning old hardware into network storage

NAS for the Masses

EasyNAS lets you set up network-attached storage in next to no time – even on old hardware. *By Erik Bärwaldt*

Do you have an old computer gathering dust? Thanks to Linux's flexibility and the innovative spirit of programmers, even computers that are more than 10 years old can still be used as network storage in current IT infrastructures.

Systems frequently used by power users in the past often offer plenty of space for mass storage. In addition, these systems usually have sufficient power supplies to run multiple hard disks as network-attached storage (NAS). EasyNAS, an openSUSE derivative, takes the pain out of configuring network storage with multiple storage media and integrating it into an existing LAN [1].

Requirements

EasyNAS comes as a hybrid ISO image with a size of about 380MB. As the minimum system requirements, the developers recommend a processor with at least 500MHz clock speed – as you may recall, this was state of the art more than 15 years ago.

A word of caution: Contrary to the original, not-yet-updated information about the hardware requirements, the current version, EasyNAS 1.0.0, requires a 64-bit processor. However, the older version 0.7.2 images available on SourceForge can also be used with 32-bit CPUs [2].

As a further requirement, the computer should have at least 1GB RAM, but 2GB or more is preferable for the system to run smoothly. The hard disk needs a capacity of at least 20GB.

The EasyNAS project does not state any further system requirements, but you will require a wired network connection for obvious reasons. EasyNAS lacks the ability to integrate with a network via WiFi. On a positive note, however, the system supports various transfer protocols. Network File System (NFS) can be used in Linux-only environments, while SMB or Samba is recommended for mixed environments.

Note that EasyNAS requires a dedicated storage area for itself. This is usually the first internal hard disk, which cannot be used for storage purposes as part of a network drive, either. Therefore, you need at least two physical mass storage devices to use the NAS system – or perhaps more depending on the desired RAID level.

Commissioning

On booting, the system first opens a GRUB screen where you can choose between installing EasyNAS or booting from the internal hard disk. The system does not support live operation.

To install EasyNAS on a mass storage device, the openSUSE derivative boots into an ncurses-based wizard. In the wizard, first select the disk on which you want to install the operating system. The installer then deletes all data on the disk, after giving you a fair warning, and does a disk check. Finally, the actual system installation takes place. The wizard then branches to a menu on the text console where you can access various administrative tasks by typing the listed numbers.

EasyNAS also enables a web server during the install, which means you complete the configuration using any web browser. To do this, call up the system's IP address in the browser, followed by a colon and the port number 1443 (e.g., *192.168.0.10:1443*). This will take you to a login screen where – if you keep the administrator's credentials – you need to enter *admin* as the username and password at the prompt.

EasyNAS then opens a visually appealing and clearly arranged dashboard with

Photo by Duy Pham on Unsplash

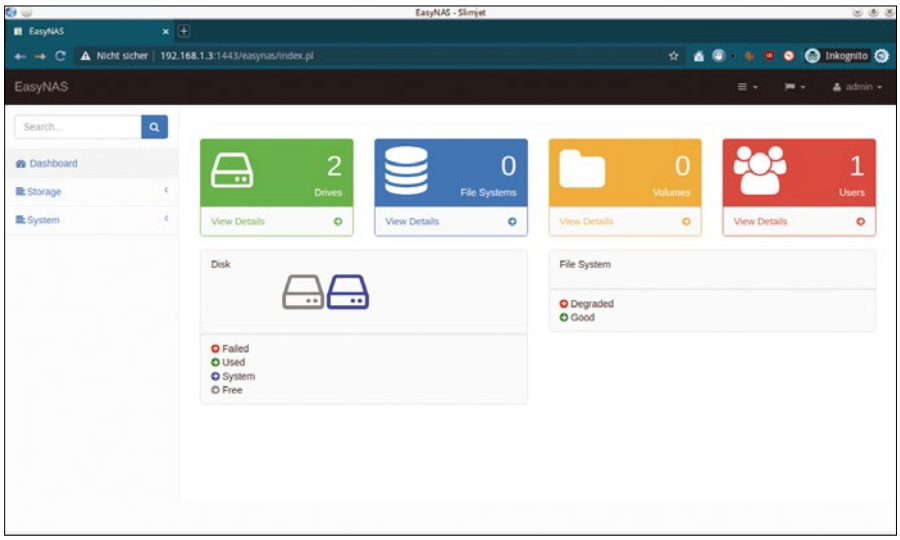


Figure 1: The EasyNAS dashboard clearly displays the NAS system's configuration.

icons that visualize the NAS system's technical configuration (Figure 1).

Dashboard

Clicking on *Drives* in the dashboard lets you manage the mass storage devices that physically exist on the system. *Drives* also takes into account removable media, such as USB memory sticks, but not optical drives. Existing mass storage devices are displayed as symbols. Note that the EasyNAS system drive always appears in blue, and network drives are shown in green.

Next to *Drives*, *File Systems* summarizes the existing filesystems used for the NAS. To the right, *Volumes* contains logical drives, and *Users* displays the users logged into the system on the far right. To find out more information for each group, click on *View Details*.

After the install, *File Systems* and *Volumes* will be empty because you need to configure them first. Also, *Users* only contains the administrator.

On the left side of the dashboard, in the vertical sidebar, you will find *Storage* and *System*, which you will use to configure your new EasyNAS.

Configuration

To create a filesystem on the drives intended for the NAS as mass storage, first open *Storage* | *File System Manager*. After clicking on *Create File System*, a clear-cut dialog appears where you can assign a name to the filesystem, select the appropriate drive from a list of mass storage devices built into the system, and specify which RAID level you want to use for creating the drive (Figure 2).

In addition, you can set a compression level here and define how the NAS mounts the drive. It is important to note that when specifying the RAID level, if there is only one storage drive in the system, you must specify the *JBOD* level. Also, the filesystem name should be at least eight characters. To apply the settings, click *Save*. The system then confirms successful creation of a new filesystem.

Next, switch to *Storage* | *Volume Manager* and create a new volume. After clicking on *Create Volume*, a self-explanatory dialog appears that is limited to essential information. Because no users or groups exist initially, the administrator still appears in this dialog as the *User Owner*. If you created several filesystems in the first step, then specify the filesystem to be used for the volume in a selection field in this dialog. You also need to name the volume and, if applicable, assign permissions to manage access to it.

The basic setting activates read and write permissions for users, and the group only has read permission. If you create quotas, you also conveniently assign storage quotas in this dialog. After completing all the settings, press *Save* to save the new volume. Similar to creating a filesystem, the volume now appears in a list in the *Volume Manager*, which you can use to modify the settings later on if needed (Figure 3).

Groups and Users

To use the memory quota sensibly and securely, you must create groups and users using *System* | *Groups Manager* and *Users Manager* respectively. Depending on which option you select, you will find the *Create Group* or *Create User* links, much like the drive settings. I recommend creating the groups first and then assigning the users to the groups by selecting the desired group in the corresponding selection field in the *Create User* dialog.

When creating a user, you can also define quotas and assign permissions. After completing the settings, press *Add* to create the group or user. The dashboard will then display the total number of logged-in users in *Users*. Clicking *View Details* under *Users* opens detailed information about the users in a table format. Under Actions, you can manage

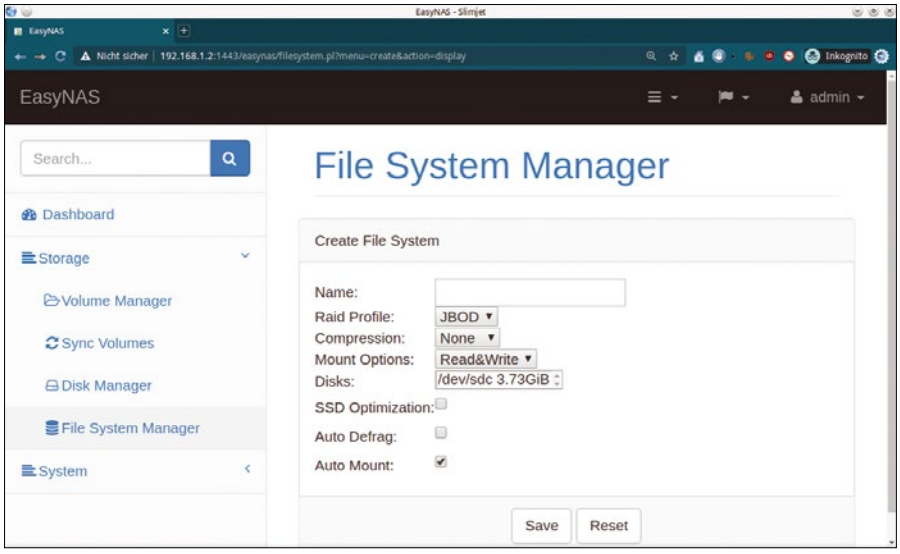


Figure 2: Creating new drives in the File System Manager.

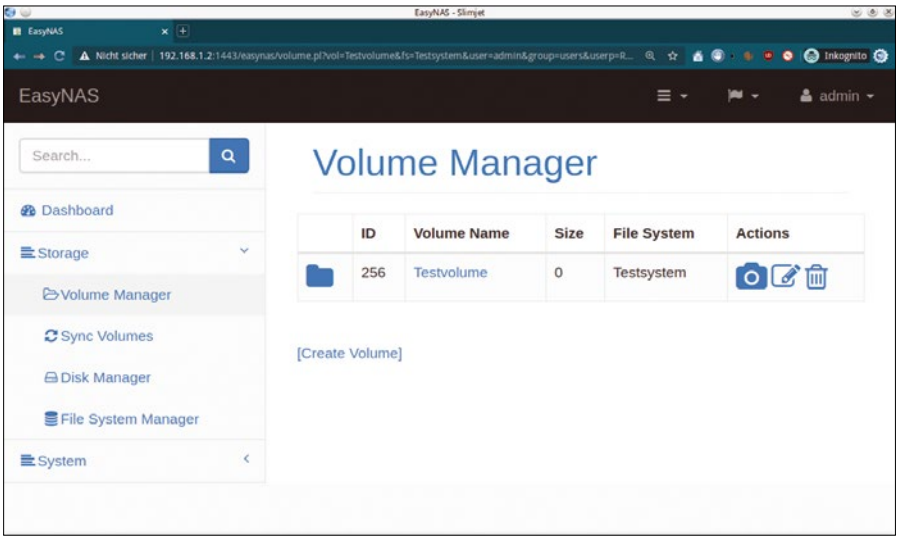


Figure 3: EasyNAS uses the same dialog design for all categories, like the dialog shown here for creating a volume.

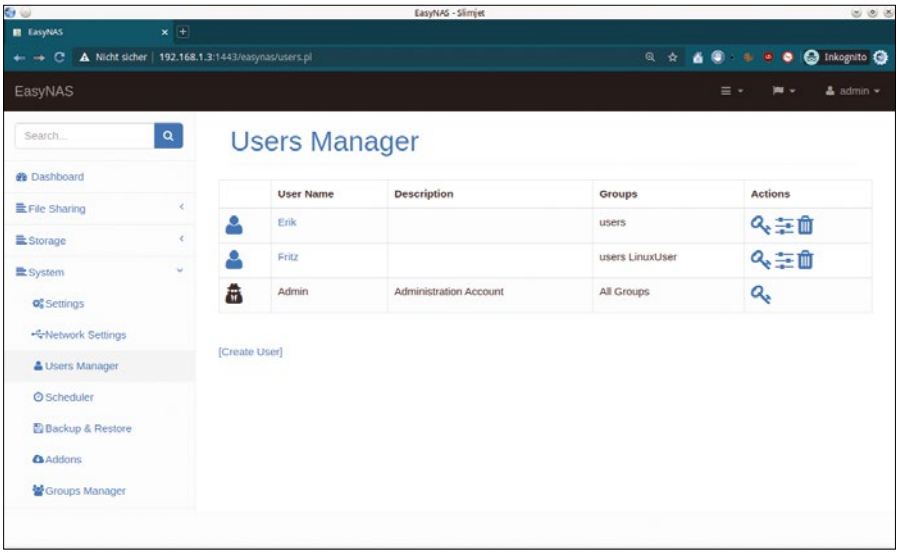


Figure 4: EasyNAS comes with a small but workable user management feature.

the entries, delete users, or modify their settings (Figure 4).

Access

To gain access to the NAS system's resources, the final step is to specify which access protocols you want to use. EasyNAS offers a total of eight different access modes, ranging from Apple's AFP protocol to CIFS/SMB, which is common in the Windows world. The SSH, NFS, and FTP protocols commonly used on Linux are also available.

You can enable support for the individual protocols under *System* | *Addons*, where you will find the individual options listed in a table in the *File Sharing* area. To integrate an option into the system, press the installation button on the right in the Actions column. EasyNAS now downloads the corresponding package from the Internet, integrates it, and then displays a confirmation message.

To access the NAS system using the respective protocol, you need to enable the protocol first. To do this, EasyNAS creates a new *File Sharing* category in the sidebar on the left, which shows the currently installed protocols. You can enable them via the sliders. The protocols are then immediately available without rebooting the system.

To access the individual volumes from a desktop PC, open the respective desktop environment's file manager and connect to the desired storage via the corresponding settings option (Figure 5). The volumes can then be used like any conventional folder.

Weaknesses

In EasyNAS v1.0.0, you'll find the *Scheduler* and *Backup & Restore* options in sidebar on the left. These options are designed to allow the use of Btrfs snapshots in addition to backing up and restoring data. Currently, *Scheduler* and *Backup & Restore* are under development and cannot be used. For a backup, you still must manually create a complete copy of the data files, for example, via the file manager on a client computer.

Conclusions

EasyNAS gives you a network storage solution that does exactly what it's supposed to do without any frills – store and manage your data on a NAS system. EasyNAS's visually and ergonomically consistent user interface eliminates the need for time-consuming familiarization. Thanks to the openSUSE base and the Btrfs filesystem, EasyNAS's basic functionality can be expected to grow in future versions.

EasyNAS also stands out because of its frugal hardware requirements. Even ancient PCs can easily be converted into a NAS storage system, making EasyNAS suitable for home users without special knowledge who want to quickly set up and use network storage. ■■■

Info

[1] EasyNAS: *https://easynas.org*
[2] EasyNAS images on SourceForge: *https://sourceforge.net/projects/easynas/files/*

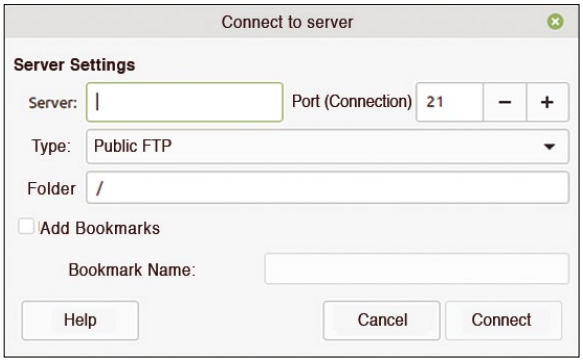
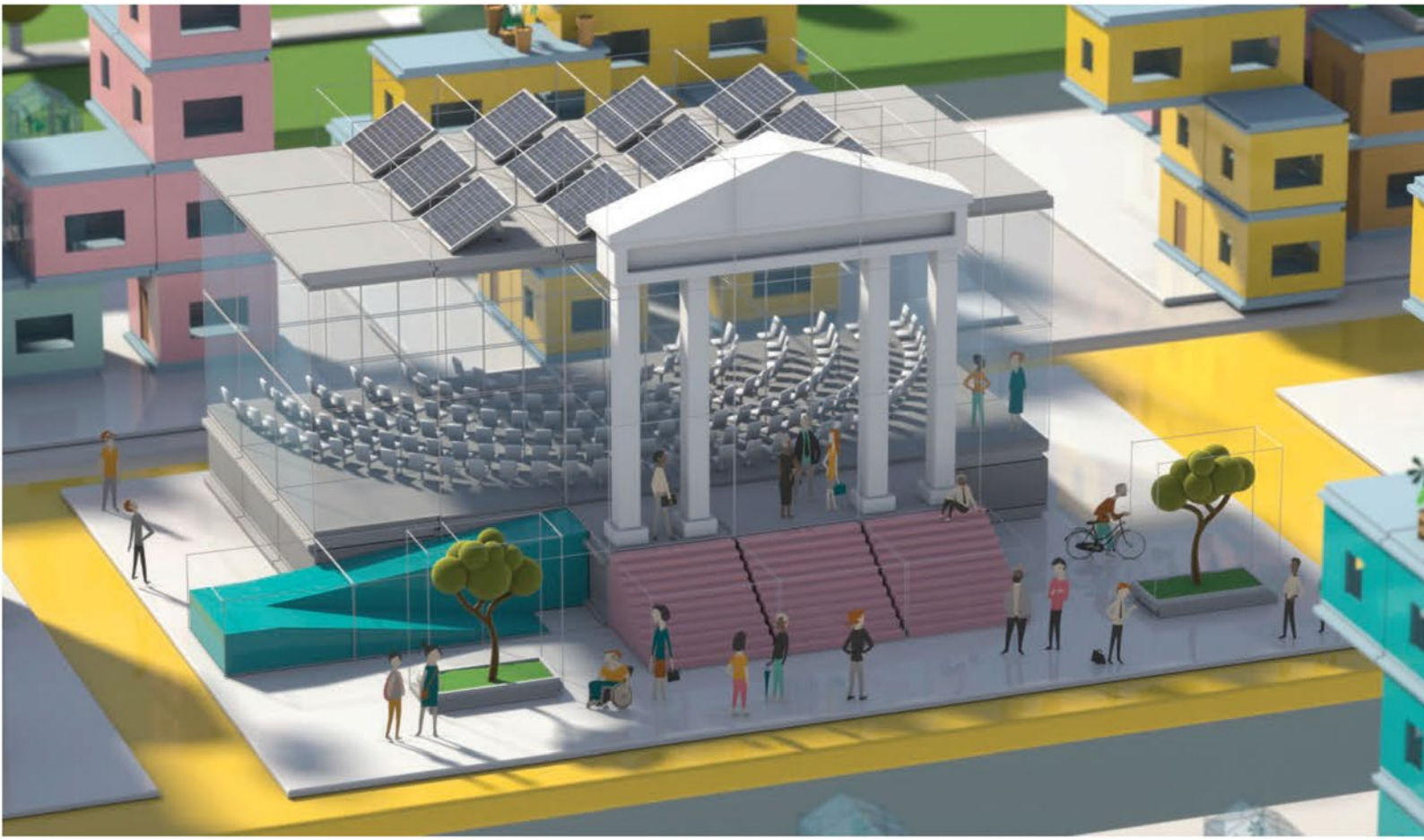


Figure 5: The common file managers offer simple dialogs to connect to network storage.

Public Money
Public Code



Modernising Public Infrastructure with Free Software

fsfe Free Software Foundation Europe

Learn More: https://publiccode.eu/

A modern search tool

Location, location, location

As the latest successor to locate, plocate produces some of the quickest search results possible on any system. *By Bruce Byfield*

Real estate agents sometimes say that the key to success is location, location, location. This saying might almost summarize the history of the `locate` command [1] and its various successors, particularly `mlocate` [2] and `plocate` [3]. As a replacement for `find`, all three commands use a database solution to reduce search time. While all three share many of the same options, `plocate` is widely considered the most efficient choice.

Slightly different versions of `locate` are available in the BSD and GNU *findutils*, but you can also find `locate`, `mlocate`, and `plocate` as separate commands. Because the number of choices can be confusing, a history seems in order. First released in 1982, `locate` uses a database that can be read by any user. If regular expressions are not used, it displays every instance of the string entered on the system, which is inconvenient if the string is common. However, `locate`'s most serious limitation is that the database has to be updated manually. Largely because of these problems – especially the need for manual updates – `locate` was succeeded briefly by `slocate` (secure locate) [4] until replaced in 2006 by `mlocate` (merging locate). Both `slocate` and `mlocate` are an improvement over `locate` in that they contain the utility `updatedb` [5] to update databases, speeding up the process by only searching for files and directories where the `ctime` has changed. Also, both `slocate` and `mlocate` show only the files that the current user has access to, thereby improving security, and they allow regular expressions to be used without a specific option. Written by Miloslav Trmač while he participated in the Google Summer of Code, `mlocate` became the preferred version until 2020, although it seems to have gone through periods of being unmaintained.

Named for the posting lists that inspired it, `plocate` was written to be a drop-in replacement for `mlocate`. While it can still use `updatedb` to create its database, `plocate` can also use the `plocate-build` utility [6] to create an index when a root user is logged in. Unlike `mlocate`, when multiple strings are searched, `plocate` returns only the files that match all the search strings, rather than any file that matches even one string. Another difference from `mlocate`: `plocate` is compatible with systemd and SELinux. Instead of scanning every entry one at a time, `plocate` scans trigrams (i.e., combinations of three bytes at a time) to increase search speeds. Although specifically designed for solid state drives (SSDs), on older hard drives, `plocate` can gain further speed by using the `io_uring` Asynchronous I/O (AIO) framework introduced in the Linux 5.1 kernel in March 2019 [7]. As its main limitation, `plocate`'s benefits may be lost when searching for strings shorter than three bytes, for non-UTF-8 file names, or for regular expressions with numerous hits. Usually, its enhancements mean that `plocate` can find two files out of 27 million in .008 milliseconds, while `mlocate` takes 20.118 milliseconds for the same operation [8]. Even though `locate` and `mlocate` are still available, in most circumstances, `plocate` should be the preferred variant.

Setting Up plocate

You can find `plocate` in the official repositories of Arch Linux, Manjaro, Debian (Buster backports and Bullseye releases), and Ubuntu. On other distributions, you can build `plocate` with zstd and `libtomic` using a C++ compiler. Whichever way you install it, before you use `plocate`, you must run `updatedb` to create at least one

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/*.

database at `/var/lib/mlocate/mlocate.db` (Figure 1). This process may take a while to run the first time if the system is not on an SSD drive and does not run at least a Linux 5.1 kernel. Otherwise, on a 2TB system, the database is created in a matter of a few seconds. To specify a non-default location for the database, use: `--database-root FILE`. You can add `--database-root FILE` (`-U FILE`) to specify a non-default location for the database, and `--require-visibility 0` (`-l 0`) to make the entire content of the database visible to ordinary users. Adding `--verbose` (`-v`) will display the files and directories being added to the database onscreen, which has the advantage of letting you know when the database creation is complete. Alternatively, you can create the database using `plocate-build`, even creating a plain text database if you choose. Either way, you can check that `plocate` is ready to use by doing a simple search (Figure 2).

Of note, `mlocate` includes a few options that `plocate` lacks. For example, `mlocate` includes `--statistics` (`-S`) as well as `--stdio` (`-s`), a C function for interacting with different physical devices, for compatibility with the BSD and GNU versions of `locate`. What remains in `plocate` are a dozen basic functions, which are more than enough for almost all operations.

On networks, `plocate` can be modified by adding multiple databases with `--database PATH` (`-d PATH`). Multiple databases can be specified one per option or in a single option with the names of databases separated by a colon (`:`). Note that versions of Ubuntu before 21.04 may not have these options and may not mention them in their man pages. Two other options signal that a search string should be treated as containing regular expressions: `--regexp` (`-r`) signals the presence of standard regular expressions and `--regex` signals that extended regular expressions are used. Because regular expressions can return a large number of results, these options can noticeably slow a search. Without one of these options, common symbols in regular expressions such as *, ?, or ! are treated as regular text.

Other options modify a search's results. With `--ignore-case` (`-i`), `plocate`'s default case sensitivity is overwritten, so that lowercase and uppercase letters are treated as the same, except in the case of some Unicode case rules, such as a German *ß* being the same as *ss*. Moreover, like regular expressions, ignoring letter case can slow the search by producing a larger number of results, but search speed can be increased. For instance, on one hand `--limit NUMBER` (`-l NUMBER`) stops a search after the designated number of hits, and `--count` (`-c`) merely shows the number of instances. On the other hand, `--basename` (`-b`), which searches only for file names without their extensions, increases search speed only minimally at best by omitting directories from the search.

A Command for Modern Times

With the recent release of `plocate`, the `locate` family of commands seems set for the next few years. While `plocate` has a few limitations, it uses the latest technology and is adopted to modern computing practices. For example, with the current hardware speeds, it no longer makes sense to set whether to follow symbolic links – following them by default can be done with no noticeable effects.

Even more important, the rise of Linux on the desktop has increased the demand for simplicity. In the past, the `find` command, with its arcane distinctions, was sufficient because users could be assumed to have the expertise and the patience to wade through the obscurities of its man page. Today, though, with less veteran users, simplicity and efficiency are expected as the norms. In the case of system searches, these expectations have resulted in `plocate`, one of the increasing number of rewrites of classic Linux commands that users have lived with for so long. ■■■

Info

[1] locate: *https://www.man7.org/linux/man-pages/man1/locate.1.html*
[2] mlocate: *https://www.unix.com/man-page/Linux/1/mlocate/*
[3] plocate: *https://www.mankier.com/1/plocate*
[4] slocate: *https://linux.die.net/man/1/slocate*
[5] updatedb: *https://www.unix.com/man-page/linux/8/updatedb/*
[6] plocate-build: *https://plocate.sesse.net/plocate-build.8.html*
[7] io-uring AIO framework: *https://blogs.oracle.com/linux/post/an-introduction-to-the-io-uring-asynchronous-io-framework*
[8] plocate's speed: *https://www.linuxuprising.com/2021/09/plocate-is-much-faster-locate-drop-in.html*

```
/var/tmp/kdecache-bb/plasma-wallpapers/home/bb/art-wallpaper.png_#3e9fba_1_1920x1200.png
/var/tmp/kdecache-bb/plasma-wallpapers/home/bb/art-wallpaper.png_#000000_2_1920x1200.png
/var/tmp/kdecache-bb/plasma-wallpapers/home/bb/art-wallpaper.png_#000000_2_1920x1080.png
/var/tmp/kdecache-bb/plasma-wallpapers/home/bb/Pictures/Art
/var/tmp/kdecache-bb/plasma-wallpapers/home/bb/Pictures/Sun-Yat-Sen-Gardens-April-2005
/var/tmp/kdecache-bb/plasma-wallpapers/home/bb/Pictures/Bruce2016
/var/tmp/kdecache-bb/plasma-wallpapers/home/bb/Pictures/Art/carol-young-bagshaw
/var/tmp/kdecache-bb/plasma-wallpapers/home/bb/Pictures/Art/Cody-LeCoy
/var/tmp/kdecache-bb/plasma-wallpapers/home/bb/Pictures/Art/Cody-LeCoy/Mousewoman.JPG_#000000_0_
1024x768.png
/var/tmp/kdecache-bb/plasma-wallpapers/home/bb/Pictures/Art/Cody-LeCoy/Mousewoman.JPG_#000000_0_
1920x1200.png
```

Figure 1: Entering `updatedb -v` shows files and directories being added to the `plocate` database.

```
/home/bb/.local/share/okular/docdata/341908.000000fe.000-000_Distro_Walk_Qubes_av_lw_ap.pdf.xml
/home/bb/.local/share/okular/docdata/341908.00000202.000-000_Distro_Walk_Qubes_av_lw_ap.pdf.xml
/home/bb/.local/share/okular/docdata/341908.00000204.000-000_Distro_Walk_Qubes_av_lw_ap.pdf.xml
/home/bb/Downloads/Qubes-R4.0.4-x86_64(1).iso
/home/bb/Downloads/Qubes-R4.0.4-x86_64.iso
/home/bb/work/journalism/2021/10-october/Qubes
/home/bb/work/journalism/2021/10-october/Qubes/Qubes.tar.gz
/home/bb/work/journalism/2021/10-october/Qubes/copying.jpg
/home/bb/work/journalism/2021/10-october/Qubes/copying.png
/home/bb/work/journalism/2021/10-october/Qubes/desktop.png
/home/bb/work/journalism/2021/10-october/Qubes/domains-on-the-menu.png
/home/bb/work/journalism/2021/10-october/Qubes/menu.jpeg
(END)
```

Figure 2: Because `plocate` is so fast, its output starts to display before the piped `less` command can operate.

TDE: Museum Piece or Modern Desktop Alternative?

Exploring the Trinity Desktop

Trinity Desktop Environment's latest release may seem like a trip back in time due to archaic features, but it does offer a functional desktop for modern computing and, more importantly, freedom of choice. *By Bruce Byfield*

In the early years of this century, Gnome 2.0 and KDE 3.0 were Linux's most popular desktops. Sometimes one won the popularity polls, sometimes the other. Eventually, both were replaced by radically new releases, but both refused to disappear. In the case of KDE 3.0, a fork was created called Trinity Desktop Environment [1] (TDE, or simply Trinity). More than 10 years later, Trinity continues to be kept alive by a small group of developers, mostly in once- or twice-yearly maintenance releases, with the latest R14.0.11 release in October 2021. The result is a functional desktop, still suitable for modern computing, but with some archaic features and limitations that users may want to update immediately after installation.

Trinity is included in few distributions, so the project cannot depend on maintainers from outside the project. Instead, the project includes downloads for images of several distributions configured to install Trinity. These include a number of images for Debian, Devuan, Red Hat, and Fedora. For convenience, the Long-Term Support (LTS) releases, such as Ubuntu 20.04 (Focal Fossa), are probably the ones to choose because newer releases may not support Trinity. In some cases, unofficial or testing versions may also be available. Yet another alternative

Figure 1: Trinity's first boot starts with KPersonalizer, a useful configuration wizard.

Photo by Joshua Earle on Unsplash

is to install from source tarballs, which is the only choice for adding Trinity to an existing installation. In each case, consult the online documentation about your distribution of choice to avoid minor, irritating glitches, especially when attempting to upgrade. Note that after 2013 and release 3.5.13.2, Trinity switched its numbering system to avoid being dismissed because of its version number alone. Releases after that began with 14.0.0. To date, there has been no other major release.

The Face of the Future or the Past?

Unless you install from source files, the installer for Trinity depends on the distribution used. Regardless of the chosen distribution, Trinity opens for the first time with KPersonalizer, a configuration guide that would improve most other desktop environments. Besides routine configuration such as the theme, country, and language (which, for some reason, defaults to French for Canada), KPersonalizer includes a choice of general behavior (Figure 1) as well as the Eyecandy-o-meter, which sets the level of special effects to use. Unless you dislike special effects, you can probably set the meter to maximum on a modern computer, although turning the effects off may help reduce problems or make Trinity run better on an older computer. By default, later logins start with a tip for users, although this feature can be turned off.

Once Trinity is running, veteran Linux users could be excused for thinking they have traveled in time. Little has changed visibly since KDE 3.5 was in its heyday (Figure 2). The default themes, for instance, are unchanged from the plastic-toy look that put some people off the whole idea of KDE 4.0. Even the version of the Qt widgets has not changed, although these days the project has to maintain them itself. More importantly, Trinity is like OpenOffice, with the changes in the latest release suggesting a relatively small team. The latest applications are mostly modest: A ported theme and an ASCII art aquarium screensaver are listed among them. Similarly, the enhancements listed are minor ones, such as variable font resolutions, several translation updates, and various cosmetic improvements.

From a modern standpoint, the results are mixed. On the one hand, with its long history of development, Trinity includes clear, concise help and an extremely complete Control Center (Figure 3). Besides the settings for the usual system and external devices settings, the Control Center also includes an extensive TDE Component section, with settings for default applications, file associations, the spellchecker, and other choices for system performance. These settings, I suspect, should be the starting points for troubleshooting. Setting which services are started at login and which on demand, for example, could easily eliminate some problems. Conversely, disabling the standard system check on startup, which reduces boot time, could also conceivably create problems in some cases.

On the other hand, TDE is clearly from the days prior to KDE's Visual Design Group, which has done so much in KDE/Plasma recent releases to give the desktop a professional polish. Nothing is absolutely wrong with the desktop, but little clues like the cluttered, sometimes

indecipherable icons in the notification bar or the fixed text size on icons are a reminder that Trinity dates from a time when functionality and catching up with proprietary desktops were more of a priority than the user experience. In fact, this success in achieving functionality is one of the reasons that later releases could concentrate on user experience. Still, from today's standpoint, the desktop seems a little rough and ready.

An apparent effort to preserve KDE 3.5 with a minimum of change does not reduce Trinity's mildly archaic impression. For instance, one of the changes in the latest release is to change the name of Bookreader to KBookreader, keeping alive KDE's irritating and rather pointless custom of beginning every application name with a k. Even more obviously, Trinity persists in using only KDE-based apps by default. Instead of Firefox or Chrome, Trinity installs with Konqueror as the browser. Similarly, KOffice is installed rather than LibreOffice. These substitutions would not matter, except that they are inferior to the applications that are commonly used today. In the case of the Amarok music player, version 1.4.10 is used, when the latest version is 2.9.71. Still others apps such as Gimp have no substitute at all. Many modern apps can be installed from the repositories of the distros that host Trinity, yet there is always the possibility of incompatible libraries. In addition, these apps are not always added to the menu. In the end, I have to wonder: Is the point of Trinity to provide a serviceable desktop or to be a museum record of the past?

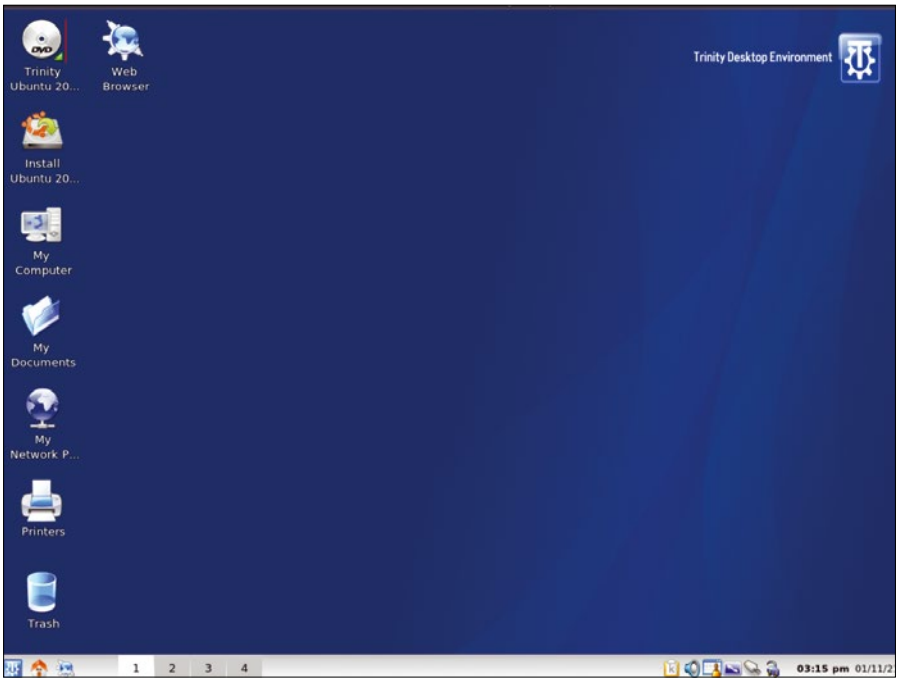


Figure 2: Trinity's latest 2021 release still looks much like KDE 3.5 did in 2008.

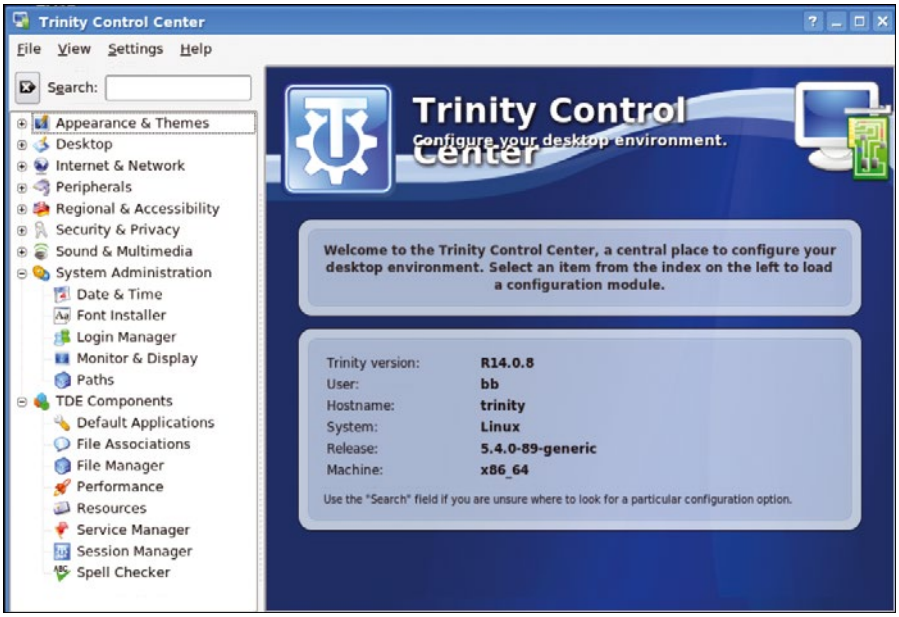


Figure 3: Trinity inherits a detailed Control Center.

The Reason for It All

Compared to Gnome 2, KDE 3 has fared poorly since it was superseded. Gnome 2 was ported to Linux Mint and became MATE, which has always balanced tradition and modernization. In 2020, the LinuxQuestions poll [2] listed MATE as the choice of 8.5 percent of users, and I suspect that a much higher percentage would list it as their second choice (the desktop used when your main one fails). By contrast, Trinity claimed only a 1.3 percent share.

So why bother with such a minority choice? To start with, Trinity scored the same as Ubuntu's defunct Unity, while elementary OS's Pantheon desktop, which receives frequent coverage, received no votes at all. The point is not user share but freedom of choice, a key value of free software. Trinity's success lies in having fought the odds to provide a small group of users what it wants and in extending the choices for everyone. ■■■

Info

[1] Trinity Desktop Environment: *https://www.trinitydesktop.org/*

[2] LinuxQuestions poll: *https://www.linuxquestions.org/questions/2020-linuxquestions-org-members-choice-awards-131/desktop-environment-of-the-year-4175687348/*

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/*.

The sys admin's daily grind: ASN

Highway Through Hell

When digging into BGP routing information, Charly avoids the highway through parameter hell thanks to the ASN tool. In addition to a system's AS number, ASN delivers other information, such as its peering partners upstream and downstream. *By Charly Kühnast*

Every admin knows how to deal with IP addresses. Unfortunately, IPs never turn up alone. They belong to a network, and the network is almost always assigned to an autonomous system (AS), which uses the Border Gateway Protocol (BGP) to exchange routing information.

There is a simple way to find out which AS a particular individual IP belongs to. By way of an example, the following is the IP address of Computec Media's web server, *www.computec.de*. The associated IP address *62.146.104.133* can be discovered using `dig` or by simply pinging.

```
62.146.104.133 ┌PTR  -
               ├ASN  15598 (IPX-AS15598, DE)
               ├ORG  Computec Media AG
               ├NET  62.146.0.0/16 (COMPUTEC-MEDIA)
               ├ABU  abuse@ext-dc.de
               ├ROA  ✓ VALID (1 ROA found)
               ├GEO  Furth, Bavaria (DE)
               └REP  ✓ NONE
```

Figure 1: A first call reveals the ASN, among other things.

```
Peering informations for AS15598 (IPX-AS15598, DE)

──────────── Upstream Peers ────────────

CORE-BACKBONE (33891) | RETN-AS (9002) | LEVEL3 (3356) | HURRICANE (6939)
PLUSNET (20676) | LINK11 (34309) | MEER-AS (34549) | ZAYO-6461 (6461)
DTAG (3320) | TKSWF-AS (48362) | GREEN (1836) | FIBERTELECOM-AS (41327)
IFOG-GMBH (34927) | WEBERCLOUD (35710) | SEABRAS-1 (13786) | OPENFACTORY-AS (58299)
IP-MAX (25091) | RASCOM-AS (20764) | ESAB-AS (39351) | HIVANE (34019)
and more: 198385 13030 58057 7195 8758 35280 24961 48858 200780 42708 29076 8218 43350
0844 15412 9050 5713 1267 9044 20485 1239 328832 20562 8708 6695 30781 8359 8220 8732
 35320

──────────── Downstream Peers ────────────

INACDE-ATOS (21150) | BIGPOINT (28902) | RAY-AS (35003) | SWERK-GERMANY (42840)
DATEV-AS (15451) | RED (47897) | IP-EXCHANGE2 (24898) | KASI-ASN (24836)
UDAG (202108) | INTERHYP-AGAS (60845) | ATOS-DE-NUREMBERG (8735) | PHOENIX-AS (34460)
WEBTREKK-AS (60164)
```

Figure 2: Calling the tool with the ASN reveals a plethora of information, including …

```
AS path to 62.146.104.133

 60294  DE-DGW Deutsche Glasfaser Wholesale Internet (Local AS)
 ┌IXP   DE-CIX Frankfurt ()
 └15598 IPX-AS15598
```

Figure 3: … the routes the packets take from the home AS to the target.

I then feed this IP address to a tool named ASN [1]. The shell script aggregates the output of several other tools and presents the results in a clear-cut way. It has a number of dependencies that vary depending on the distribution you are using. What exactly needs to be installed for ASN to work is explained in a separate section on the tool's GitHub page.

The output from calling `asn 62.146.104.133` reveals that the IP address in question belongs to network *62.146.0.0/16* and *AS15598* (Figure 1). AS numbers (ASN) are unique worldwide and are assigned by the Internet Assigned Numbers Authority (IANA).

A second call with the ASN as a parameter (`asn AS15598`) provides a variety of information about the AS, such as the upstream and downstream peering partners (Figure 2). In addition, you can see which other networks are assigned to this AS besides the already known *62.146.0.0/16* and which paths the data packets take from the local AS to the destination (Figure 3).

All this information could be obtained in other ways, but thanks to ASN, I was able to save time and avoid a detour through parameter hell. ■■■

Info

[1] ASN: *https://github.com/nitefood/asn*

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.

Merging file systems for a simple NAS with MergerFS

Come Together

MergerFS is a simple tool for bunching together disks, volumes, and arrays. *By Adam Dix*

I had to make many decisions when setting up my personal network-attached storage box. I needed a machine capable of sharing files on my local network with Samba [1]. I also wanted to be able to use the system as a Plex streaming server [2] and to run virtual machines occasionally to test out new Linux distributions. I didn't need the system to be mission critical or high performing. A big motivation for setting this server up was to learn more about Linux. With that in mind, it should not be too surprising that I built it using spare parts.

The files I wanted to store on this server were mainly replaceable media files. A high-end file system such as ZFS sounded amazing, but it was more than I needed in this case, and ZFS wasn't really financially viable because of RAM costs. I just wanted to get the most mileage out of my hard disk space, redundancy be damned. All critical information, such as personal files, would be backed up to multiple machines and to someone else's computer (Alphabet's, to be precise).

I did plan on using a RAID-0 array – for speed rather than redundancy. Using a Plex and Samba server on a home network meant that the bulk of the data would be written once and read occasionally, and that speeds of even shingled magnetic recording-based spinning drives would be more than adequate. However, one issue was the need to support Windows and the desire to format the drives to NTFS so that, in the event of a hardware failure or operator error, the drive could be removed and installed into a 3.5-inch external enclosure on a nearby Windows system.

My frustration began when I was trying to set up Samba and Plex in a way that would make logical sense to the person accessing the files on the opposite end. The easiest way to do this is to have one share that represents the contents of the entire machine, but the problem with that approach is that those files and folders are stretched across multiple disks and in no particular order, and to add chaos to confusion, not all drives are even in the same format. One disk might have TV shows, another movies, a third software, and a fourth all of the above plus some documents. Music is striped across all of the disks.

RAID could have been a possible solution from the start, but the disks were acquired one at a time, with a new disk added only when the installed disks were approaching their capacity (> 80%). I had heard of Synology NAS machines using Btrfs in a way that allows the owner to install a disk of any size and to extend the array to the size of the current array plus the new disk, but again my disks were initially set up primarily with NTFS and there was not enough space on any one or two disks to allow for the creation of an array after the fact. Backing up all of the data to an online service would have been cost-prohibitive. It would be amazing if there existed a RAID technology that would allow me to keep the contents of the drives while creating the array, but that's the kind of magic reserved for unicorns in fairy tales.

The solution to this mess was MergerFS [3]. MergerFS is a tool that lets you combine separate file systems and volumes from different partitions or disks into one single volume facing the user. You can think of MergerFS as something similar to utilities such as mhddfs, UnionFS, and AuFS. MergerFS doesn't care if your drives are formatted to NTFS, FAT, ext3, or ext4, or if they are organized in a RAID-0, 1, 5, or another level of RAID. It doesn't even matter if your drives are in a logical volume already. MergerFS loves them all just the way they are.

MergerFS runs in userspace as a FUSE device, and it can be manipulated in most of the same ways that any other volume can be, but it is composed of partitions from other volumes or disks – with logic that makes it easy to configure how the user interacts with it. The GitHub page describes MergerFS as a "union of sets." You can configure policies defining how files and folders are added to the array.

MergerFS can handle both read/write and read-only disks. Redundancy isn't prohibited or disallowed, but you need to

consider redundancy when setting up the device initially. For instance, suppose you would like to have personal files duplicated (health records, CV, Bitcoin wallet, pictures), but you also have many files that do not need to be kept safely (audio and video media, a collection of Linux ISOs etc.). You could create a folder within a pre-allocated RAID array of say, RAID-5, and create a folder within it for personal files. Then you could create another set of folders on a separate volume that is not redundant, such as one disk for movies and another disk for TV shows and music. The RAID-5 array, along with the two media disks, can be "merged" into what appears to be a single client-facing drive. As long as personal files are added to the personal files folder (initially created on the RAID-5 array) and all media files are added to the folders that were created for them initially, the files will all end up in the right places. This behavior is, of course, based on the policy that the user chooses to set. More information on the varying policies is available on the GitHub page for MergerFS.

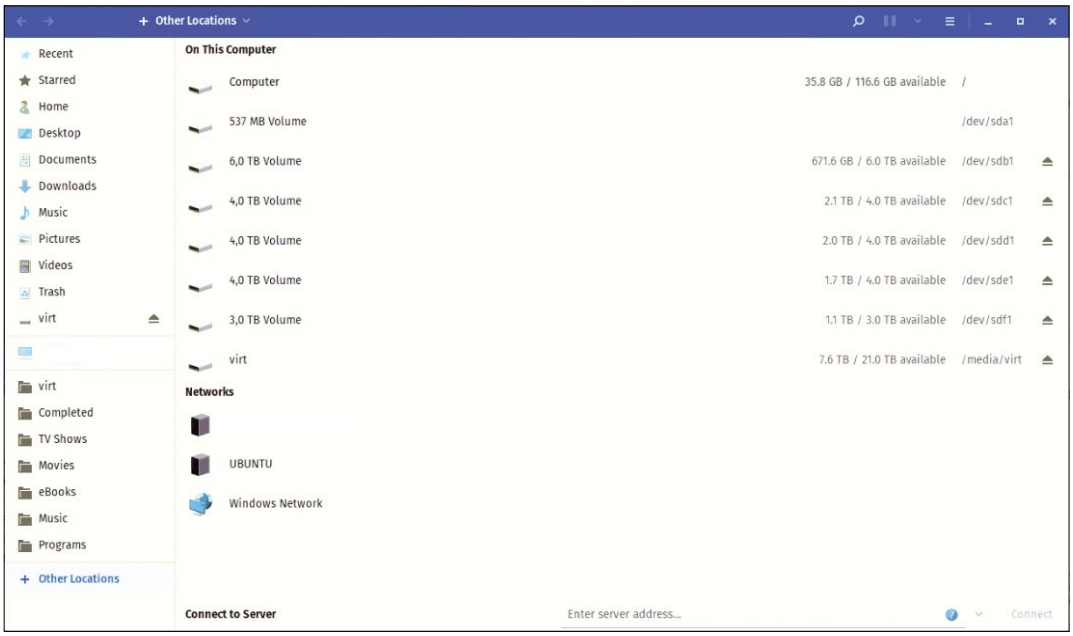


Figure 1: Virt created from disks b, c, d, and e.

If a disk in the RAID-5 array fails, then it can be replaced and the array rebuilt. On the other hand, if the disk with the movies, TV shows, or music fails, then the data is lost and will need to be recreated or downloaded again.

MergerFS allows for redundancy when needed, a lack of redundancy when redundancy is not needed, a single point of contact for the user, and a lower cost compared to setups where everything is redundant. You can even use MergerFS in conjunction with ZFS or Btrfs, and the benefits of those file systems are still achieved.

For my system, MergerFS has allowed me to take drives of different sizes, models, speeds, and even formats, and combine them in a way that makes it very simple for me to add, change, or remove data without worrying about what information is on the drive. In the future, I intend to invest in a much better system where data loss is minimized through using RAID or ZFS, but for now, MergerFS has allowed me to very easily and inexpensively manage a wide variety of data for my home server in a way that suits me and is easy to expand.

To set up MergerFS initially on Ubuntu, run the following commands:

```
$ sudo apt update $ sudo apt install
mergerfs -y
```

Next, you will need to create a folder to mount the MergerFS array (I have used `/media/virt` in the following example), apply the chosen policy, and add drives to the array. For my setup, I use the following:

```
mergerfs -o defaults,allow_other,↵
use_ino,nonempty,fsname=MergerFS ↵
/media/diskb:/media/diskc:↵
/media/diskd:/media/diske ↵
/media/virt
```

In this example, the user-facing volume containing the drive contents will be `/media/virt`, and it will consist of all of the data on `/media/diskb`, `c`, `d`, and `e`. Figure 1 shows how this appears in the file browser. The virt drive on my system can be used just as any other single drive would be. In each individual drive that makes up the array, I have separate folders for different data, such as programs, movies, and so forth, organized in a way that I set up initially. With this setup, if I have a folder named `Programs`, say, on both disk `b` and disk `c`, then MergerFS will add files to whichever drive has the most free space. If I only have a folder named `Programs` on disk `c` though, then it will automatically add anything that I put in the `/media/virt/Programs` folder into disk `c`.

Conclusion

MergerFS has made it much easier for me to organize my data in a logical way. I can simplify my configuration by setting up different folders for each type of data, and MergerFS figures out the file system and redundancy details. To complete the configuration, you can set up MergerFS to start when your system boots up. ■■■

Info

[1] Install and Configure Samba: *https://ubuntu.com/tutorials/install-and-configure-samba#1-overview*

[2] How to Install Plex on Ubuntu 20.04: *https://linuxhint.com/install_plex_ubuntu-2/*

[3] MergerFS GitHub: *https://github.com/trapexit/mergerfs*

Author

Adam Dix is a mechanical engineer and Linux enthusiast posing as an English teacher after playing around a bit in sales and marketing. You can check out some of his Linux work at the EdUBudgie Linux website.

Discarding photo fails with Go and Fyne

Wheat and Chaff

If you want to keep only the good photos from your digital collection, you have to find and delete the fails. Mike Schilli writes a graphical application with Go and the Fyne framework to help you cull your photo library. *By Mike Schilli*

Command-line programs in Go are all well and good, but every now and then you need a native desktop app with a GUI, for example, to display the photos you downloaded from your phone and sort out and ditch the fails. At the end of the day, only a few of the hundreds of photos on your phone will be genuinely worth keeping.

Three years ago, I looked at a graphical tool – very similar to the one discussed in this article – that let the user manually weed out bad photos [1]. It ran on the Electron framework to remote control a Chrome browser via Node.js. Recently, the Go GUI framework Fyne has set its sights on competing with Electron and dominating the world of cross-platform GUI development. In this issue, I'll take a look at how easy it is to write a photo fail killer in Go and Fyne.

Recently, *ADMIN*, a sister publication of *Linux Magazine*, featured some simple examples [2] with Fyne, but a real application requires some additional polish. Listing 1 [3] shows my first attempt at a photo app that reads a JPEG image from disk and displays it in a window along with a *Quit* button. The photo dates back to my latest tour of Germany in 2021, where I set out to track down Germany's best pretzel bakers between Bremen and Bad Tölz. Figure 1 shows the app shortly after being called from the command line with a fabulous pretzel from Lenggries on the southernmost edge of Germany. My only complaint about the app's presentation with the photo and the *Quit* button is that it takes a good two seconds to load a picture from a cellphone camera with a 4032x3024 resolution from the disk and display it in the application window. Building a tool for image sorting with

Author

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

Listing 1: img.go

```
package main

import (
  "fyne.io/fyne/v2"
  "fyne.io/fyne/v2/app"
  "fyne.io/fyne/v2/canvas"
  "fyne.io/fyne/v2/container"
  "fyne.io/fyne/v2/widget"
  "os"
)

func main() {
  win := app.New().NewWindow("imgtest")
  img :=
    canvas.NewImageFromFile("pretz.jpg")
  img.SetMinSize(fyne.NewSize(600, 400))

  button := widget.NewButton("Quit",
    func() { os.Exit(1) })

  con := container.NewVBox(img, button)
  win.SetContent(con)
  win.ShowAndRun()
}
```

sluggish handling like this wouldn't attract a huge user base.

Faster Loading

The lightning-fast GUI presented in this issue, which displays the images from the current directory one after the other, moves to the next or previous image through Vi-style control with *L* or *H* and dumps the currently displayed image into the trash can directory `old/` if you press *D* (for "delete"). These quick movements make it easy and fast to separate the wheat from the chaff. By the way, if you are bothered by the Vi keyboard binding and prefer to use the cursor keys instead, you can simply make a two-line change to the code and be on your merry way.

While the trivial app from Listing 1 works quite slowly, the iNuke photo app shown in Listing 2 goes through the photo collection much faster. With a few tricks from the performance treasure trove, it displays the next photo almost immediately, with a delay of less than a perceived 10th of a second after you have requested it by pressing *L*. Magic? Not on the agenda for this column – we're keeping things real.

First off, caching previously loaded photos helps. This means that the GUI renderer only has to get them out of the cache and into video memory in case the user asks for them again. But which photos are worth keeping if they don't all fit in RAM? After all, a directory could hold 5,000 photos of 4MB each – and not everyone has 20GB of memory to spare. The solution is a Least Recently Used (LRU) cache, which holds a predefined maximum number of entries, but if overfilled, simply discards the items whose last access date is the oldest. Newly added entries simply overwrite older ones if the cache is already full.

As a second tuning tool, efficient downsizing of the photos before displaying them helps. Hardly any monitor displays 4032-pixel-width images in full. If you hand over the full-scale photos to the GUI for displaying, you are making it do more work than necessary, and the GUI exacts its revenge in the form of a sluggish response for the user, who – understandably – wants to see a new image without any delay on every keystroke. The *nfnt* Go library on GitHub offers highly efficient routines for shrinking images; a powerful app always shrinks photos to screen size before they even enter the cache.

And third, a preload mechanism helps the app gain tremendous speed. By design, it always displays photos in a certain order, either forward or backward, depending on the direction the user is navigating. This means that the app can easily predict which photo should appear on the screen with the next keystroke. If the app loads the next likely photo into the cache in the background while the current one is still visible, the Fyne framework can display the next photo almost immediately as soon as the button is pressed.

Zap!

The results are amazing: With these three improvements, the display in the Go program runs at a breathtaking pace and beats many a professional app. In Figure 2, iNuke has just loaded an image showing the author as a tourist in Heidelberg during his 2021 tour of Germany. The small label widget attached below shows which keystrokes are now expected. Pressing *H* tells the app to jump back to the last picture, *L* goes forward to the next shot, *D* deletes the current photo, and *Q* tells the app to quit. Now, what does the code for this solution look like in Go?

The main program in Listing 2 first defines a new GUI window in lines 18 and 19 with `app.New()` and later crams newly loaded images into it with `showImage()` in lines 58 and 81.

First, it determines the current working directory in line 25, reads all the JPEG photos it finds therein, and stores

Figure 1: The simple Fyne app (Listing 1) displaying a photo.

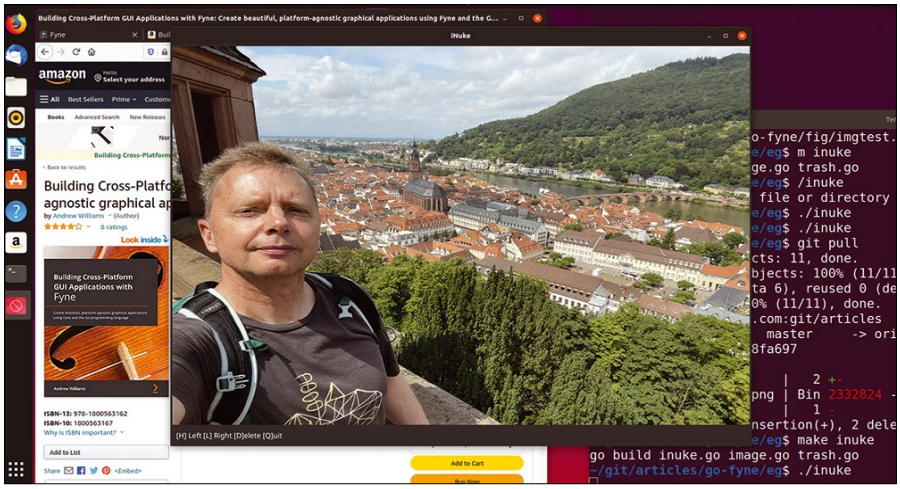


Figure 2: The new iNuke app utilizing the Fyne framework, showing a photo for selection.

Listing 2: inuke.go

```
001 package main
002
003 import (
004   "container/list"
005   "os"
006
007   "fyne.io/fyne/v2"
008   "fyne.io/fyne/v2/app"
009   "fyne.io/fyne/v2/storage"
010   "fyne.io/fyne/v2/container"
011   "fyne.io/fyne/v2/canvas"
012   "fyne.io/fyne/v2/widget"
013   "github.com/hashicorp/golang-lru"
014 )
015
016 var Cache *lru.Cache
017
018 func main() {
019   win := app.New().NewWindow("iNuke")
020
021   var err error
022
023   Cache, err = lru.New(128)
024   panicOnErr(err)
025
026   cwd, err := os.Getwd()
027   panicOnErr(err)
028
029   dir, err := storage.ListerForURI(
030       storage.NewFileURI(cwd))
031   panicOnErr(err)
032
033   files, err := dir.List()
034   panicOnErr(err)
035
036   images := list.New()
037
038   for _, file := range files {
039     if isImage(file) {
040       images.PushBack(file)
041     }
042   }
043
044   if images.Len() == 0 {
045     panic("No images found.")
046   }
047
048   cur := images.Front()
049
050   img := canvas.NewImageFromResource(nil)
051   img.SetMinSize(
052       fyne.NewSize(DspWidth, DspHeight))
053   lbl := widget.NewLabel(
054     "[H] Left [L] Right [D]elete [Q]uit")
055   con := container.NewVBox(img, lbl)
056   win.SetContent(con)
057
058   showImage(img, cur.Value.(fyne.URI))
059   preloadImage(scrollRight(images,
060     cur).Value.(fyne.URI))
061
062   win.Canvas().SetOnTypedKey(
063     func(ev *fyne.KeyEvent) {
064       key := string(ev.Name)
065       switch key {
066       case "L":
067         cur = scrollRight(images, cur)
068       case "H":
069         cur = scrollLeft(images, cur)
070       case "D":
071         if images.Len() == 1 {
072           panic("Not enough images!!")
073         }
074         old := cur
075         cur = scrollRight(images, cur)
076         toTrash(old.Value.(fyne.URI))
077         images.Remove(old)
078       case "Q":
079         os.Exit(0)
080       }
081       showImage(img,
082         cur.Value.(fyne.URI))
083       preloadImage(scrollRight(images,
084         cur).Value.(fyne.URI))
085     })
086
087   win.ShowAndRun()
088 }
089
090 func scrollRight(l *list.List,
091     e *list.Element) *list.Element {
092   e = e.Next()
093   if e == nil {
094     e = l.Front()
095   }
096   return e
097 }
098
099 func scrollLeft(l *list.List,
100     e *list.Element) *list.Element {
101   e = e.Prev()
102   if e == nil {
103     e = l.Back()
104   }
105   return e
106 }
```

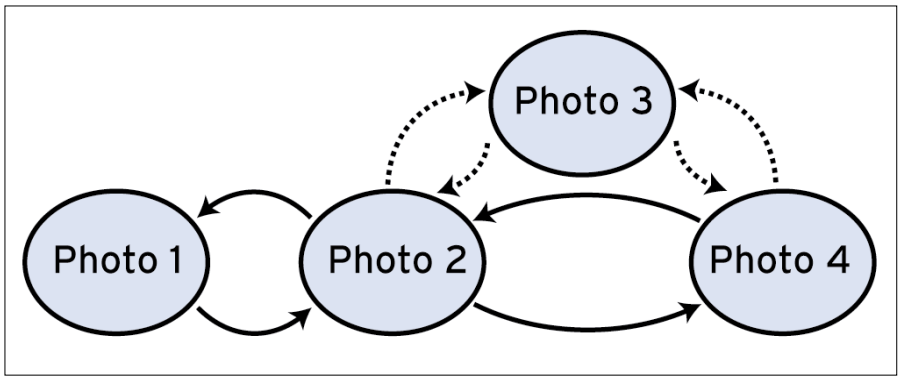


Figure 3: In a doubly linked list, users can scroll back and forth despite the gaps.

the file URLs as their paths in a Fyne framework storage structure. This mechanism is used in Fyne to abstract file paths because not all operating systems provide access to a filesystem. For example, a mobile phone can fetch data from the cloud or from a local database. Thanks to Fyne's abstraction layer, subsequent functions process the data completely transparently.

The keystrokes are intercepted by the `SetOnTypedKey()` function, providing a callback routine starting in line 62. Intercepting keystrokes is quite exotic for a GUI that tends to wait for mouse clicks. But Fyne allows it, and a keyboard cowboy like me shuns the mouse like the devil shuns holy water. Pressing *L* goes to the right, *H* to the left, and *D* hoists a delete flag for the current photo and uses `toTrash()` to send it to the recycle bin, aka the "old" directory.

The event loop in the main program, which first puts the GUI on the screen and then responds to input lightning-fast with reloaded photos, is started in line 87 by `win.ShowAndRun()`.

Array with Holes

So how is the main program supposed to efficiently manage the list of photos that the user browses through like a dervish, deleting an entry here and there that

Listing 3: image.go

```
01 package main
02
03 import (
04   "fyne.io/fyne/v2"
05   "fyne.io/fyne/v2/canvas"
06   "fyne.io/fyne/v2/storage"
07   "github.com/nfnt/resize"
08   "image"
09   "strings"
10 )
11
12 const DspWidth = 1200
13 const DspHeight = 800
14
15 func isImage(file fyne.URI) bool {
16   ext :=
17     strings.ToLower(file.Extension())
18   return ext == ".jpg" || ext == ".jpeg"
19 }
20
21 func scaleImage(
22     img image.Image) image.Image {
23   return resize.Thumbnail(DspWidth,
24     DspHeight, img, resize.Lanczos3)
25 }
26
27 func preloadImage(file fyne.URI) {
28   if Cache.Contains(file) {
29     return
30   }
31   go func() {
32       img := loadImage(file)
33       Cache.Add(file, img)
34   }()
35 }
36
37 func showImage(
38     img *canvas.Image, file fyne.URI) {
39   e, ok := Cache.Get(file)
40   var nimg *canvas.Image
41   if ok {
42     nimg = e.(*canvas.Image)
43   } else {
44     nimg = loadImage(file)
45     Cache.Add(file, nimg)
46   }
47   img.Image = nimg.Image
48   img.Refresh()
49 }
50
51 func loadImage(
52     file fyne.URI) *canvas.Image {
53   img := canvas.NewImageFromResource(nil)
54
55   read, err :=
56     storage.OpenFileFromURI(file)
57   panicOnErr(err)
58
59   defer read.Close()
60   raw, _, err := image.Decode(read)
61   panicOnErr(err)
62
63   img.Image = scaleImage(raw)
64   img.FillMode = canvas.ImageFillContain
65
66   img.SetMinSize(
67       fyne.NewSize(DspWidth, DspHeight))
68
69   return img
70 }
```

they never want to see again? An array would be the wrong data structure here because arrays with holes mean time-consuming renovation work. Instead, the main program taps into the standard `container/list` library. This is a doubly linked list in which the program can quickly move to the next element using `Next()` and to the previous element with `Prev()`, even if a `Remove()` has deleted entries in the meantime (Figure 3). The memory requirement for the entire collection in `images` is somewhat higher than for an array because of the links connecting the elements, but speedy deletion of arbitrary elements without any compromises when browsing is well worth the cost.

The `scrollRight()` and `scrollLeft()` functions in lines 90 and 99, respectively, return the next photo to be displayed when maneuvering to the right (*L*) or left (*H*). Even if the user boldly goes beyond the end of the list, no error is thrown. If the user overshoots to the right, `scrollRight()` uses `Front()` to jump back to the beginning of the list, and if the user moves further to the left from the first element, `scrollLeft()` jumps to the last list element.

The routines for scaling and loading image files are shown in Listing 3. In line 15, `isImage()` helps to determine whether or not a file is a JPEG photo. It determines the type based on the extension. The task of scaling down large-format cellphone photos to 1200x800 is handled by `scaleImage()` starting in line 21, which accesses the `resize()` function of the *nfnt* package from GitHub. The Lanczos3 algorithm implemented there definitely shrinks cellphone photos faster than Fyne after determining that an image is too large to be displayed in an assigned widget.

Faster Thanks to Prepping

Newly loaded photos are displayed by `showImage()` from line 37 onwards; it tries to load them from the cache first and, if this fails, scrapes them from the disk using `loadImage()` and decodes them. This takes time, but at some point this has to be done. Because of this delay of a good second, `preloadImage()` uses a Goroutine starting in line 31 with `go func` to handle loading in the background while the main program continues to run and respond to user input. When the user requests the next image, it is usually already in the cache, and `showImage()` fetches it and sends it to the screen at lightning speed.

Invisible Cache

Thanks to the *hashicorp/golang-lru* package from GitHub, the LRU cache maintained in the `Cache` global variable need not worry about wasted RAM. Line 23 in Listing 2 defines a cache with a maximum of 128 entries for preprocessed images, into which `Add()` (line 45, Listing 3) inserts new items under the photo's file path URL, while `Get()` (line 39) retrieves them from the cache. Although the LRU cache provides the `Contains()` function that determines whether an item is in the cache, users should always try to retrieve an item if they actually need it in order to avoid race conditions. Otherwise, although `Contains()` might report that the entry is there, a concurrent program could also cause it to disappear before the following `Get()`.

If the cache is full, `Add()` simply throws the oldest entry out of the cache before inserting the new one, according to LRU rules. For the app, which later may still be looking for the old entry because the user went back to that old photo, this is not the end of the world. It can simply retrieve the image from the disk and put it back in the cache. This takes a bit longer, but in this inconvenient case you just have to wait.

By the way, one more subtle feature of Go's strict type system: Containers such as the LRU cache let you store generic data types. Therefore, when fetching, you must make sure that the entry is given the correct type again using runtime type assertions. For example, line 42 converts a found cache entry into a pointer to a `canvas.Image` type because the retrieved photo is of that type, even though the LRU cache had stored it as a generic `interface{}` type in the meantime.

Such manipulations wreak havoc with Go's strict type system, however. What Go normally fields at compile time therefore turns into an annoying runtime error if these conversions have not been carefully tested.

By the way, Fyne normally avoids absolute coordinate values in layout instructions and scales widgets like buttons automatically. Because of the widgets' captions and the font used, this can be done without explicit layout instructions. However, this is not possible with photos because Fyne has no way of knowing how big the containing canvas object needs to actually appear on the screen. If you don't give Fyne a minimum size for the widget in the form of `SetMinSize()`, you may have to use a magnifying glass to search the screen for the widget you are looking for. Without defaults, Fyne paints 0x0-sized images, often resulting in perplexingly hard-to-find app windows. If you specify the minimum size, like in line 66 of Listing 3, you can see what's going on.

Finally, Listing 4 implements the virtual trash can that uses `toTrash()` to dump photo files to a newly created `old/` directory as needed. This is done by the `Rename()` function from the standard *os* package, which works without complaint as long as the original and the target file reside on the same storage medium.

Adapting the Look & Feel

To make Go download the Fyne packages used in the listings from the server at `fyne.io` onto your local system and compile the whole enchilada, you need

Listing 4: trash.go

```
package main

import (
  "os"
  "path/filepath"

  "fyne.io/fyne/v2"
)

const TrashDir = "old"

func toTrash(file fyne.URI) {
  err := os.MkdirAll(TrashDir, 0755)
  panicOnErr(err)
  err = os.Rename(file.Name(),
    filepath.Join(TrashDir, file.
Name()))
  panicOnErr(err)
}

func panicOnErr(err error) {
  if err != nil {
    panic(err)
  }
}
```

Listing 5: trash.go

```
$ go mod init inuke
$ go mod tidy
```

the sequence in Listing 5.

A subsequent

```
go build inuke.go image.go trash.go
```

should then generate a binary `inuke` without any errors. Upon launch, it will display the first photo in the current directory inside its GUI window. In the usual Go style, the compiler can also cross-compile for other platforms. In the case of Fyne, this even goes so far as to have the GUI code draw in the look and feel of the other platform. How this works is described in detail in the Fyne book by the Fyne guru himself [4]. The code from the three listings also compiles without problems on a Mac and results in a slightly adapted Apple look, as shown in Figure 4.

The graphical interfaces' implementation also depends heavily on the operating system you use. On Linux, Fyne taps into the *libx11-dev*, *libgl1-mesa-dev*, *lbx-cursor-dev*, and *xorg-dev* libraries using a C wrapper from Go. You need to install these on Ubuntu, for example, by typing

```
sudo apt-get install
```

to ensure that a subsequent `go build` for a Fyne app actually finds the required underpinnings.

Nobody Is Perfect

This app is by no means perfect yet. I've heard that there are actually photographers out there who take pictures in portrait format, which would require a corresponding adjustment of the fixed window dimensions. Also, some cellphones are quirky about saving photos without rotation and only note the rotation specification in the Exif header of the JPEG photo. Corresponding adjustments can be made easily, as always with open source projects. ■■■

Info

[1] "Programming Snapshot: Using the Electron Framework to Weed Out Images" by Mike Schilli, *Linux Magazine*, issue 216, November 2018, p. 46, *https://www.linux-magazine.com/index.php/Issues/2018/216/Clever-Sampling*

[2] "Program GUIs in Go with Fyne" by Markus Hoffmann, *ADMIN*, issue 66, 2021, pp. 26-29

[3] Listings for this article: *ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/254/*

[4] Williams, Andrew. *Building Cross-Platform GUI Applications with Fyne*. Packt Publishing, January 2021, *https://www.packtpub.com/product/building-cross-platform-gui-applications-with-fyne/9781800563162*

Figure 4: An app compiled for the Mac from the same code.

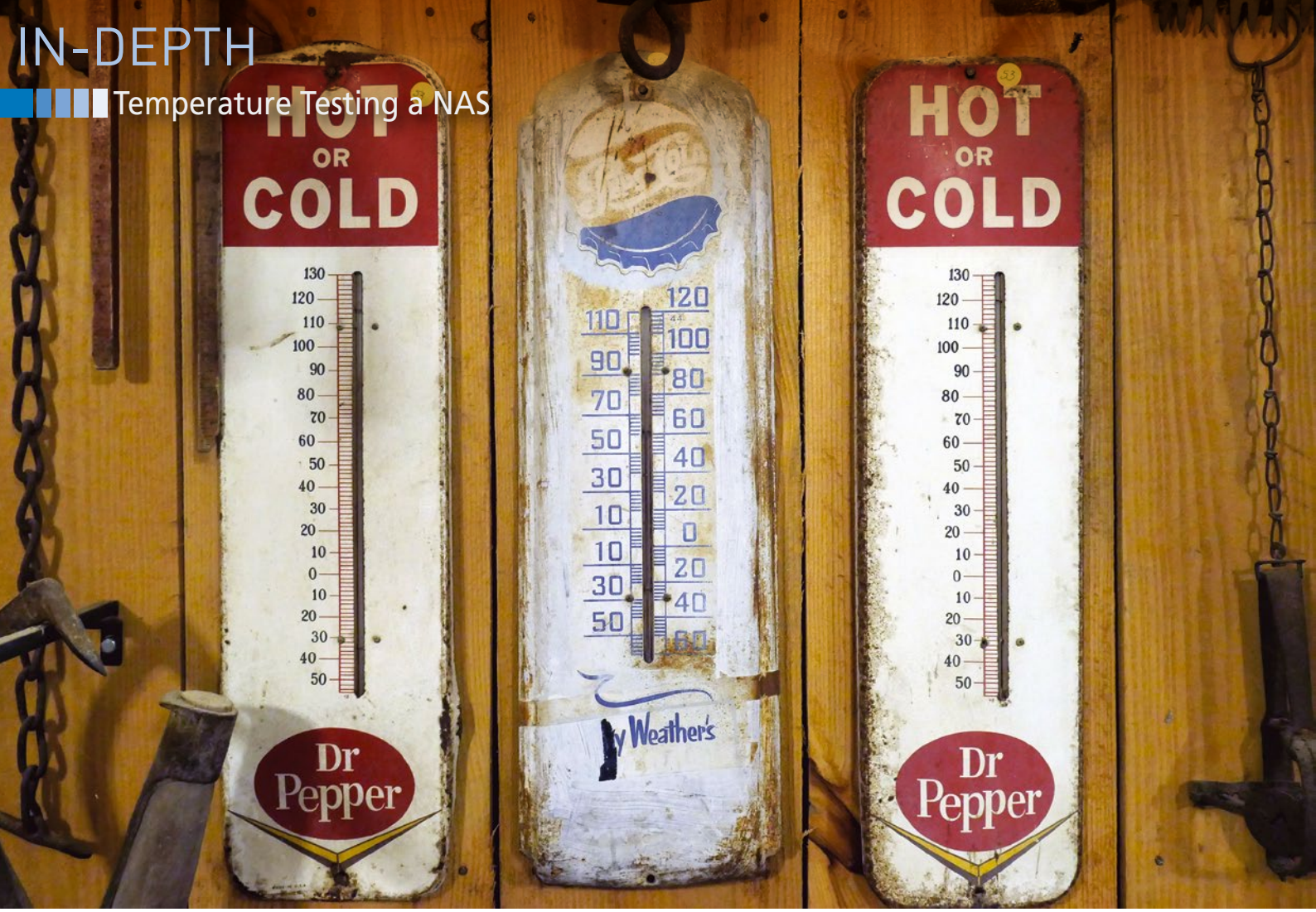


Photo by Ilse Orsel on Unsplash

Stress testing for temperature on a home NAS

Temperature Watch

Using stress, lm-sensors, and hddtemp to sort out temperature and reliability related issues with a home-based NAS box. *By Adam Dix*

I recently found myself in a difficult situation with my home NAS brought about by some sketchy construction work done at my apartment. Long story short, the workers didn't mention that they would be using my work area as *their* work area and set about cutting bathroom tiles right next to my main workstation, my home NAS server, and a toy i3-based rig that I occasionally use for testing and special projects. Consequently, my NAS suffered failure after failure – with ghosts that I am *still* chasing more than three months later.

My NAS is a home-built rig using a Supermicro X8STi with an X5680, 24GB of ECC DDR3, and six disks total, including the operating system (OS) drive itself. The OS is on an SSD, and the data is all on HDDs. There is *no* redundancy (see my article on MergerFS elsewhere in this issue). So, to be clear, this *is* a mess primarily of my own making, but one not without others at fault.

The first problem that I experienced after the aforementioned construction was that my disks would randomly disappear after having been present at boot. I cleaned the machine out as best as I could and reseated the SATA cables. This worked temporarily, but two of the drives would still drop off, requiring a reboot, and in some cases, requiring me to reseat the SATA cable. I should note that, at the time, I was using old inexpensive SATA cables that had been collected over the years from motherboard purchases, and so I decided it was time to switch to more modern cables with the locks that the SATA III specification requires. The problems remained.

At this point, having opened the box numerous times, I realized that the intake fan in front of the hard drive cage had failed. The fan was covered in red dust from the construction. Though I cannot say definitively that this was what caused its demise, it was followed very quickly by the CPU fan. Both were 120mm fans running full-tilt 24/7, and neither had filters or screens in front of them. The next item to go was the power button, oddly enough. Although I cannot assume that dust directly caused that failure, the

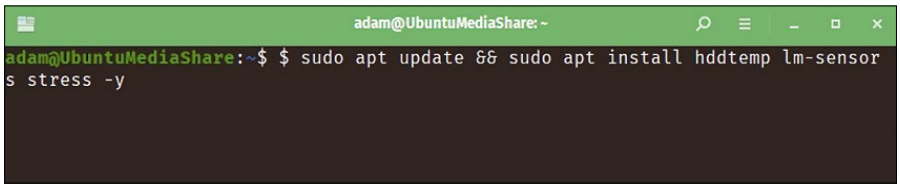


Figure 1: Installing stress testing and monitoring tools.

power and reset buttons certainly got some additional mileage after the construction was completed as I chased these demons down. The case was an inexpensive affair made of cheap plastic and jagged steel, which seemed to take its final shape only once fully populated with gear, but I hadn't had any issues with it before the construction.

At this point, I decided that a cursory cleaning wasn't enough, and I really needed to rip everything out, clean everything off well, rewire it all, replace the faulty fans, and test. Needless to say, it is good that this is only a home NAS box and not something used in any sort of real production, as this is the exactly the type of scenario that is avoided at all costs in the server world.

After taking everything out and dusting it all as carefully as possible, it would have been sacrilege for me not to replace the CPU Thermal Interface Material (TIM) between the CPU lid and the heat sink. The X5680 is a 130W CPU and lets you know it when under load as its temperature ramps up quickly.

At this point in my epic saga, I have a freshly rebuilt box with known working components but without a feel for the performance and temperature with the new fans installed. I decided to use three simple and commonly-used programs to address these needs:

- `stress` [1] (which is also available in an enhanced version known as `stress-ng`) is a workload generator that you can use to apply a configurable load to your system.

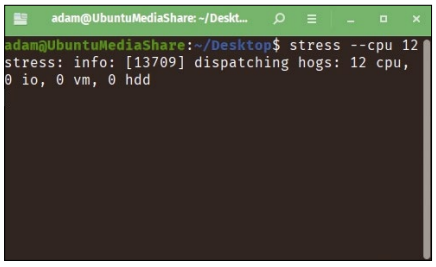


Figure 2: Running `stress` on all 12 threads of a hexa-core CPU with hyperthreading.

- `hddtemp` [2] is a tool that will monitor the temperature of your hard drive to make sure the drive is operating in the recommended range. `hddtemp` works by accessing the Self-Monitoring Analysis and Reporting Technology (S.M.A.R.T.) information available with many hard drives.
- `lm-sensors` [3] is a utility that can read and report data from sensors located in the hardware, including sensors for monitoring temperature, fan speed, and voltage for the CPU, mainboard, and other components.

These tools have all been around for a while, and you might already be familiar with them. What I decided to do was to run `stress` on all 12 threads of the CPU while at the same time having Plex add all of my preexisting media to the Plex library. This test is really the most use that those drives will ever get at one time. With the CPU working on all threads, the test would simulate not only the maximum power draw but also allow me to see how well the cheap 120mm knock-off fans perform.

Simple Stress Test

To install (on Ubuntu-based systems such as mine) and to make sure that you have the most recent versions of the programs, run the following commands in the terminal (Figure 1):

```
$ sudo apt update && ↵
  sudo apt install hddtemp ↵
  lm-sensors stress -y
$ sudo apt update && ↵
  sudo apt upgrade -y
```

My system was running Ubuntu [4] 20.04.3 LTS and was up-to-date at the time of stress testing. The install only takes a short moment, and from there, all operations are done from the terminal, so you can use Ctrl + L to clear the screen and leave the terminal open to start the stressing. I would also recommend opening a second tab or another terminal instance for the monitoring. If you are connecting to a headless server via SSH or using some other terminal emulator, such as that which is found in Webmin [5] or Cockpit [6], you will need to adjust your workflow accordingly. At any rate, I found it was easier having two terminal windows open, one to run `stress` and to be able to stop the test from running using Ctrl + C, and the other to refresh the monitoring programs as needed.

To start running `stress`, I used the following command to load up all of the threads on the CPU:

```
$ stress --cpu 12
```

The output appears in Figure 2. Once the test had been running for a few minutes,

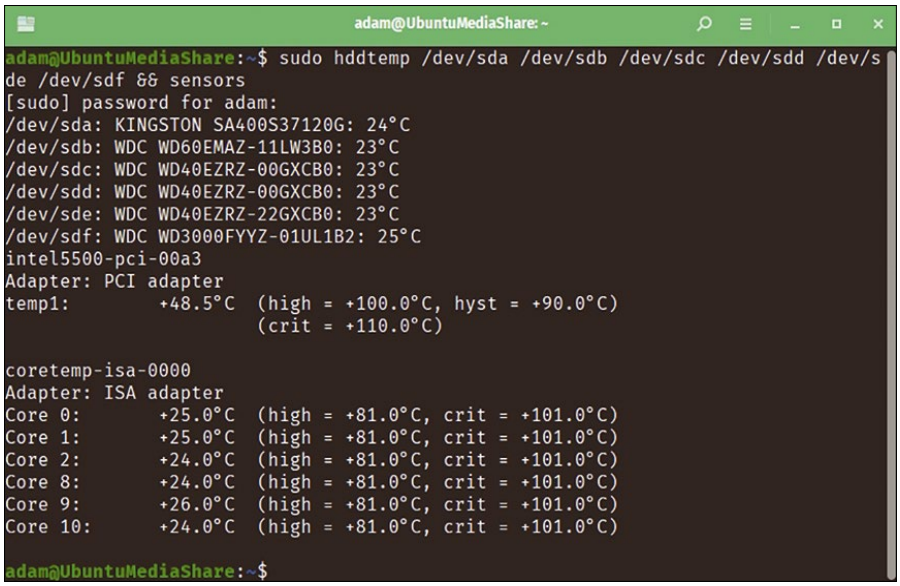


Figure 3: Monitoring for temperatures with `hddtemp` and `lm-sensors`.

I ran the following command in the second terminal window in order to see where the temps were:

```
$ sudo sensors && ➥
  hddtemp /dev/sda /dev/sdb ➥
  /dev/sdc /dev/sdd /dev/sde /dev/sdf
```

Figure 3 shows the result. You might need to run the following command first in order for it to know which sensors exist and which can be checked by the program itself:

```
$ sudo sensors-detect
```

I don't know why I had to spell out each of the drives individually for `hddtemp` to work. Simply typing `hddtemp` should list the temperatures of all drives using the default Celsius scale, with each drive appearing on its own line, but that didn't work for me. Adding the `/dev/sdX` for each drive after the command did work and displayed each drive on its own line with temps shown in Celsius. Perhaps this had something to do with one drive being an SSD and rest being HDDs, though I doubt that was the cause. Perhaps it was because I use a cheap "RAID" card (it is not really a RAID card but rather an inexpensive Marvell chip with some SATA ports connected to it), which connects via PCIe and allows for four additional SATA III drives to be installed in the system.

For about an hour, I clicked the up arrow and ran the monitoring command every so often to make sure that the temps were OK. My CPU will run without a problem and boost normally up to around 81˚C; during testing, mine peaked at about 75˚C. Typical home-use HDDs, such as the WD Blues that I was using, shouldn't exceed 35-40˚C, and mine hovered around 31˚C during testing. Some folks will do stress testing for hours and even for over a day or more, but this is a home NAS with non-critical data stored on it in a temperature-controlled environment. The heat sink for the CPU was thoroughly heat-soaked after about 10 minutes of operation and the drives are located directly behind the previously mentioned 120mm intake fan, meaning they were receiving fresh (bathroom tile dust-free) air for the test.

Conclusion

After my experience with the recent construction in my home, I can pass on one very important lesson: make sure that your devices are covered and turned off if someone is doing construction nearby. Dust from construction is not great when run through your server, workstation, or laptop.

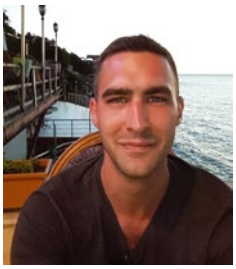
Running stress-testing software with a monitoring program is a great way to make sure that your system will stand the test of time and that your components will run optimally, especially after a new build or a rebuild. It is important to have a good baseline for your system to know if and when it is running well. One of the best things that I have found in the open source community is that there is a program, app, script, or Flatpak for just about anything you can imagine. With all of the issues that I have had to deal with pertaining to this NAS server, software was never an issue. Handy utilities like `stress`, `hddtemp`, and `lm-sensors` were easy to use and gave me important insights on my system.

It is easy to forget about the folks who have supported the open source community over the years, especially when you find yourself in situations that you would rather not be in and with hardware that seemed to have been cursed, but I can say, unequivocally, that I have far more faith in the contributors unknown to me who helped to make my NAS system work than I do in the construction workers I let into my house. For that, I would just like to say – very loudly, so that you can hear me over these cheap fans – thank you! ■■■

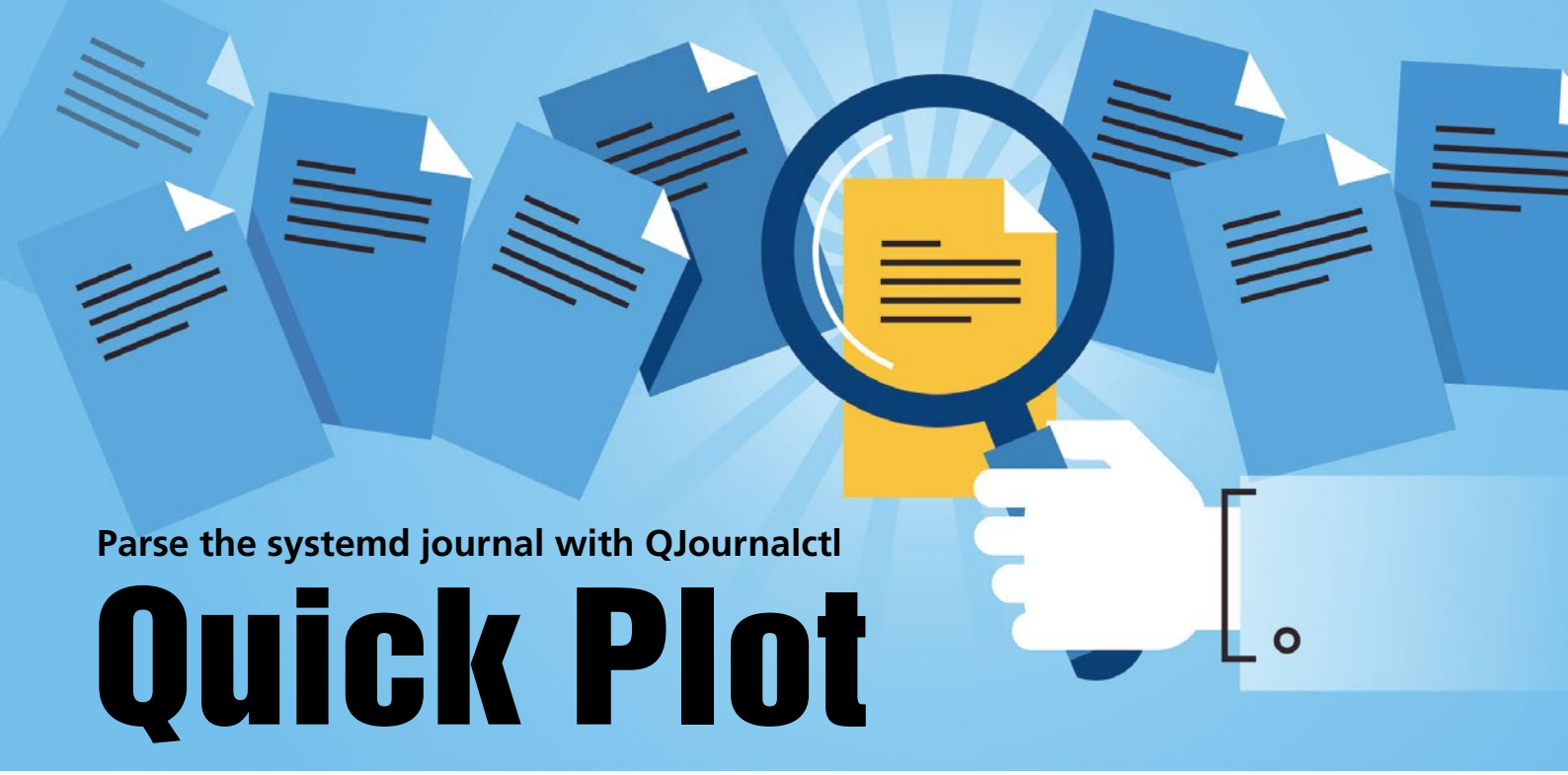
Info

[1] stress: *https://launchpad.net/ubuntu/+source/stress/*
[2] hddtemp: *https://launchpad.net/ubuntu/+source/hddtemp/*
[3] lm-sensors: *https://launchpad.net/lmsensors/*
[4] Ubuntu: *https://ubuntu.com/*
[5] Webmin: *https://www.webmin.com/*
[6] Cockpit: *https://cockpit-project.org/*

Author



Adam Dix is a mechanical engineer and Linux enthusiast posing as an English teacher after playing around a bit in sales and marketing. You can check out some of his Linux work at EdUBudgie Linux (*https://www.edubudgie.com*).



Parse the systemd journal with QJournalctl

Quick Plot

QJournalctl is a convenient GUI tool that will help you track down log data in the systemd journal.

By Ferdinand Thommes

After more than 10 years of development, systemd has undeniably arrived. The systemd system and service manager still does not find undivided approval in the open source community, but the discussions centered on what was once a very controversial topic have calmed down. No matter what you think of systemd, it *has* improved a few things.

One of the benefits of systemd is its ability to log information, warnings, and errors. In systemd, the component responsible for logging is known as the journal. Systemd-journald provides much more data than conventional log files `/var/log/messages` or `/var/log/syslog` that have been used for decades. In some cases, the journal even includes metadata that you can use to refine your journal searches.

Systemd's journaling results in a high volume of data that would be difficult to handle with conventional text files. Because of this high volume, systemd stores the data in binary files that can be read with the `journalctl` command. Various programs and services, and even the kernel, write their log entries to the journal. The systemd journal combines all these logs and offers various options for finding the information you need in a targeted way.

QJournalctl [1] is a graphical user interface you can use to view and navigate the systemd journal. Versions of QJournalctl are available for Linux, MacOS, and Windows systems. QJournalctl does not support *all* the capabilities available at the command line through the `journalctl` command, but it does offer some important options that will save you time and help you track down real-world problems on your Linux system.

Installing

In distributions based on Arch Linux, you can install QJournalctl version 0.6.3 by typing:

```
sudo pacman -S qjournalctl
```

Gentoo provides an ebuild file. For DEB- and RPM-based distros, you might need to build the application from source. The developer does pro-

Listing 1: libssh for Debian

```
$ sudo apt install qtbase5-dev cmake libssl-dev pkg-config git
$ wget https://www.libssh.org/files/0.9/libssh-0.9.5.tar.xz
$ tar xf libssh-0.9.5.tar.xz
$ cd libssh-0.9.5
$ mkdir build
$ cd build
$ cmake
$ sudo make install
```

Listing 2: Compiling from Source Code

```
$ mkdir -p QJournalctl
$ cd QJournalctl
$ git clone https://github.com/pentix/QJournalctl.git
$ cd QJournalctl
$ ./autogensh
$ make -j$(nproc)
$ sudo make install
```

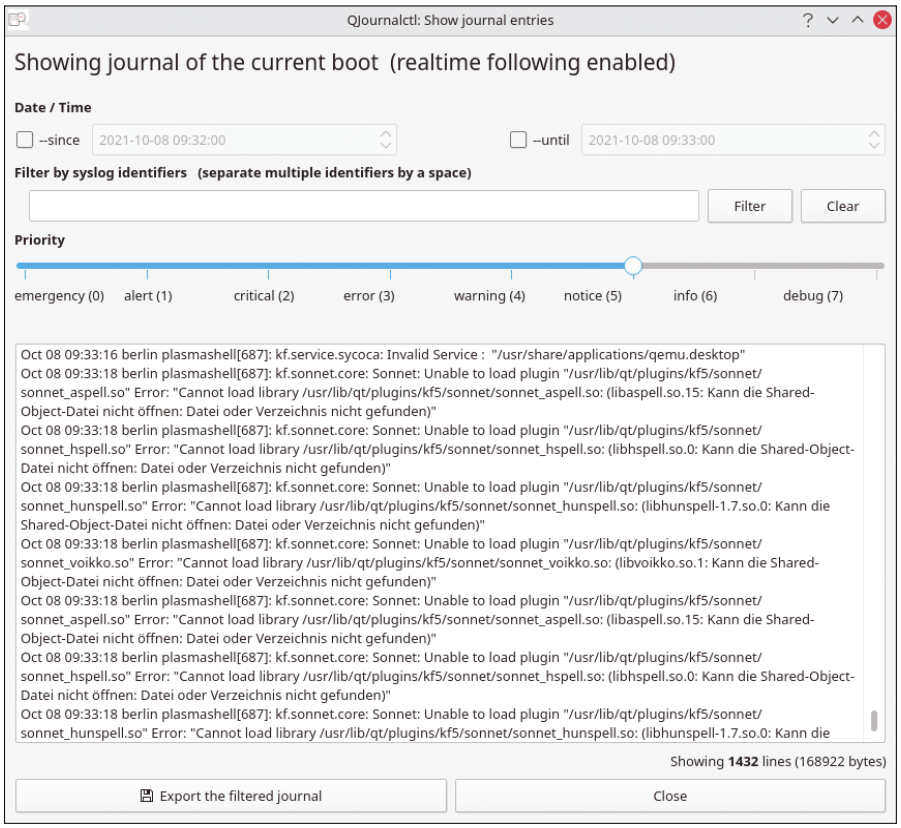


Figure 1: The most common use case for QJournalctl is to view the log for the current boot process. QJournalctl works in realtime mode and displays incoming log messages directly during runtime.

vide a Debian package [2], but, at this writing, the Debian package includes a dependency – libssh 0.8.7 – that neither Debian, nor Ubuntu, nor their derivatives provide in the required version. So you might need to build the libssh yourself (Listing 1) before you can install the DEB package. But because you are working at the command line anyway, it makes sense to compile QJournalctl from the source code right from the outset (Listing 2).

Fedora 34 and openSUSE Leap 15.2 come with the correct version of libssh, in which case, you only need to install the dependency and build QJournalctl (Listing 3).

Listing 3: On Fedora or openSUSE

```
$ sudo dnf install libssh-devel qt5-qtbase-devel git
$ mkdir -p QJournalctl
$ cd QJournalctl
$ git clone https://github.com/pentix/QJournalctl.git
$ cd QJournalctl
$ ./autogensh
$ /usr/bin/qmake-qt5
$ make -j$ (nproc)
$ sudo make install
```

Getting Started

After opening QJournalctl, you first see a large amount of free space and no journal. The menu at the top is also quite sparse. Below it are three tabs, the first of which is probably the most commonly used because it opens the boot log for the current session.

Clicking on this tab opens a new window with the boot log from the last boot. At first, you will hopefully only find a few lines – and ideally, nothing. If you let your eyes wander to the bar below *Priority*, you will soon understand why (Figure 1).

By default, the priority of the default syslog level is set to *critical (2)*, so you will only see messages that are considered critical by the system. Drag the slider to the right, and you will see progressively more messages until you get to *debug (7)*.

In the terminal, you would set this priority by typing `journalctl -p7`. The *info (6)* level is typically fine. Between levels 2 and 6, the number of lines displayed in our test increased from less than 10 to several thousand since the last boot (which was four days ago).

A total of six boot processes spanning four months were stored on the test system, claiming around 4GB of disk space. The system sets a limit, even if you have not set a maximum size yourself – but more on that later.

Filtering and Exporting

QJournalctl offers even more filter options: At the very top, below *Date / Time*, you can specify the time window for the log display. If you also know exactly what you are looking for, you can use the filter function to further narrow down a specific bug or problem. For example, if you only want to see lines relating to the kernel or NVidia, enter the

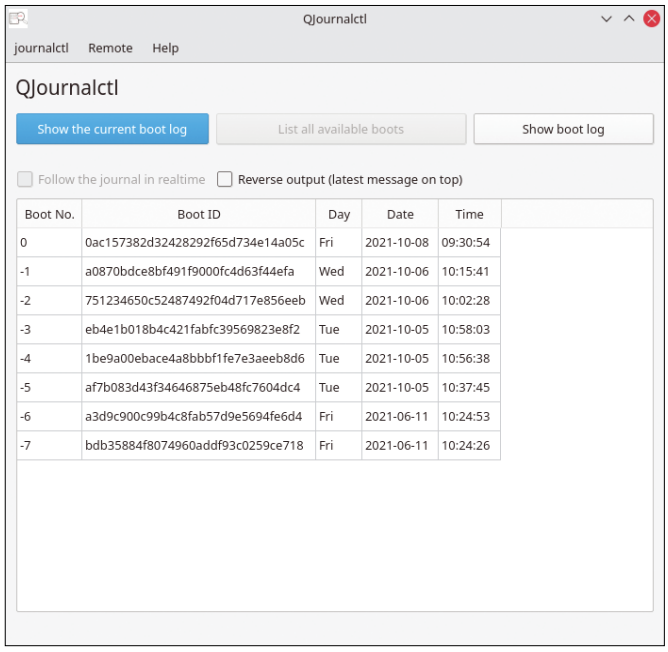


Figure 2: The middle tab lists all stored logs. After selecting the desired log, you can change the display direction and then start to display with the right tab.

Listing 4: Manual Work

```
01 $ sudo journalctl --vacuum-size=100M
02 $ sudo journalctl --vacuum-time=1month
03 $ systemctl status systemd-journald
```

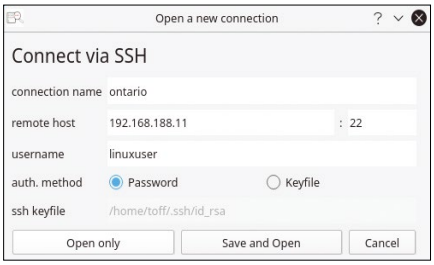


Figure 3: Easy access to remote machines via SSH is a special treat. You can save the credentials of remote machines and quickly connect later.

command in the filter line and press the button labeled *Filter*.

The middle tab in the main window lists all existing journal entries for previous boot actions. You can then select the matching logs by clicking on an entry. You can access all the stored logs at once by selecting *Show the complete journal* from the main menu under the first tab.

On a PC with a Ryzen-7 CPU, viewing all saved logs in unfiltered form took about a minute and returned more than 6 million lines. You can query the total size in the main menu using the menu item *Journalctl | Size of the journal on the disk* (Figure 2).

If you want to process a log in another way or simply save it, just export it to any path using the *Export the filtered journal* button. If you only need an excerpt of the displayed log, select it and save it by pressing the *Export Selection* button.

Going Remote

If you wondered at the beginning why QJournalctl needs the libssh library, the explanation is that QJournalctl can also display logs from remote computers via SSH. The *Remote* menu item offers commands for opening connections via SSH, as well as saving the access credentials for cases in which you query the same computers frequently. QJournalctl supports password-based access, as well as the use of a key file (Figure 3).

Under the bar with the three tabs, the main window offers options named *Follow the journal in realtime* and *Reverse output (latest message on top)*. The box to the left of *Follow the journal in realtime* always stays gray. If you are looking at the log of the currently active boot process, the live function is automatically added, which is also indicated by the additional *(realtime following enabled)* message at the end of the top line. If you are looking at the log of an older boot process, the live function doesn't make much sense, of course. But you can still check the *Reverse output* box to display the journal output in reverse order.

What's Missing?

QJournalctl deliberately does not provide functions for performing many of the administrative tasks that journalctl handles in the terminal. Systemd takes the total size of the journal, as well as the individual files, from the `/etc/systemd/journald.conf` configuration file. Optionally, you can truncate the journal on the fly; for example, the command from the first line of Listing 4 truncates it to 100MB.

You can also drop all messages more than, say, one month old (Line 2). The only status query QJournalctl allows in this context relates to the total size of the journal. You still need to query the general health of the service in the terminal (Line 3); use `journalctl --verify` to check the integrity of individual journals.

Conclusion

QJournalctl is a handy tool for anyone who doesn't use the journal regularly and thus does not have all the necessary commands for the terminal at their fingertips. In particular, QJournalctl facilitates searching within date and time constraints. The ability to display log files on remote computers and select them from a saved list is quite useful.

The QJournalctl project is maintained on GitHub. The developer told us that a major update is coming which will include a new colorized output feature that will help you highlight and organize log data [3]. ■■■

Info

[1] QJournalctl: *https://github.com/pentix/QJournalctl/tree/v0.6.3*

[2] DEB Packet: *https://github.com/pentix/QJournalctl/releases/tag/v0.6.3*

[3] Pull Request: *https://github.com/pentix/qjournalctl/pull/66*

Turn your ideas into reality!

This is not your ordinary computer magazine! MakerSpace is a new special issue that focuses on technology that you can use to build your own stuff.

If you're interested in electronics but haven't had the time or the skills (yet), studying these maker projects might be the final kick to get you started.

BONUS
Complete Raspberry Pi Geek Archive DVD free with the print edition!

This special issue will help you dive into:

- Raspberry Pi
- Arduino
- Retro Gaming
- FPGA
- and much more!

ORDER ONLINE:
shop.linuxnewmedia.com/specials

MakerSpace

A beautiful web radio for a living room music system

Jewel

An LCD in an unusual format, a self-designed housing, and matching software make a simple Raspberry Pi web radio the perfect centerpiece for any living room. *By Bernhard Bablok*

Special projects deserve special components, which is why a display with unusual dimensions acts as the basis for my living room music system. However, the path to the final version was fraught with challenges.

A web radio based on a Pi Zero (Figure 1) came to life in 2018 as a lightweight tinkering project with a keyboard, mini-speakers, and a small LCD screen [1]. Now, however, it has found a permanent place on my audio/visual (AV) receiver (Figure 2). Today, it is no longer controlled by a keypad, but by an infrared remote control. This solution works well, but it has one obvious drawback: The display is simply too small for couch potatoes.

Therefore, a new version of the project has been in the pipeline for quite some time but has failed to materialize thus far because of the available display alternatives. In contrast to the current solution, which simply wires the display, the IR receiver, and the Pi Zero with a few jumpers, I wanted the next version to match the existing device landscape in terms of appearance. However, large displays have had a fairly off-putting format

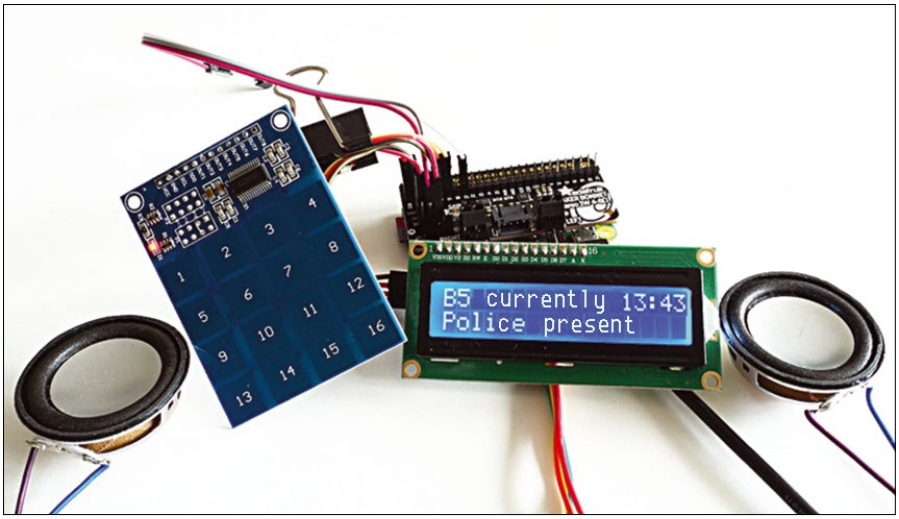


Figure 1: The original version of the web radio started in 2018 as a small tinkering project.

Lead Image © Christos Georghiou, 123RF.com

Figure 2: One drawback of the original project was a too-small LCD.

until now, and so the project didn't really get off the ground.

Rare Beast from Waveshare

This situation changed all of a sudden when Waveshare launched a touch-enabled TFT display with a 1280x400-pixel (px) resolution (Figure 3). Waveshare is well known in the small-board computer (SBC) scene, because the company provides displays in many different sizes, not just for the Raspberry Pi, and it has good support in the form of a wiki and various GitHub repositories. The price of the display is a bit steep at EUR110 (~$126), but it promises to be an eye-catcher.

The package includes the display with an aspect of 19.1x6cm (7.5x2.3in), some spacer bolts, a normal HDMI and USB cable, and various special adapters (Figure 4). The display's bezel is 8.35mm (3.3in) wide on the left and right, but only 4.7mm (1.85in) at the top and bottom.

Thanks to the adapters, you can install an SBC directly behind the display; they support the Raspberry Pi 3 and 4 and the Jetson Nano. The installation is a bit tricky, because precisely built adapters are needed for everything to fit – and they are not. In the end, mounting the Raspberry Pi with screws or spacer bolts proved to be the best solution, leaving a bit more space for the HDMI and USB adapters.

The display draws its power from the computer over USB; alternatively, it can be connected directly through a micro-USB socket. The touchscreen also uses USB and does not require a driver installation. The screen outputs its sound with a 3.5mm jack. For this to work, the Raspberry Pi has to switch the sound to the HDMI 0 output.

The display's moderate power consumption is not a problem for the intended use. Even with the VLC media player running and visual effects enabled (Figure 3), the power requirement of a Raspberry Pi 4, including the display, is only 1,100-1,500mA. However, even the fourth generation of the Raspberry Pi has not yet learned how to turn off its peripherals after shutdown, which means the system still consumes 250mA when switched off. Although not much, it is annoying because it is so unnecessary. The system could not be marketed as is, because devices are only allowed to consume 0.5W in standby in most countries.

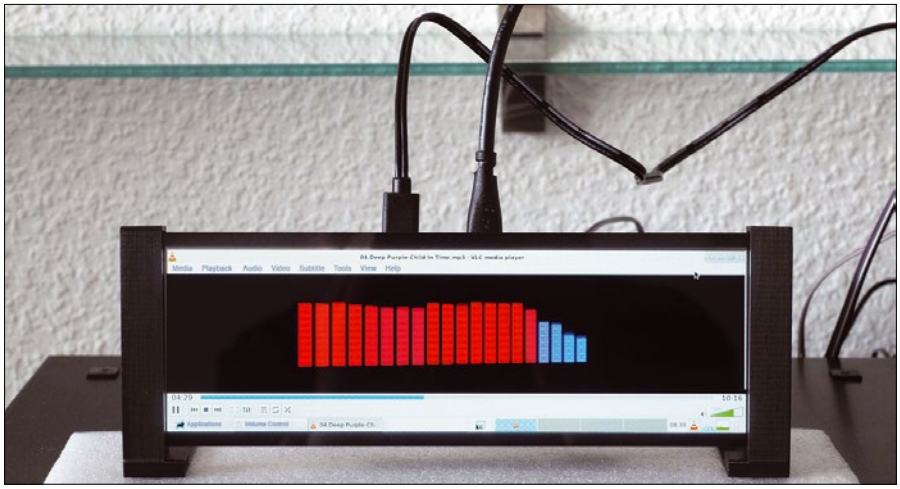


Figure 3: The Waveshare display with its unusual form factor.

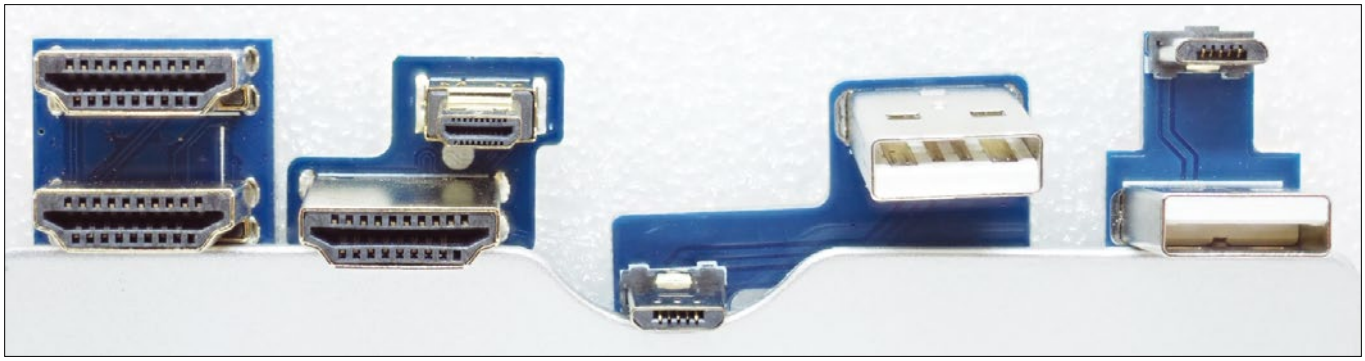


Figure 4: The display comes with some special adapters.

Listing 1: Touchscreen Configuration

```
# Match on all types of devices but joysticks
#
[...]
Section "InputClass"
  Identifier "libinput touchscreen catchall"
  MatchIsTouchscreen "on"
  Option "CalibrationMatrix" "0 -1 1 1 0 0 0 0 1"
  MatchDevicePath "/dev/input/event*"
  Driver "libinput"
EndSection
[...]
```

Gearing Up

The display works with a normal Raspberry Pi OS, but its unusual dimensions necessitate a number of settings, which the device wiki [2] describes quite well. The (slightly adjusted) configuration of the first HDMI interface from the settings in the `/boot/config.txt` file is:

```
[HDMI:0]
hdmi_group=2
hdmi_mode=87
hdmi_timings=400 0 100 10 140 1280 ➥
  10 20 20 2 0 0 0 60 0 43000000 3
display_rotate=3
```

The default orientation of the display is vertical; the `display_rotate` parameter changes that – but only for the pixel output, not for the touch orientation, which has to be done by the X server (more about this later).

Because the Raspberry Pi 4 has two HDMI outputs, all special configurations for the 19.1cm display are below the `[HDMI:0]` filter. You can easily connect a normal display to the other HDMI port and set up the computer this way. The 400px height of the Waveshare screen is not enough for some dialogs; the *OK* and *Cancel* buttons often disappear irretrievably at the bottom of the screen.

Later, when used as a web radio, the second HDMI port is not connected to a screen at all, but to the AV receiver. More settings are then needed:

```
[HDMI:1]
hdmi_group=1
hdmi_drive=2
hdmi_force_hotplug=1
hdmi_ignore_edid=0xa5000080
hdmi_force_edid_audio=1
hdmi_ignore_cec_init=1
config_hdmi_boost=4
```

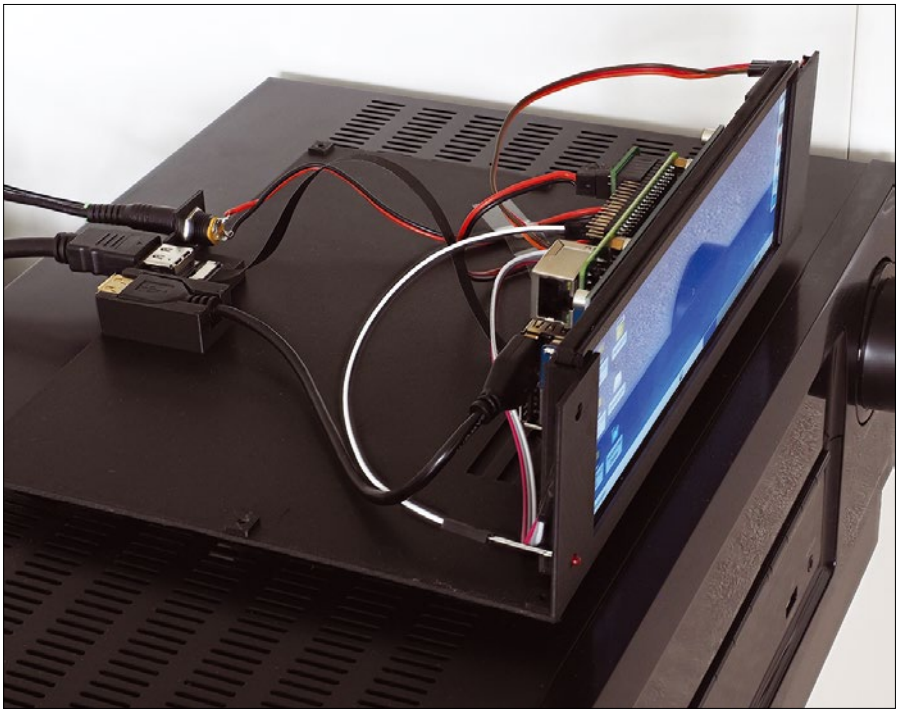


Figure 5: The case with the cabling for (left to right) USB, HDMI, and power.

In particular, they ensure that the output is configured correctly for audio output, even if the receiver is not yet running.

You can correct the touch orientation from the `/etc/X11/xorg.conf.d/40-libinput.conf` configuration file. Add a new `Section` with the lines from Listing 1 (to match the rotation from `/boot/config.txt`). The X server's input framework then controls the orientation with `CalibrationMatrix`. The Waveshare wiki provides the correct values.

The Raspberry Pi 4 offers three options for the sound output: analog output through the jack socket and two digital outputs via HDMI. The commands

```
$ pactl set-default-sink 1
$ pactl set-card-profile 0 off
$ pactl set-card-profile 2 off
```

give you the correct configuration for the PulseAudio version used by Pi OS as the default. You can also do this in the graphical user interface (command `pavucontrol`), but it is not so easy on the touchscreen. Annoyingly, many applications remember the output channel you use, so it is important to set up the configuration as early as possible.

With a working screen and sound output, the purely technical side of commissioning is now finished, but a nice screen with a mess of tangled cables is not something you would want in your living room.

In the Box

One of my goals for the project was, as I said earlier, to create a device suitable for the living room. And that means investing in a case; because I have access to a 3D printer, this is theoretically not a big problem. However, some pitfalls lurk compared with the basic setup of the operating system, as you can easily see from Figure 3, where the power connector and the HDMI cable for the sound output are jutting out vertically from the screen.

Preferably, all of the Pi's ports would be on the back of the case, but because USB and HDMI ports have to be installed "inside" the display, this is not possible by design. The problem has been around since the very first Raspberry Pi, independent of the Waveshare display. A Raspberry Pi in a

larger case always makes for a tricky install, because the important ports (if you count the SD card) are distributed over three sides.

Fortunately, some simple solutions are available. For example, I have never had a microSD card die on me. Once installed, it just gets on with its job and does not need to be accessible from the outside. Moreover, the LAN is not needed for the web radio, but if you do need the connection, simply buy a network coupler for a little money. On the inside, you need a short patch cable to the coupler.

The power supply is easy to solve, because you can also power a Raspberry Pi over pins 2 and 4 (5V) and 6 (GND), bypassing the built-in protection mechanisms with acceptable risk. Depending on your preference, a micro-USB, USB-C or barrel jack is then connected to the back of the case, which you connect to the pins with cable on the inside.

When it came to USB, I decided to go for a mini-extension (plug to socket) on the back of the case. Because this presents four USB ports, two are free for other purposes (e.g., an internal hard drive). A socket at the front wouldn't be bad either, but the 3D printer I have access to doesn't give me the scope to do that; the screen itself is 19cm wide, and with 3cm of margin on each side, this fully utilizes the printer's available width. In Figure 5, you can see the chassis base plate with the connectors.

The second HDMI port ultimately turned out to be the biggest challenge. It is only possible to route the cable out neatly to the back with a suitably angled plug. Because I have a Bluetooth-enabled receiver, I don't actually require the connection to send tracks, but the Raspberry Pi has never had a reputation for a stable, automatic Bluetooth connection, which is why this solution is a backup at best.

Finally, I found what I was looking for on BerryBase [3]: HDMI cables that you can assemble yourself, including sockets and plugs and even angled versions (Figure 6). A regular socket was attached to the other side of the ribbon cable, but it does not have mounting holes. Precise 3D printing dimensions without play then ensure that the jack is held to the back of the case. The construction does look a bit untidy, because the ribbon

Figure 6: HDMI flat ribbon cable by BerryBase.

cable is not shielded at all. My Denon receiver, which is otherwise extremely sensitive and refuses to cooperate with various cables, fortunately had no problem with that.

Finishing Touches

A button on the case for the on/off switch would not be a bad idea and is not much trouble if you are happy with something basic. The solution is simply to connect a button to GPIO3 and GND and add the line `dtoverlay=gpio-shutdown` to `/boot/config.txt`. Of course, this minimal version does not resolve the problem of standby power draw, but if you have a switchable power strip anyway, this worry can also be avoided.

Besides the button, the case also has an infrared receiver and a power LED. Both are optional, but useful, especially because my existing web radio software already supports the infrared remote control. In the meantime, the push button has been replaced by a touch sensor, but the interaction with GPIO3 then only works with an inverter in the middle.

All in all, the case design consumed a lot of time, and because of the size, the Prusa MK3S 3D printer I used also reached its limits. For example, printing takes up almost the entire area of the print bed (20x25cm of 21x25cm), so I had problems with warping at the corners – the bed heater doesn't work optimally at the edges. Printing the two case parts took more than 10 hours each. Thanks to targeted test prints of critical parts, however, one print run each was sufficient for the final version.

I provide full details of the system configuration and the case models in my project repository [4], but because of the tight tolerances and matching to the available components, anyone who wants to take on the design certainly has some work ahead of them.

Software

The hardware is now neatly configured and has found an attractive home. Next, I introduce the software for the web radio, with some adjustments that not only concern the change from a 4x20cm display to the large Waveshare screen, but that also relate to basic operation. After all, the idea is not only to operate the web radio from the touchscreen in the device itself, but to use a matching remote and a fancy web interface.

The software presented here, although intended for the luxury living room appliance (Figure 7) with Waveshare's touch-enabled widescreen, also does its job independently of this particular hardware. Implemented in Python, it runs on any Linux system, from desktop to Pi Zero. With a few minor hacks, it should also be possible to run the program on Windows; the only requirement would be a working audio configuration for the sound output.

Figure 7: Raspberry Pi-based web radio for the living room system.

Figure 8: Virtually every radio station today offers the option of listening in a web player.

Before I get started with the installation and configuration, I'll take a look at some of the technical details of web radio.

The Longest MP3 in the World

Web radio, aka Internet radio, transmits (streams) a seemingly infinite MP3 file over the Internet. Instead of storing the file and then playing it back, the player accepts the stream directly and plays the audio data.

The player software can therefore be anything that calls up URLs and plays the received data directly, especially a web browser. Because virtually every radio station is now available on the Internet, you will often find a button for live playback of a program displayed prominently on the site (Figure 8).

The simplest web radio comprises a device with a browser (e.g., a retired smartphone) and a collection of links to your favorite stations. You don't even have to compile and maintain the link collection yourself because many portals [5] on the web take care of this detail [6] and also offer apps for iOS and Android. Because many AV receivers can also receive music as a Bluetooth stream, an almost complete solution in the simplest case is for the smartphone to pass the received stream to the receiver by Bluetooth.

Totally Happy?

The solution outlined above has some disadvantages. First, portals look to earn money and therefore regularly treat users to advertising spots. Second, many stations also transmit interesting additional data about the tracks and artists, which a plain vanilla music player does not reproduce (Figure 9). Thirdly, the portals usually offer only very limited options for recording their programs.

In principle, the solution consists of its own collection of links that bypasses portal and station websites and taps directly into the music source. Everything else about program logic is only for convenient control and navigation, as well as the appealing display.

A little research on the web turns up a number of web radios for the Raspberry Pi. The most comprehensive collection is provided by a web radio connoisseur named Bob Rathbone [7]. Besides the software, the project provides a 340-page document that explains the installation and configuration of various hardware and software components on the Raspberry Pi down to the minutest detail. Even if you have never owned an SBC like this, you can build a web radio without any problems, provided you separate the wheat from the chaff, which is not so easy.

However, Rathbone's solution relies on difficult-to-install software with a less than appealing interface, which ultimately pushed my decision in favor of designing a DIY solution.

Installation

For an initial test you will want to use a new, freshly installed Raspberry Pi OS – make sure you have the latest version (at the editorial deadline, this was "Buster"). A Pi Zero with the Lite version is perfectly okay for the project, unless you want an additional screen for the browser, as in the living room radio presented here. You also need the normal desktop version of the operating system and at least a second-generation Raspberry Pi, preferably later, because the graphical interface on the earlier versions is a little rough.

The installation commands are as follows:

```
$ sudo apt-get -y remove pulseaudio
$ git clone https://github.com/➥
  bablokb/pi-webradio.git
$ cd pi-webradio
$ sudo tools/install pi
$ sudo systemctl ➥
  enable pi-webradio.service
$ sudo systemctl ➥
  start pi-webradio.service
```

You only need the first call for desktop systems, because PulseAudio, which is installed there by default, does not support audio output from a background service. If the default user on

Figure 9: Example text info supplementing a radio broadcast.

Under the Hood

The Pi web radio software is based on Python 3 and uses the `mpg123` command-line tool to play the audio data. The program provides a remote control mode with the `-R` option. You can easily try this out by calling

```
mpg123 -R
```

at the command line and then typing *h* – the program will then display all commands.

The web server used here is Flask, a micro-framework that goes about its task in many Python application servers. The web radio uses only a few of its functions, because the web server just passes API calls like `radio_play_channel` to the appropriate Python class, of which there is only one per task (e.g., `Radio`, `Player`, or `Recorder`). Some auxiliary classes are available, as well.

The web interface uses CSS3 and JavaScript; jQuery acts as an auxiliary library. The application doesn't go all out to support all possible browsers, but today's crop of web browsers all offer what it takes.

Figure 10: Channel selection: Note that you can localize your station selection in the software.

your system is not named *pi*, you also need to add the appropriate user to the installation command (command 4), who needs to be a member of the *audio* group and have `sudo` rights. After the install, activate and start the service (last two commands). After doing so, you can access the web controls in a browser.

To get a landscape aspect, I optimized the responsive interface. On typical desktops, smartphones, and tablets, the system can be operated in portrait format without any problems. By the way, the web server built into radio software delivers exactly one web page to the browser once only. After that, the server and browser only exchange small blocks of data; the web browser takes over the display completely, ensuring that the server part of the software will run without problems, even on SBCs like the Pi Zero. A few details relating to the technical implementation can be found in the "Under the Hood" box.

Configuration

By default, the tool already supports a number of German stations, but for copyright reasons you do not get to see the station logos on launch (Figure 10). The broadcasters probably don't care, but dubious lawyers take every opportunity to earn money, and I don't want to give them any reason to sue me. It takes some configuration work to make the web radio serve your musical tastes and also present the station selection in a visually appealing way.

The channel list itself can be found in the `/etc/pi-webradio.channels` file. It stores the name, a URL, and a logo (without path) for each channel. After clicking on the desired station, you can download and save the logos by right-clicking. The files in JPEG or PNG format belong in the `/usr/local/lib/pi-webradio/web/images/` directory; you then need to enter the names of the logo files in the station list.

You need a bit more detective work for a broadcaster's URL, because not all broadcasters publish them, and those that do obfuscate the info as well as they can on their websites. The IT horizon of some broadcasters obviously does not go beyond their own web presence. If you can't find the URL despite an intensive search, you can still use the developer tools in your browser. First, navigate to the radio station page that supports live listening, but do not click the play button.

First launch the developer tools from the menu or by pressing F12. Another window opens, in which you navigate to the *Network* tab. Then press the live play button. The page now fills with all the URLs that the browser retrieves (Figure 11).

When you get there, search for a URL containing `mp3`, `m3u`, `stream`, `icecast`, or `shoutcast`. The list is not exhaustive, but almost always takes you to the target. Copy the URL and start the playback with `mpg123 <URL>`. The `mpg123` playback program does not handle encryption, so you need to replace a leading *https* in the URL with *http* up front. If you enter the correct URL, you should now hear the station playing. Press *Q* to cancel the playback.

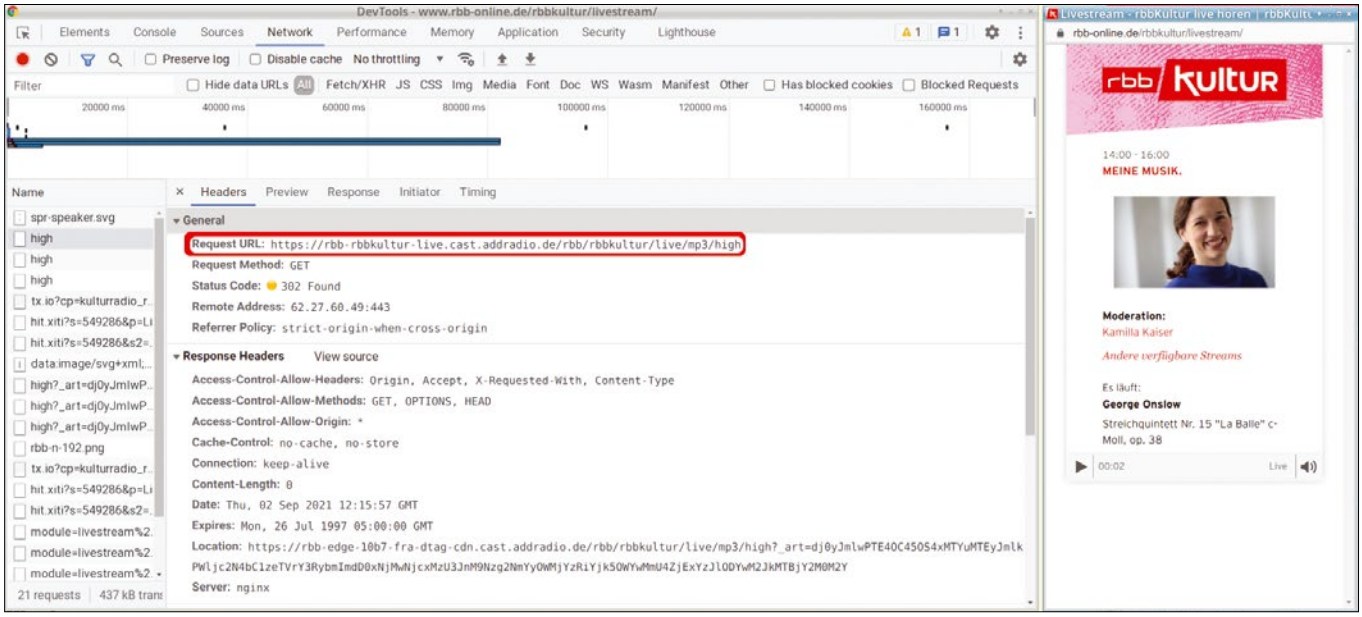


Figure 11: The developer tools in the browser reveal the URL for the media stream.

Besides the `/etc/pi-webradio.channels` file, in the `/etc/pi-webradio.conf` configuration file you can change the port for the web server (default = 8026), for example, or specify the directory in which you will be storing any recordings.

Operation

After configuring the stations, restart the service by typing,

```
sudo systemctl ➥
  restart pi-webradio.service
```

then navigate to the address *http://<Host>:8026/* in the browser. If the browser runs on the same system as the service, replace *<Host>* with *localhost*; otherwise, enter the IP address or name of the corresponding Raspberry Pi.

Operating the web radio is largely self-explanatory. On the left side, icons let you toggle the time, player, station selection, file selection, and tools views (see Figure 9). The Tools page is where you can stop the application, reboot, or shutdown the system. The File selection lets you play MP3 files.

The application can also record broadcasts on the fly if you press the Record button (far right in the button-bar below the station logo). The recording will then run as long as you specified in `/etc/pi-webradio.conf`. In addition to spontaneous recordings, web radio also supports scheduled recordings, even independent of the Pi web radio service. To do this, install the *at* package and schedule the recording with a one-liner:

```
# echo /usr/local/bin/pi-webradio.py ➥
  -r 4 120 | at 20:00 31.10.2021
```

The command records channel 4 for 120 minutes. The `at` daemon takes care of starting on time. The `atq` command will show you the recording queue, and `atrm <number>` lets you delete jobs. You might want to define regular recordings in the crontab or as systemd timers if you prefer a more modern approach.

Challenges

An integrated solution with a server and browser on one system (Figure 7) poses a few additional challenges. First, is the issue of starting the browser as an application (i.e., without the usual frames, menus, and buttons of a normal web browser). Also, you want the program to launch in kiosk mode, which prevents the user from exiting; for example:

Listing 2: Desktop Autostart File

```
01 [Desktop Entry]
02 Name=Pi-Webradio
03 Comment=Webclient (Chrome) for Pi-Webradio
04 Exec=/usr/local/bin/webradio_chrome.sh
05 Terminal=false
06 Type=Application
07 StartupNotify=false
08 X-GNOME-Autostart-enabled=true
```

```
$ chromium-browser ➥
  --app=http://localhost:8026/ ➥
  --kiosk
```

To run the browser at boot time, copy the desktop file from Listing 2 to the `$HOME/.config/autostart/` directory. The name does not matter, but the file needs the `.desktop` extension. The call to the browser is scripted in line 4; the script also terminates the browser later when you stop the radio server from the Tools page.

Another challenge is to hide the mouse cursor. To do this, you need the *unclutter* package, which takes care of the appropriate call in the startup script. The script also disables any power-saving options that would cause the screen to switch off.

Alternative Hardware

Not every piece of hardware in the Raspberry Pi universe is suitable for a web browser, which is why the Pi web server also provides a simple web API for control. The program `webradio_cli.py` allows complete control from the keyboard – or remotely, if desired.

The project also provides a minimalist control program for Pimoroni's Pirate Audio HATs with four buttons and a miniscreen (Figure 12). The HATs are available in different versions, among others with mono speaker or with headphone output. All variants come with a 240x240px screen and four buttons.

Again, the server and the client run on the same hardware. The client monitors the four buttons and uses them to change the station and volume. The display shows the logo of the current station. If you change the station from the web interface (e.g., from a smartphone), the client also notices this and updates the image.

The program for the Pirate HAT is a good starting point for DIY projects with other hardware, such as classic 4x20cm LCDs. More info can be found in my project repository [8].

Conclusions

Today's displays are increasingly coming with HDMI connections, even in small form factors, which not only simplifies configuration, but future-proofs your projects, because they do not require any manufacturer-specific special drivers. The touch feature also works without any worries thanks to the Raspberry Pi's comprehensive set of drivers.

Because of its special format, the Waveshare display presented here is suitable for more than use as web radios. It also cuts a fine figure as a control panel, oversized remote control, or system monitor.

Figure 12: The software can also run on a Pirate Audio HAT by Pimoroni.

In the case of the web radio project, however, the experience is typical of the Raspberry Pi: The peripherals often cause you more work than the actual hardware. Whether the Raspberry Pi or the monitor, the building blocks are not developed with a big picture in mind. Thanks to 3D printing technology, though, it is now far easier to create a suitable case than it used to be.

Author

Bernhard Bablok works at Allianz Technology SE as an SAP HR developer. When he is not listening to music, cycling, or walking, he focuses on Linux-related topics, programming, and small computers. You can reach him at *mail@bablokb.de*.

The Pi Web Radio software itself is quick to install and get up and running. If you value a special appearance, a bit more work is involved – the device shown in Figure 7 kept me busy for several months. On the website run by Bob Rathbone, which I mentioned earlier, you will find a gallery of neat looking radios as inspiration for your own hardware projects. ■■■

Info

[1] Sources for the original web radio project: *https://github.com/bablokb/simple-radio*

[2] Wiki for the 7.9-inch display: *https://www.waveshare.com/wiki/7.9inch_HDMI_LCD*

[3] BerryBase: *https://www.berrybase.de/en#*

[4] Project repository for the new web radio: *https://github.com/bablokb/pi-webradio*

[5] Portal with public and private stations: *https://www.radio.net*

[6] Multinational radio portal: *https://www.radioguide.fm/countries*

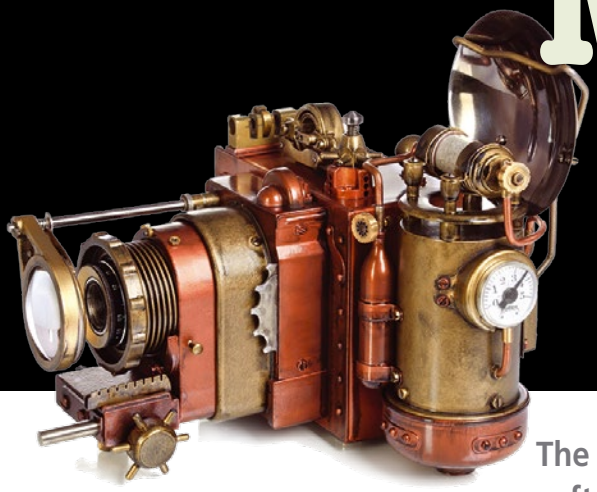
[7] Detailed web radio construction guide: *http://www.bobrathbone.com/raspberrypi/pi_internet_radio.html*

[8] Pi web radio: *https://github.com/bablokb/pi-webradio*

MakerSpace

Continuous shooting for detailed photography

On the Table



The combination of a web camera, Raspberry Pi, and simple software facilitates miniature tabletop photography.

By Günter Pomaska

Photographers often rely on tabletop photography to take product photos for advertising campaigns or illustrations for magazines. Even a digital camera or smartphone is up to this task, but you have good reasons to look into USB and web cameras, as well. Armed with a 4K USB camera, a focus distance of 3cm, an integrated zoom function, and a Raspberry Pi with USB 3.0 ports, tabletop photographers can embark on their mission.

Raspberry Pi Cameras

Currently, the Raspberry Pi Foundation officially supports the V2 camera module (also available in the NoIR variant) and the Pi High Quality (Pi HQ) camera. The V2 module was introduced as the successor to the V1 module in 2016. The predecessor was based on an OmniVision 5647 sensor with a resolution of 5 megapixels (Mpx). In contrast, a Sony IMX219 sensor with 8Mpx is used in the V2 camera [1]. In contrast to the regular module, the NoIR camera lacks an infrared filter but otherwise has the same features [2].

The cameras have a fixed focus starting at a distance of about 70cm to the subject and with a fixed focal length. They are connected to the computer with a ribbon cable via the camera serial interface (CSI). The multimedia abstraction layer (MMAL) and Video4Linux (V4L) APIs are then used for access. You can control the cameras directly with console commands, and you have an extensive Python library for camera control with the *picamera* package [3].

The Pi HQ camera [4] lets you attach C- or CS-mount lenses with the appropriate adapter. The sensor is a 12Mpx Sony IMX477R chip. The sensor measures 7.9mm diagonally and features a resolution of 4056x3040px. The 6mm lens covers the close-up range nicely, thanks to the short focus distance. However, the manual focus and aperture settings have a disadvantage.

An extensive repertoire of lenses from macro to telephoto is available for both the V2 modules and the Pi HQ. Moreover, the Raspberry Pi cameras offer solutions for a wide range of video applications. I look into what you can achieve with USB cameras in the field of still images, paying special attention to miniature photography.

Web and USB Cameras

Modern webcams stream their signals to a computer over USB and range from inexpensive, entry-level models to high-quality cameras for industrial image processing. All serve their purpose, but good image quality has its price.

In experiments I conducted, the Microsoft LifeCam Cinema [5], the C920 HD Pro [6] (full high definition, FHD), and the Brio [7] (4K) from Logitech showed good results. The Brio is perfectly suited for the intended application, with its high resolution and is therefore used here. Other products are definitely also suitable for macro photography, but you need autofocus switched off, a short focus distance in the centimeter range, the ability to zoom, and compliance with the USB video class (UVC) standard, a specification for devices that stream video data over USB.

Webcams typically meet the UVC requirement and do not require a special driver to operate on a computer. Instead, a unified UVC driver serves devices from different manufacturers.

V4L, or the second version V4L2, is a collection of device drivers and programming interfaces already integrated into the Raspberry Pi's operating system and also supported by the Open Source Computer Vision (OpenCV) library of algorithms for automated image processing and artificial intelligence for face or gesture recognition. Working at the command line, you can control UVC devices with the `v4l2-ctl` command. The command

```
$ v4l2-ctl --list-devices
HD Pro Webcam C920 ➥
  (usb-3f980000.usb-1.2):
        /dev/video0
```

shows which device file belongs to which device. For the Logitech Brio, the device is `/dev/video0`, as shown in the output.

Adding the `--list-ctrls` option tells `v4l2-ctl` to output information on the configurable controls (Listing 1). You need this data for your own application software. The output shows the value ranges and defaults for the camera and exposure.

The `gv4l2` tool and Guvcview webcam application, which you can install with your package management system, provide graphical interfaces that are helpful for fine-tuning the desired camera settings.

Listing 1: Reading the Controls

```
$ v4l2-ctl -d /dev/video0 --list-ctrls
brightness 0x00980900 (int) : min=0 max=255 step=1 default=128 value=128
contrast 0x00980901 (int)   : min=0 max=255 step=1 default=128 value=128
saturation 0x00980902 (int) : min=0 max=255 step=1 default=128 value=128
[...]
zoom_absolute 0x009a090d (int)  : min=100 max=500 step=1 default=100 value=500
led1_mode 0x0a046d05 (menu)     : min=0 max=3 default=0 value=3
led1_frequency 0x0a046d06 (int) : min=0 max=255 step=1 default=0 value=0
```

Tabletop Photography

A table, some paper as background, LED lighting, a few clamps, and a tripod are all the hobbyist needs as an improvised tabletop studio. A scene with interesting objects and unusual positions can give you attractive images. Tabletop plays a significant role in product photography and advertising. Amateur photographers use this technique for collectibles or for selling on online portals.

Macro photography refers to image scales greater than 1:1. In other words, an object is depicted on the sensor larger than in the original. The definition comes from the analog age. Smaller image scales belong to the close-up range, but if everything fits on the table, it is tabletop.

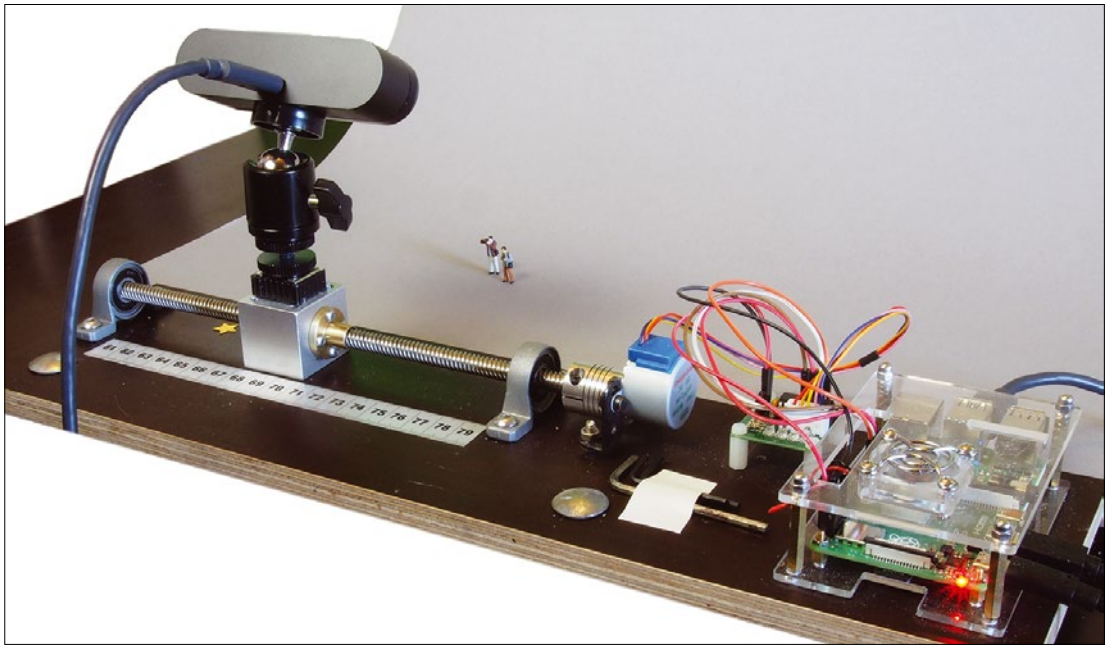


Figure 1: The Logitech Brio on a linear guide. The Raspberry Pi controls the motor.

Figure 2: A miniature scene taken on a tabletop setup featuring tiny figures on a scale of 1:87.

Whether you use bellows extensions, macro lenses, or extension rings, a photo's range of sharpness at close range is limited. Photographers therefore use the focus bracketing technique and compile an image from a series of shots with different depths of field. This stacking technique is already integrated in the firmware of modern digital single-lens reflex (DSLR) cameras.

Smartphones also use the Dynamic Depth format for 3D photos, which are popular on Instagram and the like. The technology makes it possible to create images with smartphones where, for example, the background of a portrait appears blurred, much as DSLR cameras with fixed focal length lenses depict a person with an open aperture. The series of photos taken with a webcam is sent to a stacking tool such as MacroFusion [8] for postprocessing.

For the shoot in this article, the setup additionally relied on variable camera positioning. The camera was mounted on a linear drive and, driven by a stepper motor, moved to different positions under program control (Figure 1), which enabled, for example, video and continuous shooting, lenticular images (images that change depending on the viewing angle, giving the impression of movement), and stereo imaging.

Occlusions in the miniature scenes can be controlled by minimal camera movements. A typical scene of miniature photography, with model figures in 1:87 scale built with tweezers, is shown in Figure 2. The camera can move past the set to find the most favorable position. In software terms, this is one more loop, and the Raspberry Pi hardware is perfect for this task.

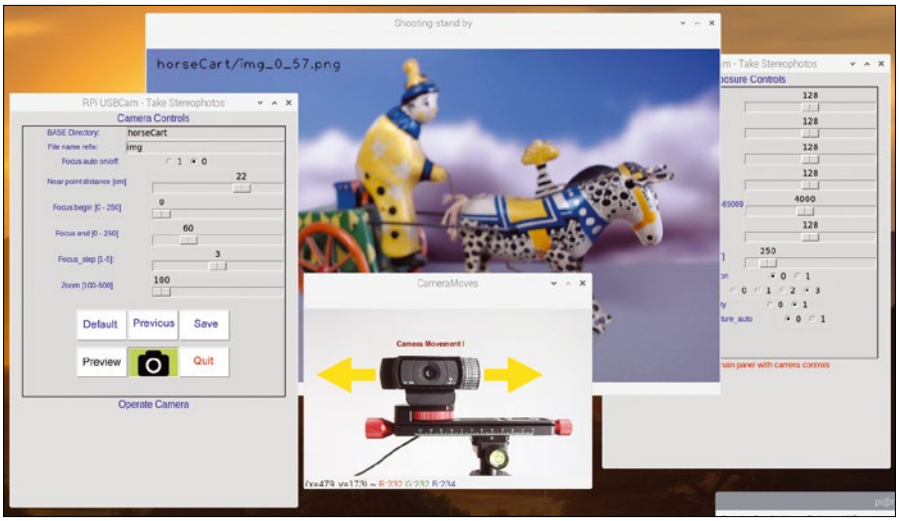


Figure 3: The graphical interface of the DIY camera control.

Listing 2: usbCamMover.py – Initialization

```
[...]
cam0.set(6, cv2.VideoWriter_fourcc('M', 'J', 'P', 'G')) # setting MJPG codec
#width    =  1920 # HD
#height   =  1080 # HD
width     =  3840 # UHD (Brio, USB3)
height    =  2160 # UHD (Brio, USB3)
cam0.set(3,width)
cam0.set(4,height
[...]
```

Continuous Shooting

Programs like Guvcview are suitable as a test tool but cannot control specific processes, which means you need your own software with a graphical user interface (GUI). The programming language of choice is the object-oriented Python 3 with its numerous practical libraries for designing graphical interfaces, such as the *guizero* library, which is easy to use. The stepper motor is controlled by the Python RPi.GPIO module, whereas OpenCV takes care of capturing and saving the images.

The software's self-made GUI is located in the `usbCamMover.py` file and comprises two panels: one for exposure settings and one for camera settings, including zoom and focus (Figure 3). I will not go into the programming in detail here; the software, which is still a bit of a hack, can be found on the *Linux Magazine* FTP site [9].

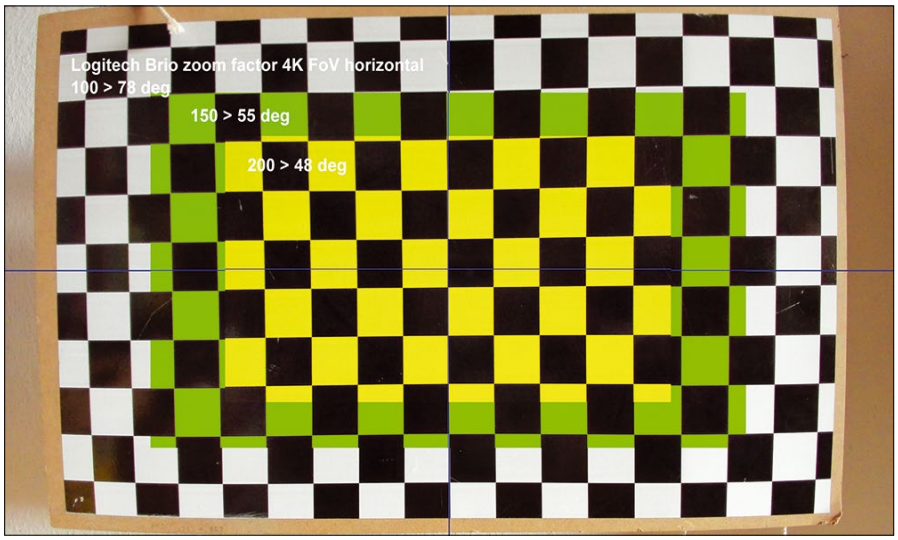


Figure 4: Zoom factors 100 (white), 150 (green), and 200 (yellow) for Logitech Brio 4K resolution.

Camera Control

The settings in the Camera Controls dialog influences the image geometry. The field of view (FoV) is limited by the *Zoom* factor. The number of shots required depends on the focus range and its step size. The horizontal camera position is controlled by the distance of the camera to the object. A list of factors for the travel distance is embedded in the program.

The accepted *Zoom* factor for the Brio is between 100 and 500 with a given step size of 1. At 4K resolution (i.e., 3840x2160px), a horizontal field of view between 78 and 43 degrees can be achieved. However, zoom factors larger than 200 no longer have an effect at ultra-high definition (UHD) resolution.

The UHD resolution of the Logitech Brio is only offered with the MJPEG pixel format. For this, the code in Listing 2 adapts the initialization with the variables `width` and `height` to suit the connected camera.

The zoom factor in FHD resolution (1920x1080px) is different. With factors up to 200, the FoV changes to the same extent as with the higher resolution, but if you go beyond that, it can narrow the angle by up to 18 degrees (zoom factor 500). Figure 4 compares the image crops at 4K resolution.

At full FoV, a high lens distortion is noticeable in the peripheral area,

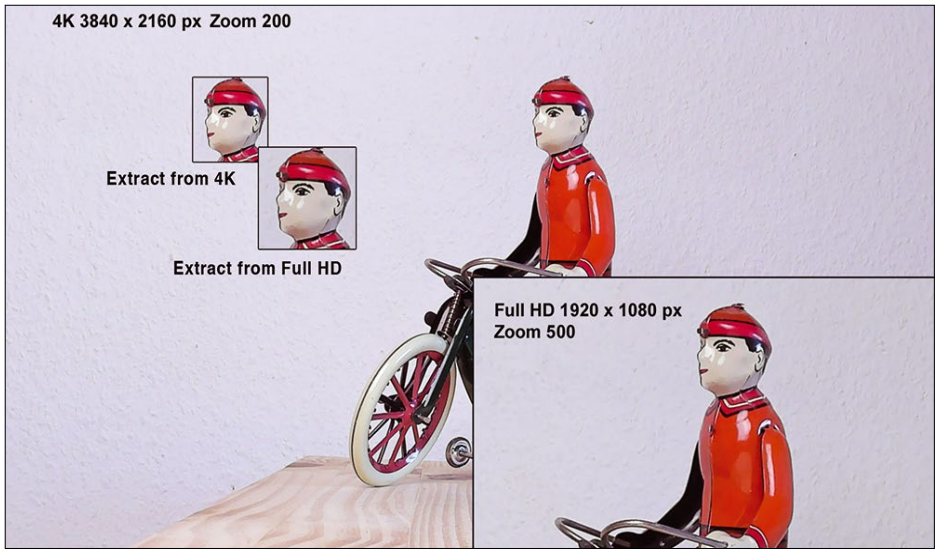


Figure 5: A comparison between resolution and maximum zoom factor. In some situations, it is more convenient to work with lower camera resolutions.

Figure 6: The focus in this shot was between 3cm and 8cm.

Figure 7: Focus planes in the close-up range. To get all the figures in focus, you must process several shots together.

which might need to be corrected by optical inspection or mathematically with the camera's calibration data. Even a low zoom factor reduces the extreme distortion.

The HD, FHD, and UHD camera resolutions each result in an image in 16:9 format. The relevant image area does not always fill this format, so you often work with image crops in postprocessing.

Listing 3: usbCamMover.py – Saving Frames

```
[...]
for i in range(4):           # Read buffer
  cam.grab()
ret, frame = cam.read()      # Take a picture
cv2.imwrite(img_name, frame) # Save image
[...]
```

Therefore, it is necessary to weigh between object distance and zoom factor the extent to which optimum resolution can be achieved.

If you can get close enough to the object, it's best to opt for 4K resolution. If you need to get farther away, consider the crop factor. If necessary, switch to 2K resolution with a smaller FoV.

For example, in Figure 5, only the cyclist's face is of interest. With 4K resolution and zoom factor 200, you capture too much unneeded information. At an identical shooting distance with 2K resolution, a smaller section can be captured with a zoom factor of 500, resulting in a higher resolution for the image area of interest. In general, however, the USB camera is only significant for still images with the 8Mpx sensors.

Focus Range

Another aspect of the rational image acquisition technique is the focus range. The parameter values are between 0 and 250 with a default step size of 1. Too many exposures would overload the stacking software, so it is best to limit the range and thus sensibly minimize the number of shots.

An example about 2cm (close-up range) from the object and focusing with a value of 220 becomes interesting, whereas a parameter of

Figure 8: Comparison between automatic and manual white balance (WB) shown with Captain Scarlet figures as an example.

around 110 is useful up to about 8cm. With a step size of 5, 22 exposures were created for the builders in Figure 6 in the close-up range. The figure shows the results after stacking.

If you increase the distance to the subject to about 10-40cm, focus settings from 40 to 100 on the right side are recommended. In Figure 7, the first red doll was focused with a value of 79 and at a distance of about 17cm. A range of 40cm therefore translates to about 200 focus units; given a step size of 5, this corresponds to 40 exposures. You therefore have a defined design framework that you can validate in a preview.

Exposure Control

In conventional photography, exposure is controlled by aperture, ISO, and shutter speed, plus white balance and color palette. Tabletop photography is mostly about static scenes, so you do not need to influence exposure time directly.

Despite the webcam's good adaptation to ambient light, you will want to use several light sources to ensure optimal illumination of the scene. You can do without the automatic mode, especially for continuous shooting. If possible, leave brightness, contrast, saturation, and gain (comparable to ISO) at medium values.

Color adjustment is mainly achieved with white balance; make sure you do not tweak the settings too quickly. Gradual, small changes will give you the best results. Figure 8 shows an example of the differences between automatic and manual white balance.

The automatic white balance delivers very cold colors, and the break in the background can still be seen. A color temperature of 5500 Kelvin improves the white areas in particular. The background is also now evenly black thanks to the manual white balance. Some caution is required with saturation; otherwise, you can easily go too far.

Processing Continuous Shooting

After you have completed the image settings, let the camera do its work. In the example, I used a webcam with rolling shutter, which streams to the USB port. The image data is buffered for transfer. For a still image, the application reads individual frames from a buffer area (Listing 3).

The image is saved, the focus setting is updated, and the next image follows. An inner loop iterates through the focus range, whereas, in an outer loop, the program moves to the capture positions. Finally, it returns the camera to the starting point and asks how to proceed. The file references comprise the directory name, the project name, a prefix, the camera position, and the focus value, followed by the image format. This way, each image can be mapped uniquely.

You can run the program on the Raspberry Pi with HDMI and a mouse as a standalone setup, but it is better to use VNC on the host. You would be transferring the image directories to the host anyway with an FTP client so you can run an image processing program for stacking.

Conclusions

The first webcams transmitted images of a coffee machine at Cambridge University in 1991 and went online starting in 1993. The now world-famous coffee machine was auctioned off in 2001, repaired by the manufacturer, and on the web until 2015. Today, people prefer to stream urban situations, or participants in an online conference put themselves in the picture.

Now that webcams deliver a resolution of 8Mpx and many industrial applications work with USB cameras, you can give them a try in photography. Miniature and tabletop photography with continuous shooting are options; easy handling, program control, and a decent image quality make the venture seem promising. A Raspberry Pi 4 controlling the setup rounds off the tabletop scenario. ■■■

Info

[1] Pi Camera Module 2: *https://www.raspberrypi.org/products/camera-module-v2*

[2] Pi Camera Module 2 NoIR: *https://www.raspberrypi.org/products/pi-noir-camera-v2*

[3] Pi Camera: *https://picamera.readthedocs.io*

[4] Raspberry Pi High Quality Camera: *https://www.raspberrypi.org/products/raspberry-pi-high-quality-camera*

[5] Microsoft LifeCam Cinema: *https://www.microsoft.com/de-de/d/lifecam-cinema/94f910x3714n*

[6] Logitech C920 HD Pro: *https://www.logitech.com/de-de/products/webcams/c920-pro-hd-webcam.960-001055.html*

[7] Logitech Brio: *https://www.logitech.com/de-de/products/webcams/brio-4k-hdr-webcam.960-001106.html*

[8] MacroFusion: *https://sourceforge.net/projects/macrofusion/*

[9] Code for this article: *ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/254*

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

Expand your Linux skills with:

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

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

Subscribe now!

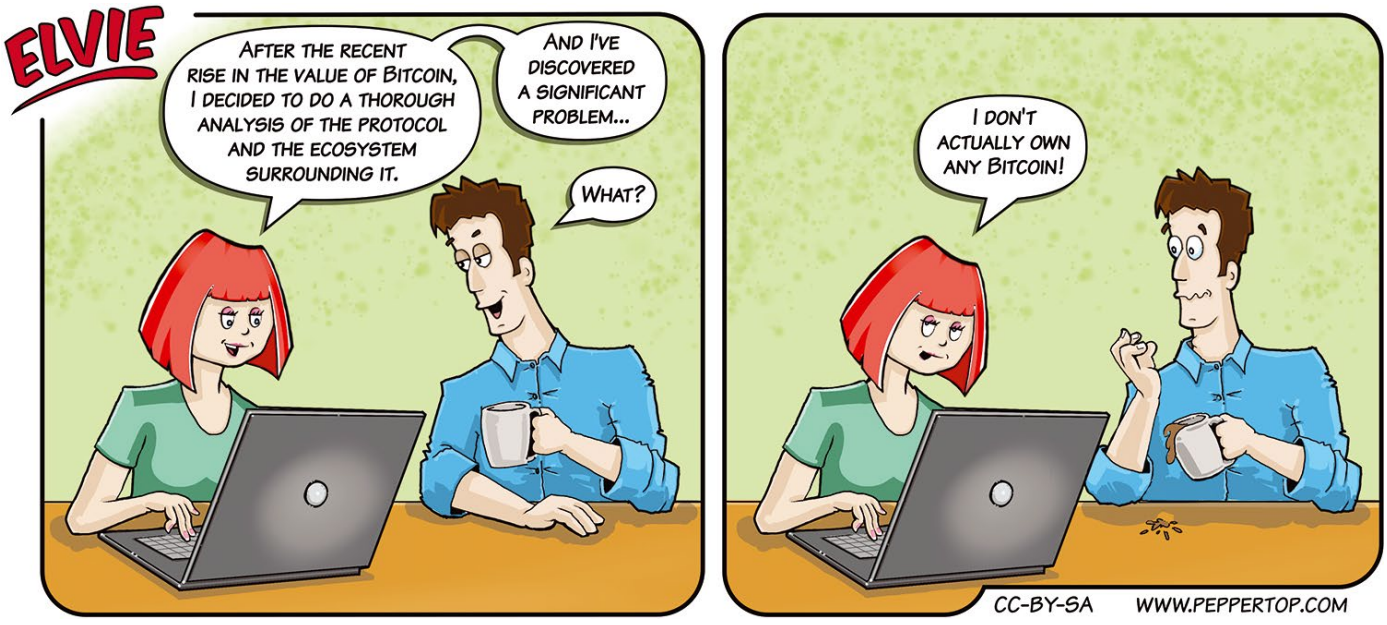
shop.linuxnewmedia.com/subs

GET IT NOW!
FAST DELIVERY WITH OUR PDF EDITION

Linux gaming has come a long way since the early days. In the beginning, it was just those first few native games. Then the Wine API brought Windows gaming to Linux. Later Steam added Linux support for its extensive gaming platform, and all the while, the retrogaming community was keeping the classic games of the past alive for Linux users. The Lutris open gaming platform ties these threads of the Linux gaming experience together in a single interface. Lutris uses Wine for Windows gaming and also supports the Steam, Goc, Battle.net, Origin, and Uplay gaming platforms, as well as offering support for native Linux games. The Lutris team also takes a special interest in the preservation of classic video games from the past. This month we introduce you to Lutris. Elsewhere in this month's issue, we show you how to back up your data with FreeFileSync and help you get started with managing your appointments from the command line.

Image © Olexandr Moroz, 123RF.com

LINUXVOICE



MADDOG'S DOGHOUSE

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

FreedomBox offers a private, secure personal server that is now available on more platforms and with its own base system from Olimex. BY JON "MADDOG" HALL

Privacy with a Home Server

Over two years ago, I wrote an article [1] about the FreedomBox Foundation and how the FreedomBox software [2] allows you to set up a private, secure personal server that can plug into an outlet in your home.

The project was started over 12 years ago by Dr. Eben Moglen, lawyer and law professor at Columbia University and author of the first version of GPL 3.x, among many other accomplishments. Moglen was concerned about people putting their most personal and private data on corporate clouds and sending the data over public networks.

Moglen conceived of a very small server that could be installed and used at home. He talked about it and people joined the project.

The goals were to have completely free software and open hardware – with no binary blobs that could hide trap doors and Trojan horses – which would allow long-term supportability. Another design goal was to make the software easy to install and configure with various privacy and security tools such as Tor, VNC, proxy software, and others.

Over the past two years, the FreedomBox software has been ported to many platforms, and they provide pre-compiled ISOs for many small, low-power single board computers (SBCs), as well as support various virtual machines and emulators. While installing and operating on the bare metal of an SBC would be the way I would normally run the code, installing it to an emulator or virtual machine might be an easy way to learn the system.

Also over the past two years, the software has been integrated with the Debian distribution of GNU/Linux (Debian 10 Buster) and so is now available on all of the platforms that Debian 10 supports, as well as quite a few SBCs (many of which require no binary blobs to run FreedomBox code), such as the BeagleBone Black, the Banana Pro, ROCKPro64, and more.

One company that has partnered with the FreedomBox organization is Olimex [3], who offers a relatively inexpensive base system called the Pioneer-FreedomBox-HSK.

The Pioneer comes with a suitable power supply and a choice of three different power plug types, and HDMI and USB 2.0 connectors for a display, keyboard, mouse, and other "slow" devices, as well as a 1 Gbps RJ-45 Ethernet port, 1GB of RAM, a 32-bit ARM processor, and a 32GB microSD card that holds the operating system. In addition the unit has a lithium battery that can keep the system going for up to three hours (useful for short electrical outages), as well as a very solid metal case with the FreedomBox logo.

Many of the SBCs do not have WiFi support, and many people might look at this as a deficit. But you could use a USB WiFi dongle or (more appropriately) use your WiFi router to distribute the data throughout your house.

If you want more storage for the FreedomBox server, the Pioneer comes with an eSATA connector, so you can hook up an external eSATA device, whether a docking station or a single storage device.

Relatively new is an Olimex metal storage bay that can hold a 2.5-inch SATA storage device. Your Pioneer slips right onto the top and connects directly to an eSATA connector on the bay. You can order the bay with several different storage sizes up to 2TB of HDD or 512GB of SSD, or you can order it with NO storage and use a 2.5-inch SATA storage device that you may have left over from some other laptop or project.

Olimex hardware is open hardware. They publish circuit diagrams and all the information necessary to maintain the system far into the future.

Some people might look at the specifications for this SBC and think that it is underpowered, or its memory is too small, but remember that it is a server, and for the most part it will be I/O bound, picking up blocks of data and pushing them out through the Ethernet port and out over the Internet, whether WAN or LAN. It needs no fan, makes no noise, and draws less power than an incandescent light bulb.

Once you have installed the microSD card that holds the system code and boot, you find that you access the FreedomBox through a web interface, and you really do not need any keyboard, mouse, or monitor connected directly to the FreedomBox itself.

FreedomBox documentation has become fairly complete and translated into many languages. The documentation is not only for its original direction as a personal server (for you or your family) but also for community use, especially if you combine some of the federated social media from *https://fediverse.party*, such as Mastodon, GNU social, diaspora*, and others. ■■■

Info

[1] "maddog's Doghouse – Freedom Box" by Jon "maddog" Hall, *Linux Magazine*, issue 226, September 2019, p. 68

[2] FreedomBox: *https://freedombox.org*

[3] Olimex: *https://www.olimex.com*

Turn Linux into a gaming platform with Lutris

Playful

Gaming on Linux has traditionally involved a complex installation and configuration procedure, until now. Lutris lets you enjoy games from different worlds quickly in a unified interface.

BY ERIK BÄRWALDT

Linux has not been most people's preferred gaming platform until now because many interesting and also graphically sophisticated games are only available for other operating systems. But with PlayOnLinux [1], a tool enters the scene that makes games developed for Microsoft systems usable under the free operating system with the help of the Windows runtime environment, Wine. PlayOnLinux is also aimed at users who want to use conventional Windows applications on Linux. It therefore also integrates numerous office applications and programs for productive use.

With the release of Lutris [2], there is now another solution that uses Wine for Windows games and also integrates native Linux games. On top of this, Lutris supports games that are distributed and managed via the Steam platform, and the program can handle other digital distribution platforms that offer game collections or retro games. In this way, Lutris opens up a variety of integration options for games from a wide range of genres to serious gamers looking for a uniform interface.

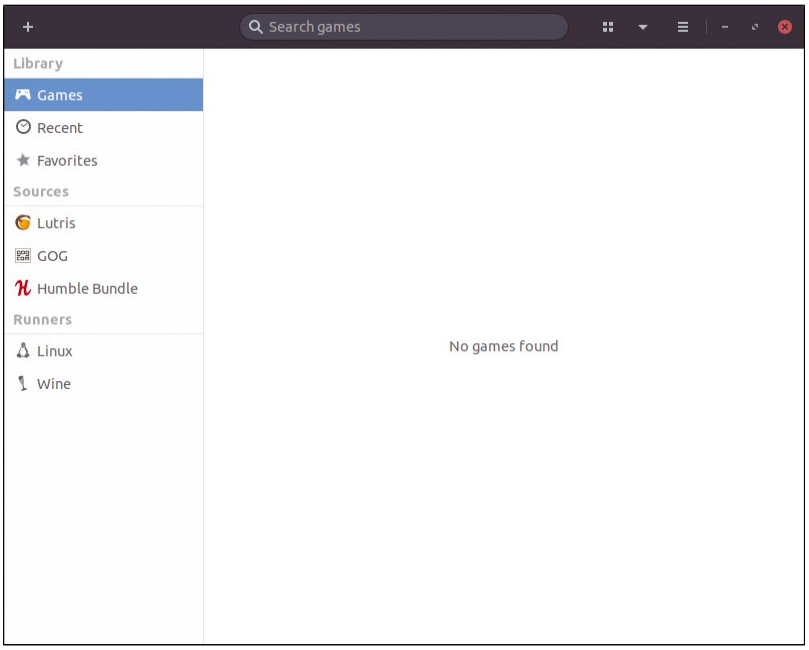


Figure 1: The start window in the Lutris client offers virtually no controls.

Getting Started

In order to install Lutris and use it with Windows games, you may first need to add Wine as a runtime environment on your system. Depending on the distribution, you can either use the Wine packages included in the respective package sources or download them from special repositories (check out the detailed documentation on the Wine wiki page [3]).

Then proceed to install Lutris. Numerous distributions such as Solus, openSUSE, Fedora, Arch Linux, Mageia, or Slackware are supported from the distributions' own archives, or you can use a separate repository – there is one available for Ubuntu and elementary OS and their derivatives. The openSUSE Build Service is a good choice on systems with Debian.

The website also provides a generic tarball and source code for download. The generic package can be run after unpacking the archive on all distributions for which there are no native packages or repositories. The Lutris developers describe all installation methods in detail on the project's homepage [4].

First Launch

When the program launches for the first time, you can expect delays. After all, Lutris needs to complete its configuration in the background. Afterwards, an initially largely empty program window appears. You will find a vertical control bar on the left, plus a titlebar with some configuration elements. It lists Linux and Wine in addition to the supported game stores such as Humble Bundle, GOG, and Steam in the *Runners* group (Figure 1).

Runners are not games but the platforms on which the games subsequently run. Lutris does

not limit you to Linux, Wine, and DOS: It also supports numerous emulators and runtime environments for game consoles. The pool of usable games can be expanded significantly.

To make more detailed program settings, click on the hamburger menu top right in the titlebar and open the *Settings* option in the context menu. The most important options are grouped in the *System Settings* tab (Figure 2). They mainly include the path specification for the default folder where Lutris installs new games, as well as options for video and audio control.

You can change many settings with the help of slide controls. If you check the *Show advanced options* box (bottom left), you can also include command parameters or scripts, which Lutris then executes when calling or closing a session.

The developers list the games you can use on each runner on the project's website [5]. To install additional runners, click the hamburger menu top right in the program window's titlebar and select *Manage runners* in the context menu that opens.

Managing Runners

In a new window, you will now find all available runners. Clicking on the install button to the right of each runner lets you install the respective entry on your system. After downloading and integrating the runner, a button for configuration appears to the right of the runner icon, replacing the install button.

To add games to the system after this, you can enter their names in the central search field in the titlebar. The routine contacts the Lutris server and searches its database. Any games found by this action are then displayed shortly afterwards on the right side of the main window. If you only enter one term or name component in the search field, the program will list all games containing this term (Figure 3).

Click on the desired game and then on the *Install* button that is now displayed. Depending on the platform on which the game is offered, multiple-stage installation dialogs will now follow in a new window. For some games, the routine pops up a window where you can choose between several versions of the game.

If you choose a game offered via Steam, you also need to install the Steam client for Linux and possibly the Steam client for Windows to access the game. The Windows client runs in Wine. When installing Windows games with Wine, you often need to download the Gecko and Mono packages.

After completing the installation, the software asks if you want to create a starter for the freshly set up game on the desktop and in the menu hierarchy of the working environment. These two options did not work with the current Lutris variant in our lab; checking the box did not create any shortcuts.

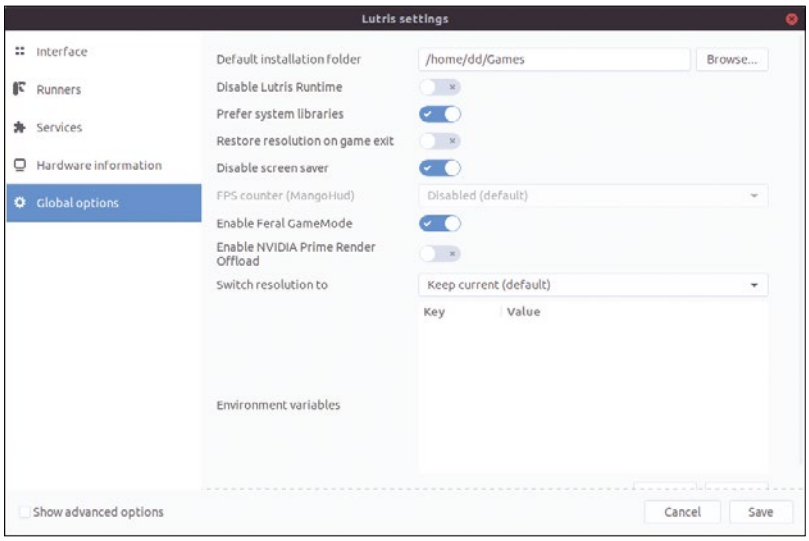


Figure 2: Lutris has detailed configuration dialogs.

Get Gaming

Once you click on the *Games* option in the *Library* group top left in the program window after the install, the installed games appear on the right of the window. Select a game and press the *Play* button bottom center to launch the game in its own window.

Pressing the small triangle button to the right of *Games* opens a context menu that helps you configure further options. You can uninstall the selected game or create a shortcut on the desktop and in the Games menu of the desktop environment, if necessary. To do this, select the *Create application menu shortcut* or *Create desktop shortcut* option. In this way, the games can be started without having to open Lutris first.

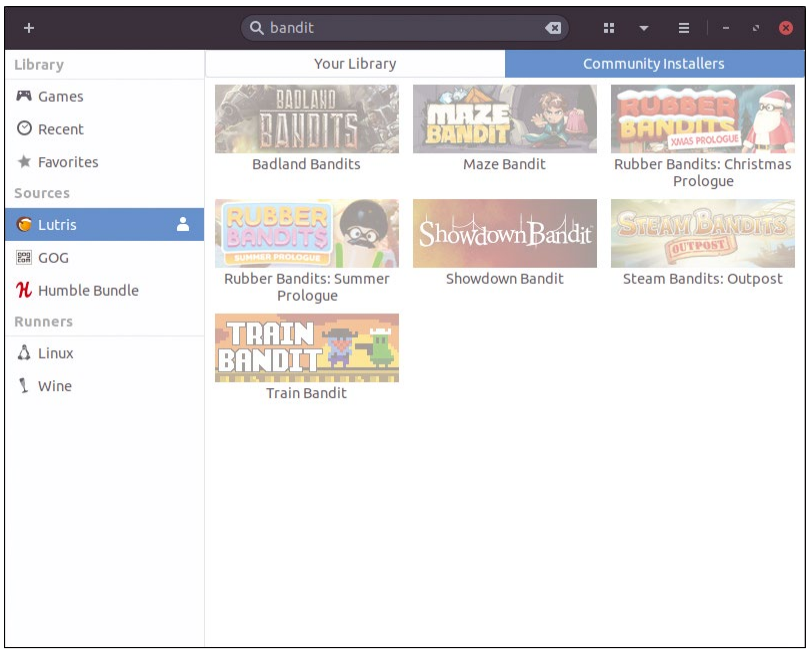


Figure 3: The Lutris server lists all available games based on keyword searches.

To remove existing starters again, use the *Delete desktop shortcut* and *Delete application menu shortcut* options in the same context menu. The *Install other version* option also lets you integrate one of the other supported variants into the system for games of which two or more versions are available.

The *Look at Lutris.net* entry opens a web browser and calls the web page belonging to the selected game. The page provides further information about the respective game (see the "Online and Offline" box). If you want to change technical settings for the respective game, you can call up a detailed settings dialog via the *Configuration* entry. To remove a game's icon from the library after deleting it, click the hamburger menu top right in the client window and check the box to the left of the *Only installed games* option.

Uninstalled games that previously appeared grayed out in the library now disappear, making the library view clearer. If, contrary to expectations, an installed game fails to launch correctly, then in many cases you will find more detailed information via the *Show logs* entry in the game's context menu (Figure 4). In most cases it will be missing libraries, preventing the game from running. You can then install the libraries in question conveniently via your distribution's package manager.

Game Center

Optionally, Lutris also lets you integrate games you installed manually – this makes the program the go-to location for all games. To do this, first make sure you know the path and command parameters for the game in question. Then press the plus button top left in the Lutris client's titlebar.

Figure 4: The logfiles provide detailed troubleshooting information.

```
Running gamemoderun /home/erik/.local/share/lutris/runners/dosbox/bi-
n/dosbox /home/erik/Downloads/bandit-kings-of-ancient-china/KOEI.COM-
 -scaler normal3x
Initial process has started with pid 13382
Start monitoring process.
/home/erik/.local/share/lutris/runners/dosbox/bin/dosbox: error whil-
e loading shared libraries: libSDL2_net-2.0.so.0: cannot open shared-
 object file: No such file or directory
Monitored process exited.
Initial process has exited (return code: 32512)
All children have exited.
Exit with returncode 32512
```

Online and Offline

As soon as you create an account on the Lutris server, you will see details of the more than 13,000 games integrated in Lutris on the project's website without detouring via the client. They appear in a tabular view after clicking on the *Games* tab, and the list can be narrowed down using a number of different selection criteria.

This opens a window where you can configure the game. In the *Game info* tab below *Runners*, define the runner to be loaded, if the game was not developed natively for Linux. The selection field shows the runners activated in Lutris.

The *Game Settings* tab in the dialog then displays various configuration options as a function of the selected runner, and the *Runner Options* and *System Settings* tabs let you modify the parameters for the system and the selected runner. You can adjust the runtime behavior of the game in this way.

Once you have made all adjustments, close the dialog by pressing *Save* bottom right in the window. Any games integrated in this way can be now used conveniently in Lutris.

Conclusions

Lutris turns out to be a powerful management platform for games that are not tied to an operating system. By integrating runners – that is, emulators and runtime environments – the Lutris client enables access to more than 13,000 games. The software is easy to use and configure in the process.

Some functions, such as creating launchers on the desktop and integration into desktop menus, are still buggy, but this is unlikely to diminish the pleasure of using the Lutris client. For gamers who want to use old games or games they used to play on consoles, or to play Windows games on Linux, Lutris is a serious alternative to PlayOnLinux. ■■■

Info

[1] PlayOnLinux: *https://www.playonlinux.com/en/*

[2] Lutris: *https://lutris.net/*

[3] Install Wine: *https://wiki.winehq.org/Download*

[4] Install Lutris: *https://lutris.net/downloads/*

[5] Game List: *https://lutris.net/games*

Synchronize and back up data with FreeFileSync

Always in Sync

Many users still fail to back up important data. FreeFileSync automatically synchronizes specified folders to multiple storage media on demand.

BY ERIK BÄRWALDT

Backing up data is still one of the most unpopular tasks for many users, even though it is no longer necessary to laboriously install and configure special backup programs to do the job. In addition, you can save yourself the time-consuming task of backing up entire disks or partitions if only some data has been modified.

Linux offers the right solution for all conceivable backup and synchronization scenarios. If you only want to keep small or medium-sized datasets synchronized on two data carriers, FreeFileSync [1] is a mature free tool.

Installation

FreeFileSync can be found in the software repositories of various popular distributions and can be conveniently installed on, for example, Arch Linux, Fedora, openSUSE, Mageia, and PCLinuxOS using the integrated package managers. For other distributions, you will find a generic tarball on the project's website. After downloading and unpacking the roughly 25MB tarball, you will find yourself with a roughly 27MB binary package that installs the software on your system.

To do this, Setup opens a small dialog with just three options. Besides the installation option for multiple users, it prompts you for the installation path and creates a starter on the desktop (Figure 1). You can change each specification, if necessary, by entering the corresponding line number, or simply accept the default values by pressing the Enter key.

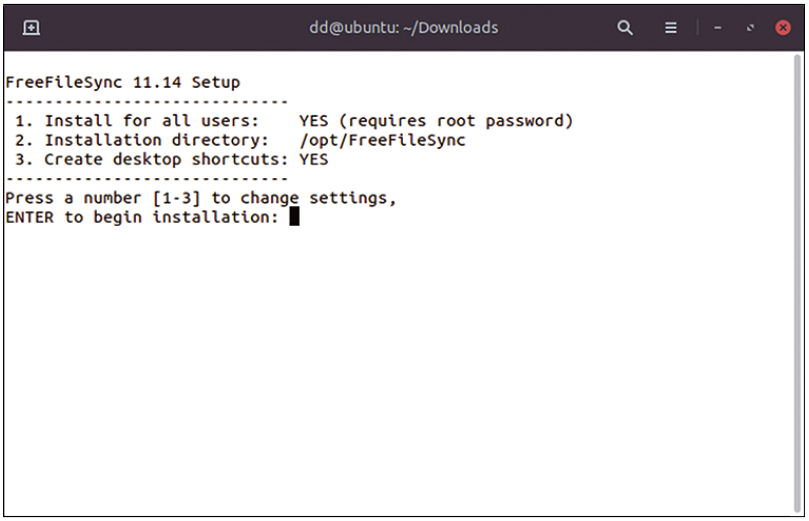


Figure 1: The installation routine is simple but looks outdated.

The routine creates two starters on the desktop as well as in the menus. While the *FreeFileSync* entry starts the application for manual synchronization runs, the *RealTimeSync* option opens a separate window where you can configure automated synchronization runs.

Donation-Supported Version

In addition to the standard version, the developers also offer a donation-supported version. It contains some additional features that the free version lacks, as well as a limited update right. After paying an amount of your choice via PayPal, credit card, or giropay, a download link to this version is mailed to you. The amount donated determines the time period in which you will receive free updates: For every EUR10 (~$12), the developers grant you six months of free updates. This gives you two years for a donation of EUR40 (~$46), for example.

You will see functions that are only included in the donation-supported version of FreeFileSync, such as parallel copying and email notification, in the standard version's configuration dialogs as links that take you to the project website for more detailed info.

Manual Sync

For occasional synchronization tasks, such as those that occur in combination with removable media, simply use the manual variant of the tool, which you call via the *FreeFileSync* starter. The routine first opens a program window that takes a bit of getting used to, with two empty list areas predominating. After the initial configuration, the contents of the directories you want to synchronize appear here. To the left of this you will find two further segments arranged vertically. They

contain the configuration profiles, which are also empty on first launch.

Configuration

Because FreeFileSync is a powerful tool, it has a fairly large set of options to configure. You can set up the general parameters for the tool in the *Extras* | *Options* dialog, which you open via the menubar at the top of the screen. You will find all the configuration options that need to be made independently of synchronization tasks here (Figure 2).

In this dialog, you define, for example, how long logfiles remain in storage, where the software stores them, and whether the application should output audible signals when actions are performed. You can also customize existing context menus in this dialog.

In contrast to these settings, which are generally loaded when the software is called, you will find three further configuration dialogs in the Actions menu. The *Compare Settings*, *Filter Settings*, and *Synchronize Settings* entries let you create profiles that the program loads as a function of the file, directory, and action.

In the first tab, *Comparison*, use the Comparison Settings dialog to define the criteria that FreeFileSync applies to compare files. You have the choice here between *date and size*, *file content*, and *file size*. In addition, you can specify whether the program should follow symbolic links to locate identical files. The *Filter* tab (the funnel icon located top center in the dialog) opens an area where you enter the directories to be included and excluded in a list. You can also filter the data here based on file sizes.

The right tab *Synchronization* offers different ways to synchronize data. This is also where you can define individual criteria for synchronization. Basically, the options depend on whether an object already exists in one of the active directories and how the software should proceed if different versions of a file exist in the source and target directories.

If FreeFileSync detects unresolvable conflicts in object pairs during synchronization, the dialog also offers a configuration option for this. In addition, you can specify in this tab what happens to deleted and overwritten data. You can choose between deleting the data permanently or moving the data to the recycle bin so that you can restore if needed.

Dry Run

After finishing the configuration, start comparing the directory contents. To do this, press the *Browse* button below the *Compare* and *Synchronize* buttons in the program window, respectively, and select the desired directories in the file manager. The application automatically adopts menu hierarchies recursively.

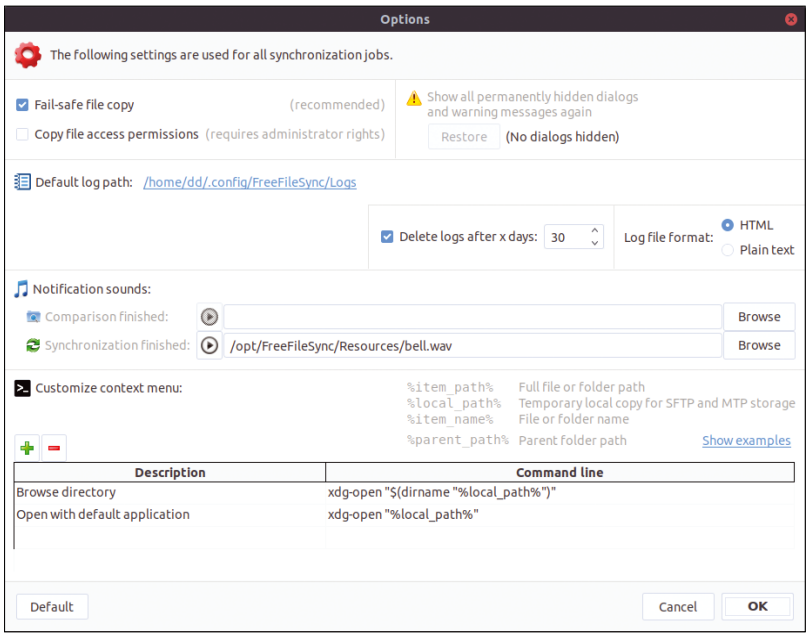


Figure 2: FreeFileSync has extensive configuration options due to its rich feature set.

If you want to compare multiple folder pairs with the same compare and sync settings in a single operation, press the small plus button on the left below the large *Compare* button. The software now adds another input line for a folder pair.

To additionally include cloud storage or a central server on the intranet in the data synchronization, press the button with the cloud icon to the right of *Browse*. The program now opens an overlapping configuration window that lets you either include your Google Drive online storage in the synchronization or talk to a server via FTP/SFTP. In both cases, the respective target can be configured in a few steps in this dialog so that FreeFileSync accesses the directories available there.

After selecting the directories, press the large *Compare* button. The software now compares the folders, displaying the files it finds on the left and right in the two large window segments at the bottom. The green arrows with the plus symbols pointing in different directions between the two folder displays show you which files the software is backing up if you compare from left to right and vice versa. The arrow colors show the transfer direction: Arrows in green mean the file is being transferred from left to right, while a blue arrow refers to the opposite transfer direction (Figure 3).

Synchronizing

Depending on the profile you created, you can then synchronize the two directories by selecting *Synchronize*. If necessary, you can modify the existing profile settings before the sync by pressing

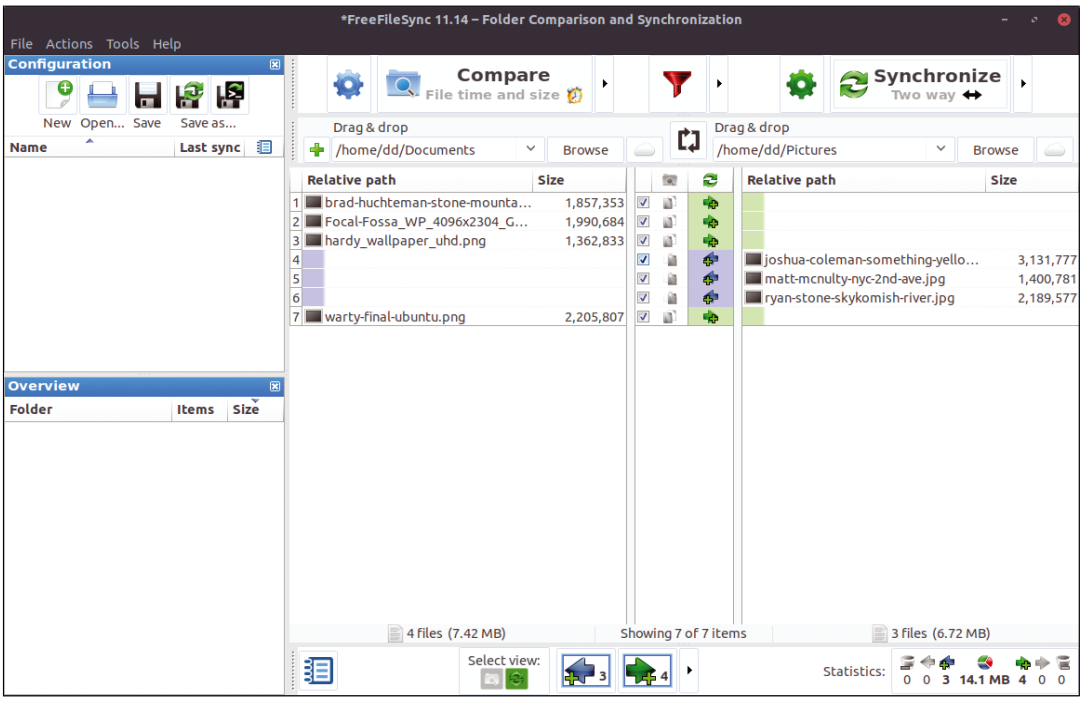


Figure 3: The number of files to be transferred is represented by arrows: green from left to right, and blue in the opposite direction.

the small triangle button to the right of the *Synchronize* button and selecting the desired configuration dialog from the context menu.

In addition, the software lets you customize the sync for each listed file, if needed. There are three alternatives to choose from: Either copy the file (which is usually suggested) in the respective direction specified, delete the file, or ignore it.

In addition, some details of the intended sync are displayed in a small statistics area bottom right in the program window. The stats provide information on the number of files the software copies or updates in each direction and quantify the number of planned deletions as well as the transfer volume. If syncing involves massive copying processes, you can see in advance how much disk space you will need. You can adjust the configuration later by clicking on the funnel icon top center. This opens the filter dialog, where you can include or exclude directories or certain file formats from the sync.

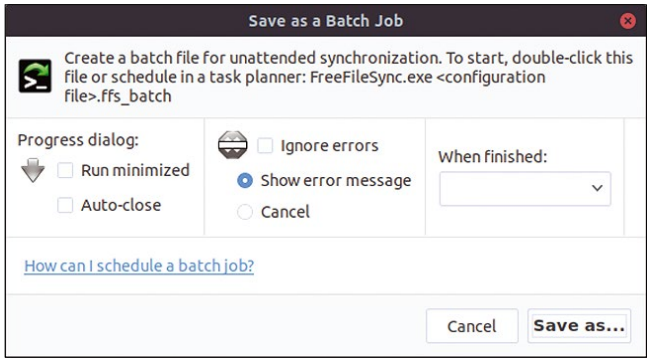


Figure 4: Batch jobs generated by FreeFileSync help you automate data synchronization.

After completing the settings, start the process by pressing the big *Synchronize* button. FreeFileSync now synchronizes the data based on the defaults and displays the action's progress in an overlapping window. It tells you, almost in real time, the number of files that are still to be synchronized.

Note that in the case of slower removable media that communicate with the computer via USB 2.0, writes may still be taking place even if the software is already indicating a full sync. This is why you need to wait before unmounting the removable disk to make sure that the synchronization is complete, and – if the system is still working with the disk – wait for it to be released.

Profiled

Since a detailed configuration of the software can take some time, especially for extensive synchronization tasks, the software offers you the possibility to save your changes in a profile. Doing so permanently saves not only the respective options for data comparison and synchronization but also the source and target paths.

After completing the settings, click on the disk button in the upper left corner of the main window and enter a meaningful name for the profile in the corresponding dialog. You also need to select the storage path for the profile here. After doing so, FreeFileSync saves the profile at your

chosen location and displays it in the list view top left in the program window. This view can store any number of profiles, giving you the ability to choose the appropriate profile without further complex configuration – which is especially useful if you use multiple removable media as the source or destination drives.

Batch Operation

Batch files, which store the jobs to be executed, are also used to automate database syncs. They are based on the pre-configured profiles. The dialog for this can be opened by clicking on the right disk symbol top left in the program window.

In the window that now opens (Figure 4), enter the details of how FreeFileSync should execute the batch job and what should happen to the system on completion. You also specify here how the system should react in the event of an error. After saving the batch file, it can also be called by a cronjob if so desired.

Fully Automated

Especially in larger infrastructures with distributed storage capacities, permanent data synchronization makes perfect sense. With an automated routine, FreeFileSync can synchronize arbitrary directories and datasets. The developers have spun off this function into an independent program window named RealTimeSync, which also has its own launcher. This opens a small dialog in which you can activate any number of folders for permanent synchronization either by drag-and-drop or by typing in the paths (Figure 5).

In a command-line bar, you first define the file to start the program including its path and point to a batch file you created up front that contains the functions for the sync. Pressing the *Start* button bottom center in the window creates a corresponding icon in the system tray. Clicking on it opens the dialog again. The routine now monitors

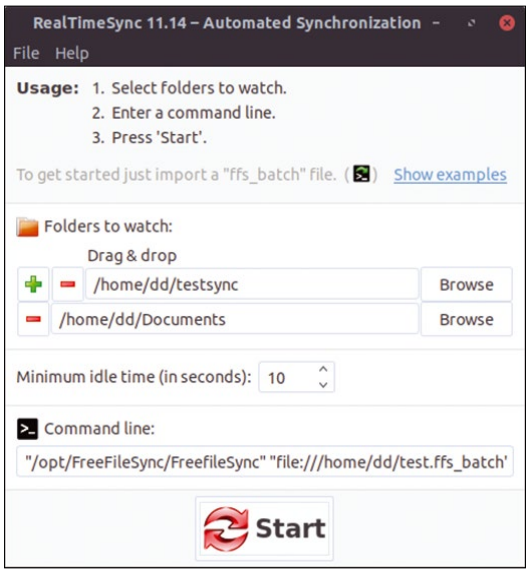


Figure 5: If so desired, RealTimeSync synchronizes datasets permanently in the background without your intervention.

the specified directories and automatically synchronizes when datasets change. In this way, you can keep the data up-to-date without having to do anything yourself.

Conclusions

FreeFileSync gives you a stable, mature program for synchronizing a wide variety of databases. The software has an easy-to-use interface with a logical structure that requires very little training. At the same time, the software also synchronizes extensive data inventories that extend over numerous directory levels. Thanks to the option of saving various configurations in profiles, and even executing them unattended as batch jobs if required, the software is also useful for backing up data in the enterprise. ■■■

Info

[1] FreeFileSync: *https://freefilesync.org*


FREE DVD!
openSUSE Leap 15.3
JOIN THE LINUX REVOLUTION!
ALL THE SOFTWARE YOU NEED!
GETTING STARTED WITH
LINUX
• MORE POWERFUL • MORE SECURE • MO
LEARN HOW TO SET UP A LINUX SYSTE
• Listen to Music • Play Games • Process P
• Surf the Web • and Much More!
STAR
LINUX NEW MEDIA
WWW.LINUX-M
301 BEST BASH COMMANDS
LINUX SHELL
HANDBOOK 2021 Edition
LINUX Special
SUPERCHARGE
YOUR LINUX SKILLS
Power at Your Fingertips
• Pipe and redirect output
• Monitor processes
• Create custom scripts
FREE DVD!
LibreOffice Full Version
Become a LibreOffice Expert!
2020 Edition
LibreOffice
Dive deep into the world's greatest free office suite
Write Your Own LO Macros
Save time and automate common tasks
Digital Signatures
Lock down your private documents
Edit and Save
LINUX NEW MEDIA
TUNE YOUR LINUX SYSTEM
2021 EDITION
101 COOL LINUX HACKS
Tricks and shortcuts for Linux geeks
■ Recover deleted docs
■ Send files without a target IP
■ View a handy cheat sheet for your favorite commands
STREAMLINE:
Clean up hidden files
EXTREME CHROOT:
Change to a second distro
Discover the secrets of the experts
LINUX NEW MEDIA
WWW.LINUX-MAGAZINE.COM
ARCHIVE DVD
RASPBERRY PI GEEK
THE COMPLETE ARCHIVE
2,000 pages of maker projects and more!
FREE DVD $39.90 VALUE!
MakerSpace
HANDS-ON PROJECTS FOR MAKERS
Dive Into
• Raspberry Pi
• Arduino
• Retro gaming
Discover FPGA
Learn to program hardware for retro computing
LINUX NEW MEDIA

FOSSPicks

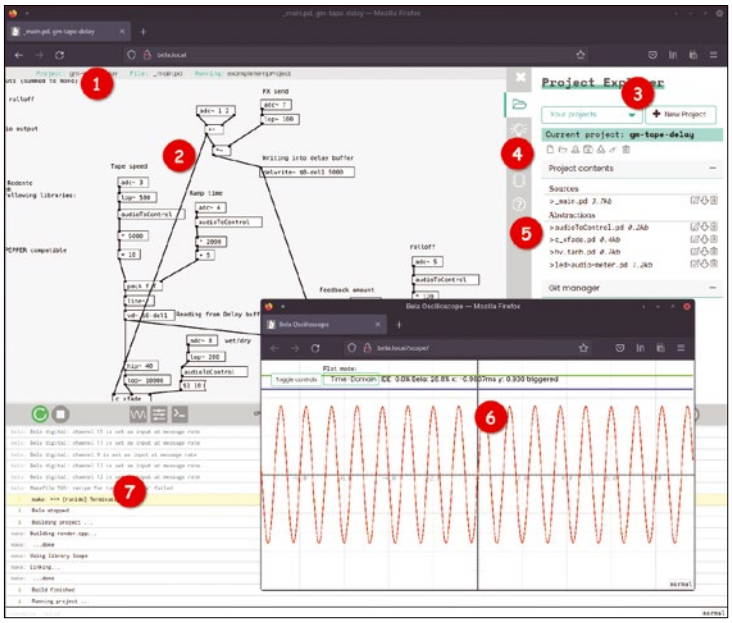
Sparkling gems and new releases from the world of Free and Open Source Software

After backing too many now-delayed, tech-related crowdfunding campaigns in 2020/21, Graham is going to be swamped by new gadgets in 2022. BY GRAHAM MORRISON

Realtime audio platform

Bela

Bela is a programmable audio platform with incredible performance, an integrated development environment, a large active community, and virtually no limits. But it's also a slightly unusual project because you need some extra hardware to make it properly work. That hardware is a BeagleBone Black with an optional shield that adds multiple digital, analog, and audio inputs and outputs. This setup, when combined with the Linux-based software stack, promises a 0.5ms (half) delay between when a sound is triggered by the software to when the audio exits the hardware. For comparison, a Mac can manage a few milliseconds with the very best expensive hardware, desktop Linux with a realtime kernel adds a few more milliseconds to this, an Arduino is in the 10ms range, and a Raspberry Pi the 20ms range. All these delay times are much quicker than what you'd get from a default PulseAudio configuration without specialist hardware, which may typically operate with a delay of 100ms.



1. **Project management:** Load and switch between a variety of onboard projects. 2. **IDE:** Code your projects directly, or view the projects you upload. 3. **Project explorer:** Upload and download projects to the BeagleBone. 4. **Tutorials and examples:** Browse lots of examples plus the excellent documentation on each function and capability. 5. **Pin explorer:** An annotated diagram of the device pins can be shown for easy programming. 6. **Oscilloscope:** This feature is integrated into the IDE but connects to the hardware. 7. **Log and debug:** The output from your projects is listed in a separate pane.

An ultra-low audio delay makes a big difference because it enables Bela to do all kinds of things transparently, before delays and their associated latency would start to break the immersion. A good example might be if you were to use Bela to create a software-generated drum kit with a DIY hardware trigger. The ultra-low response time would allow you to hit the trigger and hear the sound almost instantaneously, just as you would with real drums. It's this lack of delay that most differentiates digital instruments from their analog counterparts, and the low delay values seen on the Bela platform with the BeagleBone Black and the audio shield really are game changing.

None of this would be worth mentioning if there weren't an accessible and easy-to-use software stack that's both ready to take advantage of this power and easy to use when you want to build your own hardware or software experiments. High performance comes from Xenomai realtime Linux extensions, and the flexibility comes from an integrated web-hosted development environment that becomes accessible as soon as you connect the device to your Linux computer. You can use this Integrated Development Environment (IDE) to code in SuperCollider, Csound, or C++, and even to upload Pure Data patches with graphical previews of C++. All of this software can of course be run on your Linux desktop, where patches can be developed and tested before being transferred to the onboard IDE. But developing with the IDE is also perfectly feasible, and you'll get an immediate response from the hardware when you do press play. It may sound complicated, but there's enough here for even beginners to get started.

A fresh install includes many working examples, plus a community library from which you can simply copy and paste. Much like the Norns platform we've looked at previously, you can get a lot out of the Bela platform without ever writing a line of your own code. Copying and pasting patches, modifying variables to better suit your use cases, and building your own solutions are all possible without any programming know-how. Examples include complex digital signal processing, such as reverb, delay, and convolution effects, plus lots of synthesizer and sequencer patches, alongside many more esoteric examples that often sit alongside a hardware input project. It's creative and educational, practical and ephemeral, and one of the best (and cheapest) ways to get into high-performance experimental audio design.

Project Website

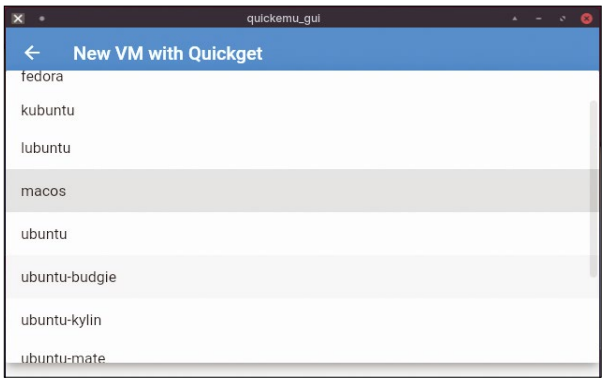
https://bela.io

Virtual machine launcher

Quickemu

Even the most cynical anti-cloud Linux users will admit that virtual machines (VMs) have changed how we use computers. In the cloud, VMs can provide unlimited scale and flexibility, at a cost, while locally they let us isolate environments, run more than one distribution, and build for different architectures. But they can still be tricky and less than intuitive. VirtualBox has a functional GUI, but it requires you to download an appropriate image, define its configuration, and run through the installation. The libvirt library has its own GUI, Virtual Machine Manager, with a similar set of configuration panels to navigate. Docker, Multipass, and LXC perhaps offer the best options, because they're backed by their own images that install and prepare themselves transparently, but they still require you to navigate the their tools and know how to access the operating systems you've deployed.

Quickemu, another possible solution, is really nothing more than a clever script (for now!), but it's a script that makes a lot of sense when all the underlying technologies are already present in our Linux systems, with most of us simply lacking the time and skill to marshal their capabilities into a predictable environment. In the background, it's downloading images and configuring Qemu, SPICE, VirGL, USB and smartcard passthrough, SSH port-forwarding; full-duplex audio, and TPM emulation without requiring any user input. Quickemu promises to "Quickly create and run highly optimized desktop virtual machines for Linux, macOS and Windows; with just two commands." And it delivers, even with the last operating system on that list, Windows 11, which is perhaps just as well considering Linux might be the safest place to install it. The two commands are `quickget` and `quickemu`. The first will download the operating system of your choice, such as *ubuntu focal*, *macos catalina*, or *windows 11*, with plenty of options for variations and flavors, and all the images are legally sourced. But most importantly, it just works – two commands and you have your chosen virtual environment.



There are already a couple of GUIs for Quickemu, including this one by Mark Johnson, friend of Quickemu's creator, Martin Wimpress.

Project Website
https://github.com/wimpysworld/quickemu

A tmux fork for graphics

sixel-tmux

There can be little doubt that one of the best command-line utilities of all time is tmux. Like `screen` before it, tmux transforms your humble command line into a fully fledged productivity portal, allowing you to split the terminal, create new background sessions, install plugins, and easily detach and reattach to concurrently running sessions. If you spend a lot of time on the command line, tmux quickly becomes the equivalent of your desktop environment, and you typically find yourself wanting to context switch to your graphical environment less and less. This in turn leads you to wanting to perform more and more of those desktop tasks on the command line. There are lots of great tools that can be used as substitutes to those desktop tasks, but there's one element of the desktop that has been difficult to re-engineer on the command line, and that's graphics.

The idea for the terminal predates modern graphical capabilities, which is why terminals so often resort to ASCII. Ancient terminals could only display these ASCII characters because single pixels couldn't be directly referenced. Of course, modern screens aren't like this, but our terminal emulators stay true to those early limitations to ensure the broadest possible compatibility. In theory, they should be able to do so much more. Which is why this fork of tmux is so interesting. It's called sixel-tmux because the sixel part is referring to a bitmap graphics format supported by a more advanced terminal. It's still not desktop-good, but it does add 64 possible patterns to a six-pixel-high and one-pixel-wide special "character" that can do a far better approximation of a cat photo than `^._.^`. This might not make a huge difference if images aren't important to you, but it will help anyone else who needs to see pictures while they're on the command line, whether that's generating charts or processing photos. Sixel-tmux helps you see the output instantly and without any context shifting.

Before sixel-tmux will work, you need to carefully configure the colors in your terminal.

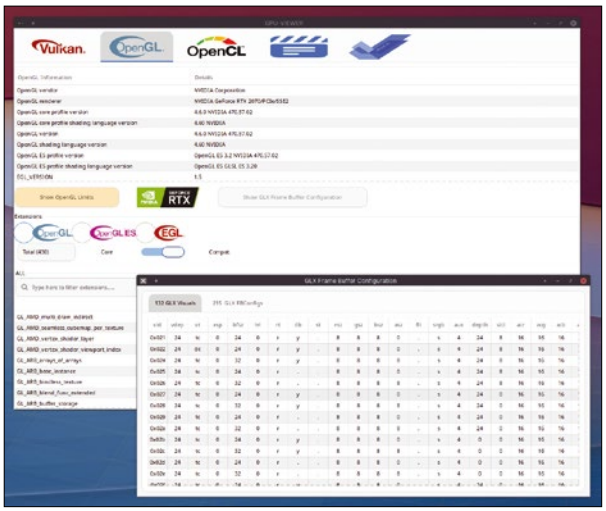
Project Website
https://github.com/csdvrx/sixel-tmux

GPU analyzer

GPU-Viewer

Many tools can help when analyzing your system, but many of these have a focus on your CPU, mainboard, or storage. Not many look at your graphics hardware specifically, possibly because, until recently, many Linux users didn't need a powerful graphics accelerator outside of machine learning and didn't need to see whether their current one was performing adequately. Graphics capabilities were simply judged on the size and resolutions of the displays they could drive. But the popularity of Linux gaming is changing that; PC gamers are fastidious about their systems' performance and capabilities. If you want to know everything about your system's graphics hardware, there isn't a single way to survey the multifaceted APIs most hardware now support – until GPU-Viewer.

For anyone with a decent GPU, GPU-Viewer shows your GPU's capabilities and how much of them have been recognized and are accessible by your system. GPU-Viewer is a clever combination of preexisting tools, including `glxinfo`, Vulkaninfo, and `clinfo`, with healthy doses of `grep`, `cat`, and `awk` to parse the data, and a GTK+3 GUI. Looking through the GUI, you can't tell this data has sometimes humble and disparate roots, because it's so overwhelmingly comprehensive. Five large buttons at the top let you switch between the five main views, showing your hardware's compatibility with Vulkan, OpenGL, OpenCL, and encoder and decoder support, followed by an



If you've been unable to upgrade your GPU because of the chip crisis, spend your time studying your current hardware instead.

About page. Every page is full of information, from the device and driver versions, to the available memory and the exact capability of every system call. The OpenCL page will tell you if a compiler is available, and the OpenGL page will even let you switch between OpenGL, OpenGL ES, and EGL for the broadest possible hardware compatibility.

Project Website

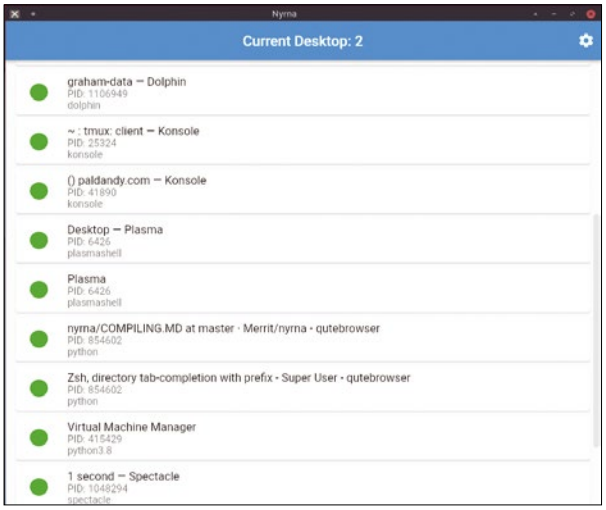
https://github.com/arunsivaramanneo/GPU-Viewer

Suspend resume

Nyrna

Valve's Steam Deck is bringing a lot of needed attention to the Linux desktop and its associated applications, even when there's no direct gaming connection. Gaming sessions are run within something that Valve calls Gamescope, which sounds new but is actually a Wayland compositor that better allows for gaming integration within Linux. It's using Wayland with XWayland, but it's still a positive step towards more general Wayland acceptance and surely its eventual total replacement of Xorg. It's decisions like these, and Valve's decision to use Arch Linux as its SteamOS base, that are bringing Linux to wider attention. But we've seen other projects that are also trying to bring games' console-like features to Linux.

Nyrna is a brilliant little utility that must surely have been inspired by Valve's Steam Deck, because it implements a feature that Valve was still working on at the time of the announcement. This is the ability to suspend and resume games, much like you can on a modern games console such as the Nintendo Switch. It's most useful for games because they don't always have a convenient save position from which to suspend and resume your gameplay, and you want something instant if your attention is diverted. Nyrna does this but also does the same for whatever windows and applications you have open. It's built using Google's Flutter UI and features a simple user interface that makes the most of Flutter's flawless transitions. By default, it will list the applications and games it finds on your desktop, and you can simply click a green button alongside an entry to freeze it, freeing up its resources



You can choose to filter out all the windows that are hidden, including those on other desktops.

until you press again to resume the process. The action is almost instant in both directions and beautifully animated with your chosen target floating into and back from your panel like a genie into and out of a bottle. But most importantly, it works flawlessly and quickly becomes one of those tools you wonder how you ever lived without.

Project Website

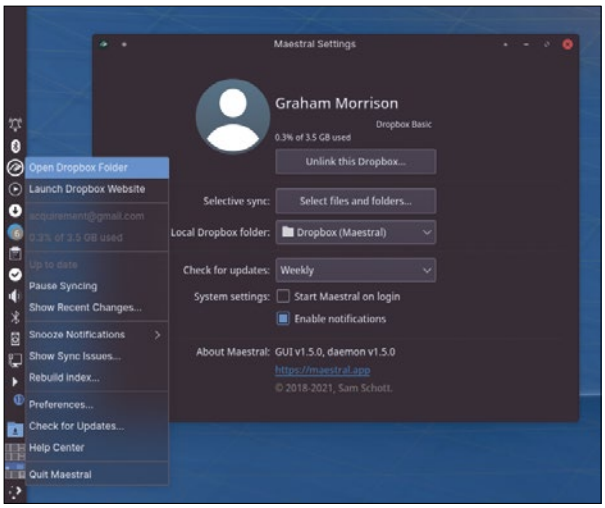
https://github.com/Merrit/nyrna

Dropbox client

Maestral

Dropbox was revolutionary when it first appeared in 2007. The cloud storage it offered was something new, and the convenience of having a directory shared with cloud storage that would then synchronize itself with other machines was alluring. Combine this with the gamification of inviting new users to add to your personal storage capacity, and many of us Linux users were hooked. Even though the whole platform was closed and proprietary, the relatively open API and a choice of third-party Linux clients meant we could all pretend Dropbox would eventually support Linux as a Tier 1 system. But the lack of Linux progress and an alleged hack in 2012 (widely reported in 2016) that exposed 68 million user passwords dampened our enthusiasm, especially in what was now a competitive market.

But Maestral could just do enough to reboot our enthusiasm for Dropbox. It's a completely unofficial client but links to your account via an easily generated API key that allows access to the public API for your account. The most impressive thing about the project is that there's both a command-line tool and a GUI, with the command-line tool able to perform all the same actions. There are options to pause and resume syncing, start or stop the background daemon, add files and folders to be ignored or included, and desktop notifications. The GUI is brilliantly designed and is very clear and easy to use. Its panel application icon has a right-click menu that mirrors many of the same functions too, and it's by far the best client we've ever used for Dropbox. But the best thing about it is that it requires very few resources and won't needlessly eat CPU cycles, which can't be said of the other clients for even some of Dropbox's competitors.



Maestral integrates itself into the Linux desktop environment far better than we imagined any official Dropbox could.

Project Website

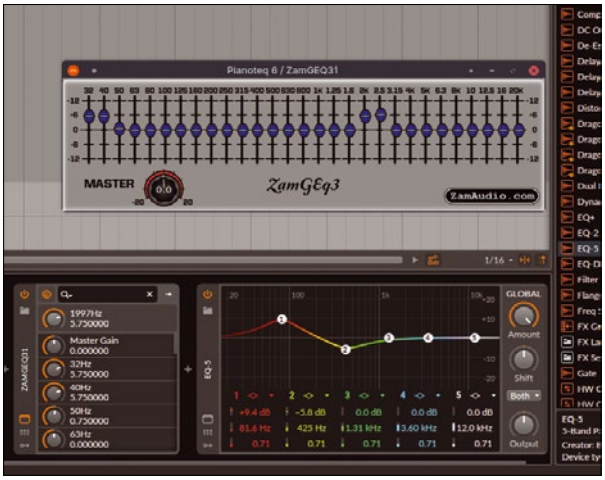
https://github.com/SamSchott/maestral

Headphone improver

AutoEQ

This is a phenomenal project that tackles what most people would consider a minor issue: the audio quality from their headphones. But it does this in such a hugely impressive and well-documented way that even audio cynics would be impressed. At the core of the project is a command-line tool that can listen to and analyze the frequency response of your hardware. The frequency response of a set of headphones (or speakers) can be visualized as a graph that shows how accurately the sound output reflects the sound input. Can it accurately reproduce low bass frequencies, for example, or the mid frequencies typically occupied by the human voice, or the high frequencies generated by high hats in a drum kit, or the raspiness of Miles Davis's trumpet? The truth is that most headphones massage their frequency response to flatter the sound, often inflating the bass, mid, and high frequencies, but sometimes their design can adversely affect the sound you want.

AutoEQ can help fix your headphone audio but not directly. As mentioned earlier, it's able to measure the frequency response of your headphones and create a map of where the output is lower or higher than an ideal flat frequency response. It can then give you the ideal settings to fix your hardware or to boost or reduce in the frequency areas you prefer. But you don't even need to take the measurements yourself. The project has amassed more than 2,500 profiles for headphones, and there's a good chance yours will already have been analyzed. It's then just a simple matter of programming the suggested compensation into your favorite equalizers. An equalizer is an audio processor that can boost or attenuate the audio at specific frequencies, just like the graphic or parametric equalizers on old stereos. Linux has plenty to choose from, and the project documentation does a great job of recommending plugins for a variety of platforms, including Android and Linux, for which (for transparency) a previous FOSSPick, PulseEffects, is recommended.



AutoEQ generates equalizer settings for you to use with your own headphones to help improve their audio quality.

Project Website

https://github.com/jaakkopasanen/AutoEq

Linux for Android

Termux and AnLinux

Termux is one of the best applications you can install and run on any Android device. It's a terminal emulator with an integrated package manager that creates a comprehensive command-line Linux environment on your phone. It can currently only be installed as a third-party package, or via the F-Droid open source app store, but it doesn't require root privileges. This means anyone can install it and instantly carry Linux with them wherever they go. It even augments whatever onscreen keyboard you use with some essential Linux keys, including cursor arrows, escape, control, alt, and tab, so you can perform almost any command-line task without resorting to arcane button-conjuring tricks.

But the best thing about Termux is the integrated Apt-based package manager and its associated packages. Even from a humble containerized Android sandbox, it allows you to install some of your favorite Linux tools, such as an SSH server and client, tmux, and the Vim text editor – and all from within a variety of different shell environments. With a few packages such as these installed, Termux feels exactly like having a Linux distribution in your pocket while also enjoying the advantages of Android. But with help from AnLinux, Termux can also be used to install what is effectively a fully fledged Linux distribution, including even a desktop if your phone is up to the task.

AnLinux is a collection of tools, scripts, and associated distribution-based images that add hugely to Termux's capabilities. It's initially installed as an app, either from Google Play, GitHub, or F-Droid, but all the app really does is provide you with a customized command to copy and paste into Termux. It does this by first asking which distribution you want to install from a list that includes Ubuntu, Debian, Fedora, openSUSE, and Arch, among many others. This results in a command that you copy and paste into Termux. The command will download and execute an open source and easily parsable script that will itself download a customized root filesystem image for your chosen distribution. Running a launch script for each distribution will encapsulate the unpacked root filesystem within a PRoot environment, a chroot-like container tool that requires no Android privileges to create an entirely coherent Linux environment.

You can now interact with your distribution much like you might a native Linux installation, including installing packages, building projects, pulling in your configuration files, and accessing your regular online services and resources. With Ubuntu, for example, you can access all the same packages, build tools, languages, and desktop applications, with just a few restrictions related to the compromises made for Android. In particular, all the images are heavily modified to remove incompatible elements such as systemd. This means installing a desktop environment requires you to revisit the AnLinux app to generate an additional installation script. After this has been run, however, you can then access your desktop through a local VNC client. We were even able to connect remotely via an SSH connection forwarding port 5901 to our remote host. It works brilliantly, and a modern smartphone is more than capable of native desktop speeds, even over VNC, letting you get real work done.

Project Website

https://github.com/EXALAB/AnLinux-App

Termux with AnLinux is like having a native Linux smartphone with all the convenience of a native Android installation.

If you don't want to use your phone's small screen and keyboard, most phones can now connect to a screen, keyboard, and mouse over USB-C for the full desktop experience.

Classic pinball

Space Cadet

Software recreations of real pinball games shouldn't work. Left and right mouse buttons can't simulate the feel of the paddles, nor can the space bar replace the spring and thunk of the ball launcher. Pinball is a physical game that needs to exist within a real space, alongside other games in a real environment. The physicality of the mechanism, the momentum in the chrome ball, the flashing lights, and the crude sound all contribute to the heady playability cocktail that anyone who loves pinball will know. And yet, despite all this, software recreations of pinball games can and do work. This was proved early on in the 8-bit era and with the wonderful Pinball Dreams and Pinball Fantasies on the Commodore Amiga, which all effectively made pinball a whole new software arcade genre that continues to this day.

Decades later, you can now play full 3D recreations of real and invented pinball games in virtual reality, and many ardent players have built their own physical machines with PCs and large LCD panels to replace the mechanisms beneath the glass. But there's now also a huge group of players who crave the instant playability and gratification of those early software recreations, and on the PC, few were more loved than Space Cadet. Space Cadet was a Windows-only game from before the turn of the millennium. It was significant for using a 3D model for the layout and physics and was incredibly accurate and playable for a virtual recreation. In a hugely impressive feat of software development, the original binary has now been reverse engineered and refactored to build on any modern system, Linux included. All you need are the original data files and the game becomes 100 percent playable. It's still a brilliant blast of pinball.

Software pinball requires perfect design and fine tuning for the gameplay to be addictive, and those elements don't age.

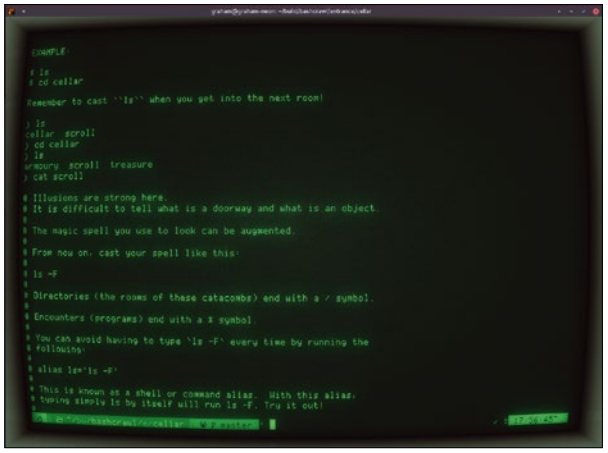
Project Website
https://github.com/k4zmu2a/SpaceCadetPinball

Terminal trainer

bashcrawl

Despite its ubiquity, many of us forget how intimidating the command line can be to people who have never used it before. This is a tricky issue to overcome because this initial trepidation only dissipates with experience. But to gain that experience, no matter how much you've read beforehand, you need to do actual things with the command line. It's a vicious cycle, but it's also one that Bashcrawl attempts to break. Bashcrawl isn't really anything other than a collection of files and directories, but it's a collection that's been put together in such a way that **what you need to learn for the story to progress is entertaining** and could provide that vital first step for people who want to use the command line but don't know how. To start, you simply open a terminal in the `entrance` directory of the download. This directory contains a text file called `scroll` and another directory called `cellar`. The only instruction you have to start with is `cat scroll`, which will of course print the contents of the `scroll` file to the screen.

In this way, you begin your tentative first steps with mastering the POSIX terminal, first learning `cat`, `ls`, and `cd`, before learning how to create aliases and variables to hold values. These elements are then used within scripts to allow you to progress within the current environment, making the whole experience feel very much like a text adventure game built using only the commands that are already part of your terminal environment. The game is short but hopefully does enough to coax the player from within the relative safety of its directories, text files, and scripts and into their wider Linux environment, where hopefully they can start a brand new adventure of their own.



Playing bashcrawl could be considered a little like playing Ender's Game.

Project Website
https://gitlab.com/slackermedia/bashcrawl

Working with calendars in the shell

Time Machines

Command-line aficionados do not have to forgo calendars and appointment reminders. The shell offers many tools for user-friendly handling of date definitions in scripts. BY HARALD ZISLER

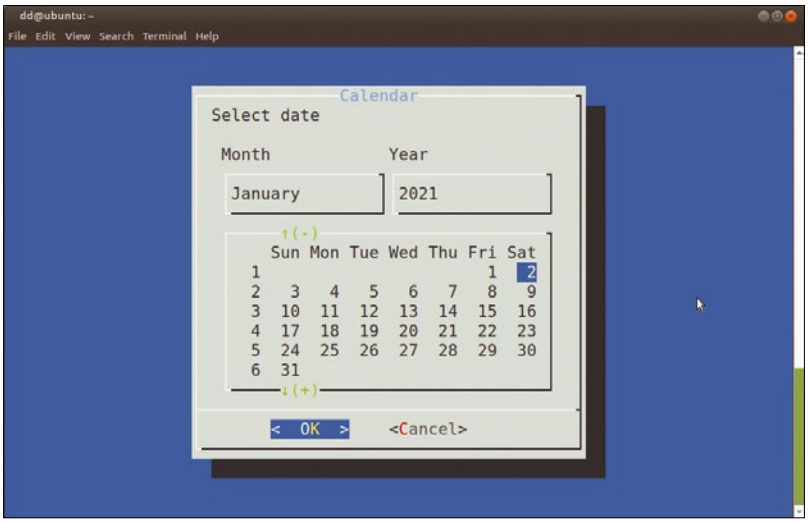
Working with dates in the shell often causes problems. Aside from the usual typos, this is usually due to mismatched language and country settings. In practice, you will especially be confronted with this when working with databases.

Manual input can be relatively easily checked for formal correctness.

```
dd@ubuntu:~$ sh listing-01.sh
Enter date: 16.09.2021
Date converted: 2021-09-16
dd@ubuntu:~$
```

Figure 1: Using a script to convert a date format. Date formats are defined by standards such as DIN 5008 (for Germany) [1] and ISO 8601 (international) [2].

Figure 2: The `dialog` date mask in a terminal window.



Listing 1: Formatting the Date

```
#!/bin/bash
# Comma as field separator
# Output in format YYYY-MM-DD
# Input
read -p "Enter date: " input
# Replace comma with dot for uniformity
input=$(echo $input | tr \, \. )
# Output to variable with Awk, minus sign as separator
output=$(echo $input | awk -F \. '{ print $3 "-" $2 "-" $1 } ')
echo "Date converted: $output"
```

The simplest aspect to change is the separators or the order of day, month, and year. An example of this is shown in Listing 1, and the corresponding output is shown in Figure 1. The short script supports both correct input with the dot as a separator and input with the comma in the numeric keypad for quick data entry.

dialog and YAD

If you attach importance to user-friendliness as well as correct date values, why not use the attractive (semi-)graphical input options offered by `dialog` or YAD?

With `dialog` [3], simple date input takes the basic form of:

```
dialog --calendar 0 0
```

The two zeros automatically set the size to match the terminal. In the calling shell script, `dialog`'s various exit codes can be used for control, as shown in Listing 2. In the resulting mask (Figure 2), you can use the tab and arrow keys to navigate and press the Enter key to complete the entry.

YAD [4] lets you output a shell script's user dialog in the GUI. Because of the very extensive options, you might want to read the YAD's documentation or at least excerpts of it. The input is conveniently mouse controlled. In the example

Listing 2: A dialog Query

```
#!/bin/bash
date=$(dialog --stdout \
  --title "Calendar" \
  --calendar "Select date" 0 0)
if [ $? -eq 0 ]; then
  datum=$(echo $date | tr \/ \.)
  echo "Input date: $date"
elif [ $? -eq 1 ]; then
  echo "Input canceled"
fi
```

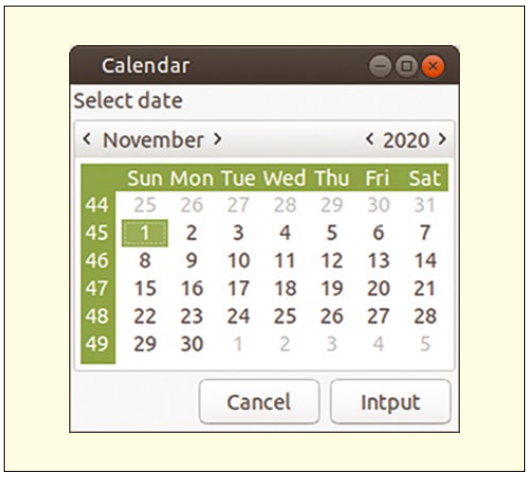


Figure 3: A YAD date entry in the GUI.

shown in Listing 3, I replaced the standard buttons with my own, resulting in different exit codes (i.e., 2 for input). You can see the output of the script in Figure 3.

smenu

Display redirection via SSH does not work everywhere. In addition, displaying `dialog` masks can cause problems with some terminal settings. However, there is another method you can use if it is important to avoid incorrect entries (such as April 31). You specify the possible values with smenu [5].

For the calendar year, the user can select the current, previous, and next year in Listing 4 (line 7). The selection for the day works in the same way (line 17). To save space in the terminal display, smenu's `-n1` option is used to limit the selection to one line. You can use the arrow keys to select the day. A `cal` command (line 16) determines beforehand which specifications are permissible for it. The script extracts additional specifications such as the month, year, and weekday from this. To make the date suitable for further processing, single-digit day specifications are given a leading zero (lines 19 to 20). (Figure 4 shows the script output from Listing 4.)

Listing 3: A YAD Query

```
#!/bin/bash
date=$(yad --title=Calendar \
  --text="Select date" \
  --calendar --show-weeks \
  --button="Cancel":1 \
  --button="Intput":2)
if [ $? -eq 2 ]; then
  date=$(echo $date | tr \/ \.)
  echo "Input date: $date"
elif [ $? -eq 1 ]; then
  echo "Cancel entry"
fi
```

Shell Calendars

To display a calendar, you can use `cal` or the spruced-up, newer `ncal` variant, which the popular distributions already preinstall. The options for both programs can be found in Table 1. Called with the `-C` parameter, `ncal` behaves exactly like `cal`. Figure 5 shows the two resulting calendars, which display three months along with the week numbers.

calendar

With `calendar`, which most distributions preinstall with the *bsdmainutils* package, you can output the current date's events in the terminal. If you set the command without further options,

```
dd@ubuntu:~$ sh listing-04.sh
Select calendar year

2020 2021 2022

------------------------

Select month

01 02 03 04 05 06 07 08 09 10 11 12
------------------------
Use arrow keys to select day

16  ▼
16.08.2021
dd@ubuntu:~$
```

Figure 4: The script output from Listing 4.

Listing 4: Querying a Date with smenu

```
01 #!/bin/bash
02 # Year: current, last, next
03 ayear=$(date +%Y)
04 vyear=$(echo $ayear -1 | bc)
05 fyear=$(echo $ayear + 1 | bc)
06 # Use Smenu to select months
07 year=$(echo $vyear $ayear $fyear | smenu -m "Select calendar year")
08 echo ""
09 echo "------------------------"
10 echo ""
11 monat=$(echo 01 02 03 04 05 06 07 08 09 10 11 12 | smenu -m "Select
          month" )
12 echo "------------------------"
13 # Create header for selection
14 menuhead=$(cal $month $year | head -1)
15 # Calendar field
16 calendar=$(cal $month $year | tail -6 | tr -d [:alpha:])
17 day=$(echo $calendar | smenu -m "Use arrow keys to select day" -t1 -n1 )
18 # Prepend zero to single-figure days
19 if [ $(echo $day | wc -L) -eq 1 ]; then
20   day=$(echo "0$day")
21 fi
22 date=$(echo $day.$month.$year)
23 echo $date
```

Table 1: cal and ncal Options

Option	ncal	cal
Do not highlight today	`-h`	–
Month (current year)	`-m M` [1]	–
Month (any year)	`MM YYYY` [2]	`MM YYYY / -d YYYY-NN`
Annual calendar for year	`YYYY`	`YYYY`
Calendar this year	`-y`	`-y`
Three-month calendar	`-3`	`-3`
Last month	`-B <number>`	–
Next month	`-A <number>`	–
Number weeks	`-w`	`-w`
Display weeks horizontally	`-C`	–
Display weeks vertically	–	`-v`

[1] `M` = Numeric month input
[2] `YYYY` = Four-digit year

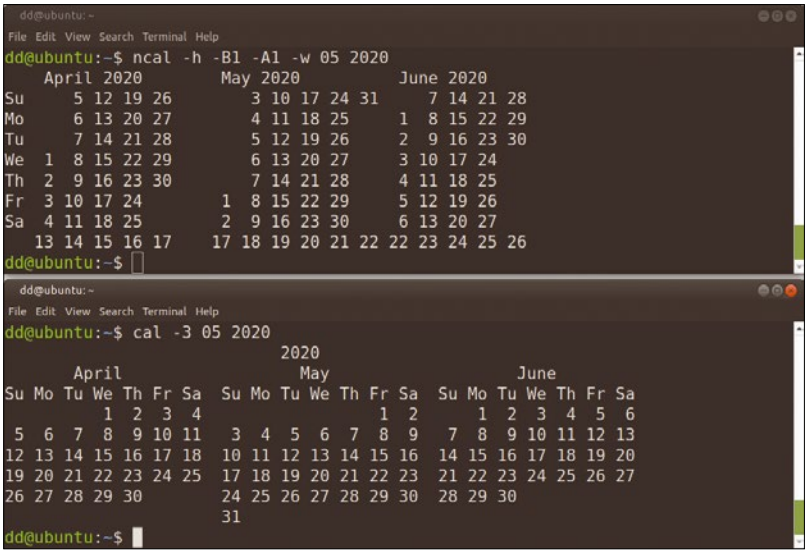


Figure 5: A three-month calendar output with `ncal` (top) and `cal` (bottom).

you will see a long list of mostly irrelevant information. You can restrict the query to the UK, for example, by calling `calendar` with the `-f` parameter and an appropriate event file, as shown in Figure 6. Such files can be found in `/usr/share/calendar/en_UK/` on most distributions.

You can enter your own additional calendar data in your personal `.calendar` file. For example, on Debian and Ubuntu, copy the `/usr/share/calendar/en_UK/calender.all` file to `$HOME/.calendar`. Below the `#include` statements, enter your own anniversaries and events (in Listing 5, the entries for January 6 and 7). Each record begins with the month number, and you separate the following day with a slash. You use the tab key as a field separator between the date and the entry text.

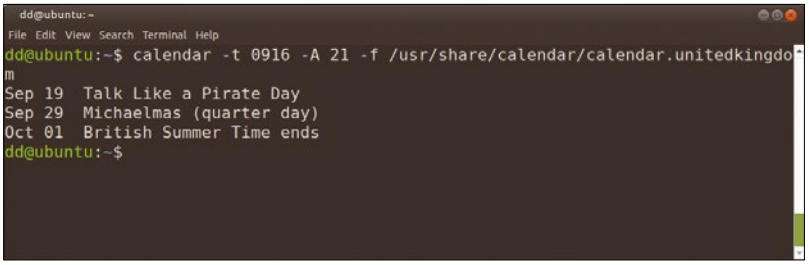


Figure 6: Restricting the query to UK events for a three-week period.

Taking a brief look at the `calendar.christian` file, you can see how `calendar` calculates the dependent holidays based on Easter Sunday. This function can be used to create your own calendars.

Table 2 contains some important options for using the program. You can save the call along with your preferred options as an alias in the shell's personal or system-wide configuration file.

Figure 7 shows some examples, where all commands refer to the user's personal calendar file. If you do not specify your personal file, Calendar resorts to the system-wide variant.

Calendars with pcal

The `pcal` program [6], which can be found in the repositories of all the popular distributions, creates a

Listing 5: Personal Calendar Entries

```
/*
 * UK Calendar
 */

#ifndef _en_UK_all_
#define _en_UK_all_

#include <en_UK/calendar.holiday>
#include <en_UK/calendar.history>
#include <en_UK/calendar.christian>
#include <en_UK/calendar.literature>
#include <en_UK/calendar.music>
#include <en_UK/calendar.science>

01/06  Annivery of First ascent Blaabjerg
       north face
01/07  Birthday mother in law

#endif /* !_de_DE_all_ */
```

Table 2: calendar Options

Action	Option
Specify the configuration file	`-f <file>`
Output the weekday	`-w`
Number of days to show (future)	`-l<number>` [1]
Number of days to show (past)	`-B<number>`
Show specific day	`-tYYYY-MM-DD`

[1] `-l0` restricts output to today

calendar in PostScript or HTML format. Because of the extensive options and the granularity of the calendar data, working with it seems complicated at first. The tool's most important options are summarized in Table 3. The program's man page could certainly be a tad more user-friendly.

You store the configuration of the desired calendar in a separate file, named `.pcal` in our example. In the file, you define a calendar year with its holidays. Listing 6 shows an example for Germany with special regional holidays for Bavaria.

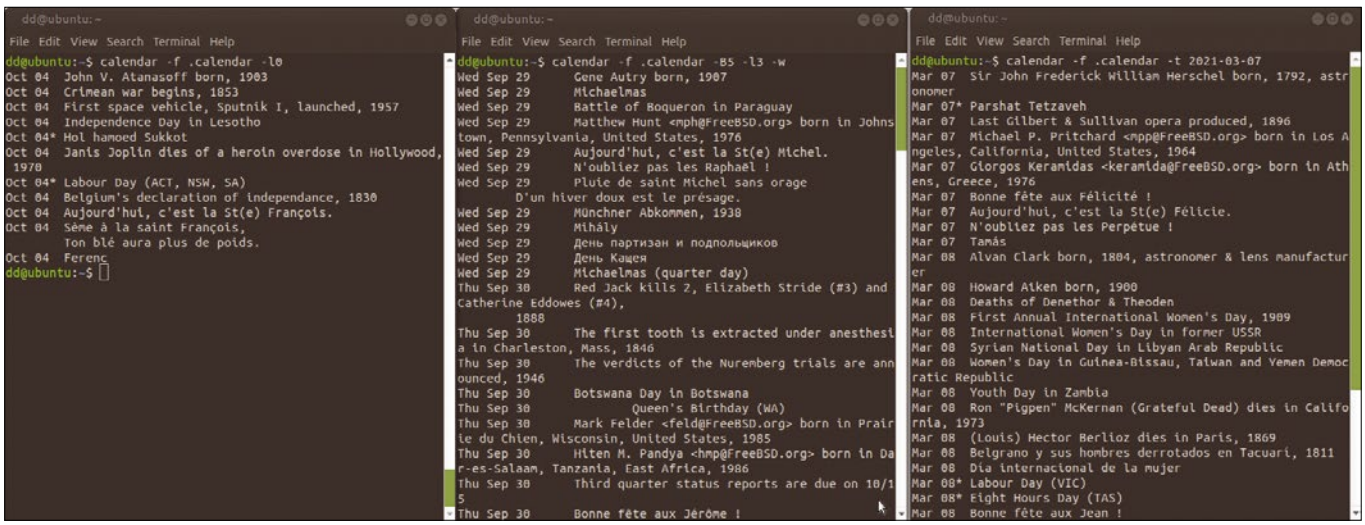
Some special rules apply here. There are different specifications for the specific day. With `all <day>`, you designate a day for which you want special highlighting. Listing 6, line 1, displays the word "Week" and the consecutive week number every Monday. Line 17 shows New Year's Day as a public holiday. There must be a space between the day and month entries.

To display a day as a public holiday, add an asterisk (`*`) followed by the desired text. Many holidays are a fixed number of days before or after

Table 3: pcal Options

Action	Option	Example
Specify calendar config file	`-f <file>`	`.pcal`
Select language	`-a <country>`	`en`
European date format	`-E`	–
Paper format	`-P <format>`	`a4`
Empties fields for days not in the month	`-B`	–
First weekday	`-F <number>`	`1` for Monday
Show moon phase daily	`-M`	–
Show moon phases	`-m`	Full moon, half moon, new moon
Font	`-n`	Helvetica, 12 point
Output file	`-o <file>`	–
Calendar month	`MM`	–
Calendar year	`YYYY`	–
Annual calendar on one page	`-w`	–
Color black for specified day	`-b <day>`	Saturday (`-b sat`)
Deviating color	`-g <day> -s <foreground>/<background>`	Red on light beige for Sundays and holidays: `1.0:0.0:0.0/1.0:1.0:0.8`

Figure 7: Examples with the user's personal `calendar.all` file.



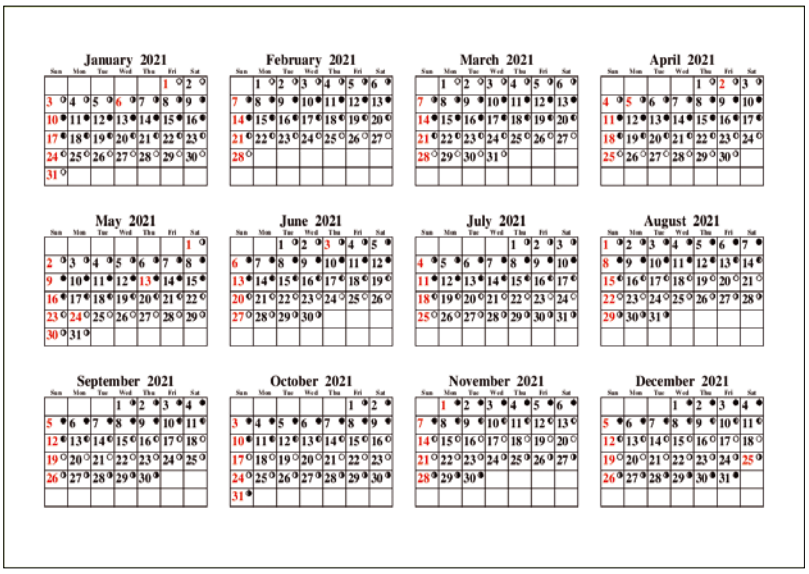


Figure 8: An annual calendar created with `pcal`.

Easter (see Listing 6, lines 4-15). Other holidays work like New Year's Day: You specify the holiday with a fixed date, including an asterisk if it's a public holiday.

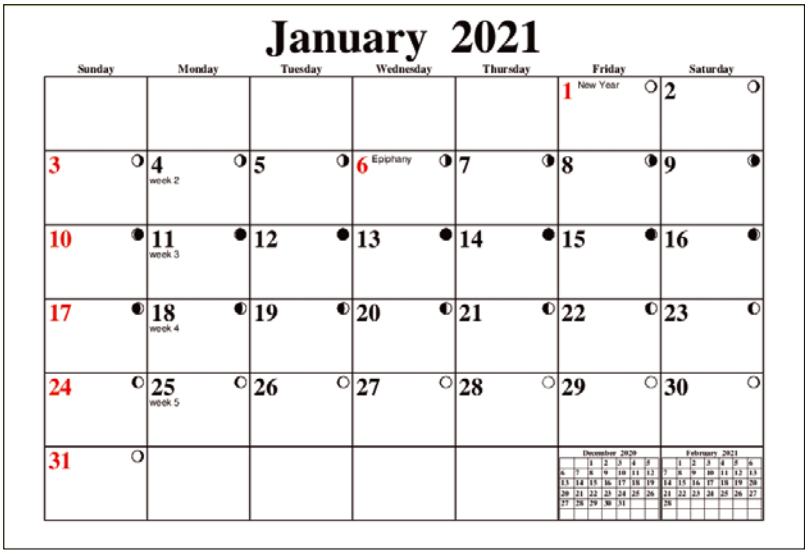
Christmas can be complicated because December 24 can also be the fourth Sunday in Advent (e.g., in 2023). You need to use `on_or_before` here. This allows December 24 to be the fourth Sunday in Advent if it happens to be on a Sunday. You can then define the dates of the remaining Sundays in Advent (see lines 29-32 of Listing 6).

Nonstandard characters can cause problems here. Modern distributions use UTF-8 encoding, but `pcal` insists on ISO-8859-15.

By default, `pcal` outputs Sundays and holidays in gray, which usually makes these days more prominent in black and white prints. You can create an overview calendar with Sundays and holidays printed in red and the moon phases, using the code in Listing 7. Figure 8 shows the results.

You can create a monthly calendar without the `-w` option; instead, you specify the month before the year (Listing 8). The result is shown in Figure 9. You can process the resulting PostScript file to create a complete calendar page by adding a photo, for example, using Inkscape.

Figure 9: A monthly calendar created using `pcal`.



Listing 6: Personal *.pcal* file

```
01 # Week number on Monday
02 all Monday in all week %w
03
04 Easter # +/- is computed relative to Easter
05 # * by date = holiday
06 Monday after Easter* Easter Monday
07 Good_Friday* Good Friday
08 49th day after Easter* Whit Sunday
09 50th day after Easter* Whit Monday
10 39th day after Easter* Ascension Day
11 60th day after Easter* Corpus Christi
12 # none * = not a holiday
13 48th day before Easter Shrove Monday
14 46th day before Easter Ash Wednesday
15 7th day before Easter Palm Sunday
16
17 01. 01.* New Year
18 06. 01.* Epiphany
19 01. 05.* Labor Day
20 08. 08.* Peace Festival (Augsburg)
21 15. 08.* Assumption
22 03. 10.* German Unification Day
23 01. 11.* All Hallows
24 24. 12. Christmas Eve
25 25. 12.* 1st Day of Christmas
26 26. 12.* Boxing Day
27 31. 12. New Year's Eve
28
29 sun on_or_before 24. 12. 4th Sunday in Advent
30 sun on_or_before 17.12. 3rd Sunday in Advent
31 sun on_or_before 10.12. 2nd Sunday in Advent
32 sun on_or_before 3.12. 1st Sunday in Advent
```

Listing 7: Annual Calendar

```
$ pcal -f .pcal -a en -E -P a4 \
 -B -F 0 -M -g sun \
 -s 1.0:0.0:0.0/1.0:1.0:0.8 \
 -b sat -n Helvetica/10 -w \
 -o Calendar.ps 2021
```

Listing 8: Monthly Calendar

```
$ pcal -f .pcal -a en -E -P a4 \
 -B -F 0 -M -g sun \
 -s 1.0:0.0:0.0/1.0:1.0:0.8 \
 -b sat -n Helvetica/10 \
 -o January-2021.ps 01 2021
```

Appointment Management

A small but good command-line program named `calcurse`, which you will find in the repositories of all the popular distributions, lets you manage general tasks and tasks with deadlines. The simple interface is more or less self-explanatory. If you have worked with Nano before, you will get along with the menu control used in `calcurse` right away. You can jump between the *Appointments*, *Calendar*, and *To do* fields by pressing the tab key; you use the arrow keys to reach targets in your appointments and calendar. Figure 10 shows the full screen.

To enter a new appointment, press *A* and enter the start time, the end time, and the description text (Figure 11). Pressing Enter saves the new appointment (Figure 12). Press Ctrl+T to create a new task (Figure 13). Start by entering the task text, then press Enter, and then set a priority. This affects the order in which tasks are displayed. Figure 14 shows the appointment and task entries.

Pressing *O* takes you to more commands and the settings. You can export the calendar entries in the ICAL or PCAL format or import data in these formats. If you call `calcurse` with the `-a` option, it displays all appointments for the current date onscreen (Figure 15).

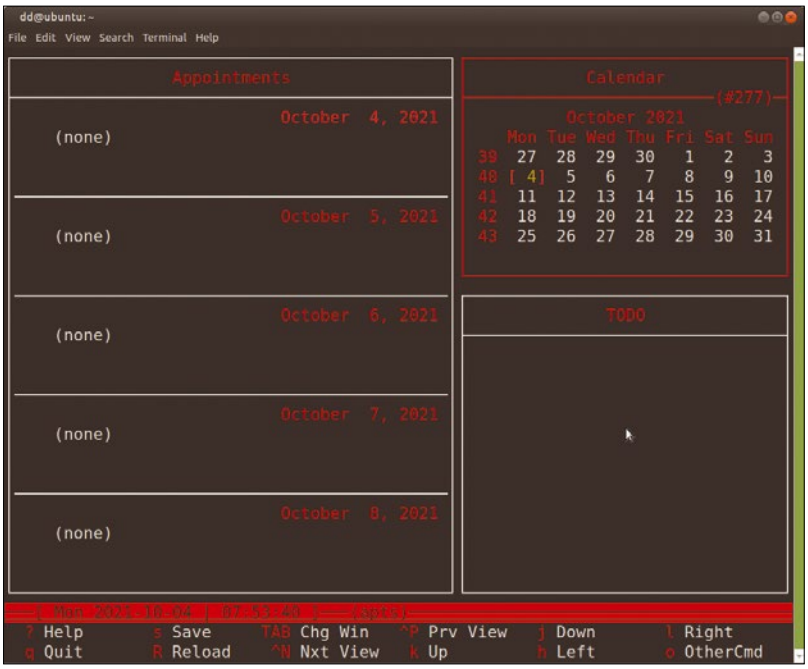


Figure 10: With `calcurse`, you can manage tasks and deadlines.

Remind

Due to space limitations, I did not cover the Remind [7] console program in this issue: Its possibilities easily fill a separate article.

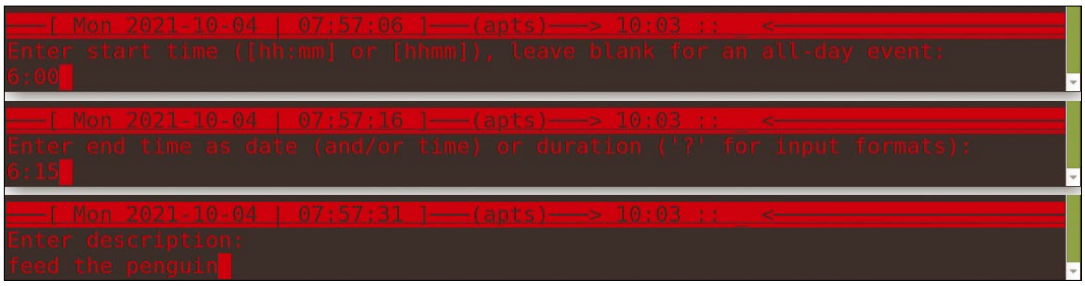


Figure 11: In `calcurse`, enter an appointment's start time (top), end time (middle), and a description text (bottom).

Conclusions

Creating calendars in the shell is a painless process, with numerous settings available for designing your final product. Simple date entries in the shell can be easily handled using your own scripts given the right choice of tools. You can also create selection calendars as an alternative for entering dates using `dialog`, YAD, or Zenity [8]. ■■■

Figure 12: The complete appointment in `calcurse`.

Figure 13: In `calcurse`, you enter tasks (top) and set a priority for them (bottom).

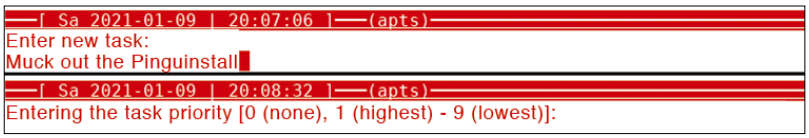


Figure 14: Appointment and task entries in `calcurse`.

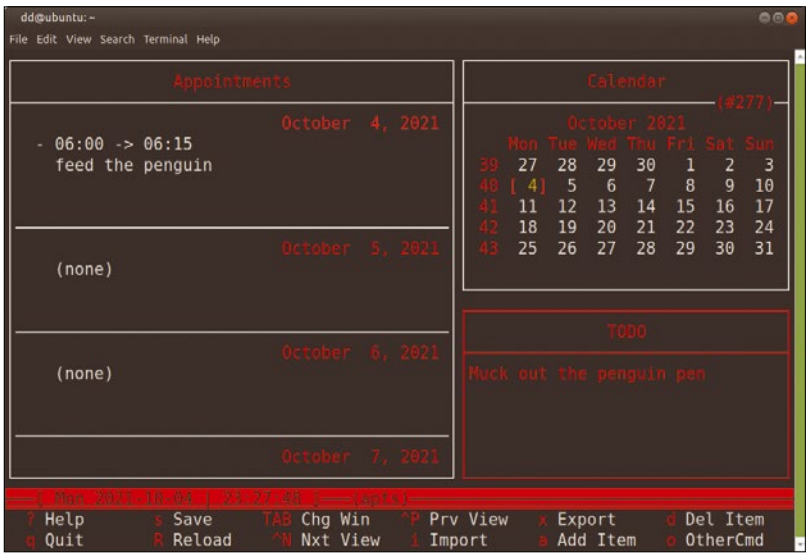


Figure 15: An appointment query for the current date.

```
dd@ubuntu:~$ calcurse -a
10/04/21:
 - 06:00 -> 06:15
        feed the penguin
dd@ubuntu:~$
```

The Author

Harald Zisler has been working with FreeBSD and Linux since the early 1990s. He writes magazine articles and books on technology and IT topics.

Info

[1] DDIN 5008 date format: *https://en.wikipedia.org/wiki/DIN_5008*

[2] ISO 8601 date format: *https://en.wikipedia.org/wiki/ISO_8601*

[3] dialog: *https://linux.die.net/man/1/dialog*

[4] YAD: *https://github.com/v1cont/yad*

[5] "Create a Select Menu with smenu" by Harald Zisler, *Linux Magazine*, issue 205, December 2017,*https://www.linux-magazine.com/index.php/Issues/2017/205/smenu/(language)/eng-US*

[6] pcal: *http://manpages.ubuntu.com/manpages/bionic/man1/pcal.1.html*

[7] Remind: *https://dianne.skoll.ca/projects/remind/*

[8] "Create Menus and Dialogs for Shell Scripts" by Harald Zisler, *Ubuntu User*, issue 21, 2014, *https://www.ubuntu-user.com/Magazine/Archive/2014/21/Create-menus-and-dialogs-for-shell-scripts*

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.

#253/December 2021

OpenBSD

BSD Unix has been around longer than Linux, and it still has a loyal following within the Free Software community. This month we explore the benefits of a leading BSD variant from the viewpoint of a Linux user.

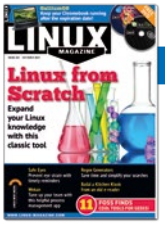
On the DVD: Tails 4.22 and Q4OS 4.6

#252/November 2021

Locked Down!

The security landscape keeps changing, and experienced users know they need to keep their eyes open for tools and techniques that give an edge. This month we study smartcards, hard drive encryption, and a less-bloated alternative to Sudo.

On the DVD: Debian 11 and Redcore Linux 2101



#251/October 2021

Linux From Scratch

Building an operating system is not like compiling a desktop app. You'll need to create a complete development environment – and if you follow the steps carefully, you'll emerge with a deeper understanding of Linux.

On the DVD: Linux Mint 20.2 Cinnamon Edition and Garuda Linux KDE Dr460nized Edition

#250/September 2021

Inside the Kernel

The only real way to celebrate the 30th anniversary of Linux is to write about Linux itself – not the agglomeration of software we know as a Linux distro, but the real Linux – the beating heart in the center of it all: the Linux kernel.

On the DVD: AlmaLinux Minimal 8.4 and SystemRescueCd 8.03

#249/August 2021

Turn Your Android into a Linux PC

UserLAnd lets you run Linux applications on your Android phone – all without replacing Android OS.

On the DVD: openSUSE Leap 15.3 and Kubuntu 21.04 Desktop

#248/July 2021

Brain Tools

Sometimes you want the computer to think for you, and sometimes you want the computer to make you think. This month we present a selection of free Linux tools for learning and thinking.

On the DVD: Ubuntu 21.04 and Fedora 34 Workstation

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*.

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.

DeveloperWeek

Date: February 2-9, 2021

Location: Oakland, California and Online

Website: *https://www.developerweek.com/*

DeveloperWeek 2022 is the largest developer conference & event series including the DeveloperWeek 2022 Conference & Expo – 1,000+ attendee hackathon, 800+ attendee tech hiring expo, and a series of workshops, technical talks, and keynotes. This year, the entire event series will be hosted online!

SCaLE 19x

Date: March 3-6, 2022

Location: Pasadena, California

Website: *https://register.socallinuxexpo.org/reg6/*

The SCaLE team is pleased to announce that the 19th Annual Southern California Linux Expo – SCaLE 19X – will be held on March 3-6, 2022 at the Pasadena Convention Center in Pasadena, California, near Los Angeles. Registration is open now.

Events

KubeCon + CloudNativeCon + Open Source Summit China	December 9-10	Virtual Event	https://www.lfasiallc.com/kubecon-cloudnativecon-open-source-summit-china/
Open Source Summit Japan	December 14-15	Virtual Event	https://events.linuxfoundation.org/
Open Compliance Summit	December 16	Virtual Event	https://events.linuxfoundation.org/
DeveloperWeek	February 2-4	Virtual Event	https://www.developerweek.com/
SCaLE 19x	March 3-6	Pasadena, California	https://www.socallinuxexpo.org/scale/19x
Open Networking & Edge Summit Europe 2022	March 8-9	Antwerp, Belgium	https://events.linuxfoundation.org/about/calendar/
CloudFest 2022	March 22-24	Europa-Park, Germany	https://registration.cloudfest.com/registration?code=CFMEDIA22
LinuxFest Northwest 2022	April 22-24	Virtual Event	https://lfnw.org/conferences/2022
DrupalCon Portland 2022	April 25-28	Portland, Oregon	https://events.drupal.org/portland2022
Linux Storage, Filesystem, MM & BPF Summit	May 2-4	Palm Springs, California	https://events.linuxfoundation.org/lsfmm/
KubeCon + CloudNativeCon Europe 2022	May 17-20	Valencia, Spain	https://events.linuxfoundation.org/
OpenJS World 2022	June 7-8	Austin, Texas	https://events.linuxfoundation.org/openjs-world/

CALL FOR PAPERS

We are always looking for good articles on Linux and the tools of the Linux environment. Although we will consider any topic, the following themes are of special interest:

- System administration
- Useful tips and tools
- Security, both news and techniques
- Product reviews, especially from real-world experience
- Community news and projects

If you have an idea, send a proposal with an outline, an estimate of the length, a description of your background, and contact information to *edit@linux-magazine.com*.

The technical level of the article should be consistent with what you normally read in *Linux Magazine*. Remember that *Linux Magazine* is read in many countries, and your article may be translated into one of our sister publications. Therefore, it is best to avoid using slang and idioms that might not be understood by all readers.

Be careful when referring to dates or events in the future. Many weeks could pass between your manuscript submission and the final copy reaching the reader's hands. When submitting proposals or manuscripts, please use a subject line in your email message that helps us identify your message as an article proposal. Screenshots and other supporting materials are always welcome.

Additional information is available at:
http://www.linux-magazine.com/contact/write_for_us.

Authors

Contact Info

Editor in Chief
Joe Casad, jcasad@linux-magazine.com

Copy Editors
Amy Pettle, Aubrey Vaughn

News Editor
Jack Wallen

Editor Emerita Nomadica
Rita L Sooby

Managing Editor
Lori White

Localization & Translation
Ian Travis

Layout
Dena Friesen, Lori White

Cover Design
Dena Friesen

Cover Image
© digitalshape and wannawit vachirakhueankhan, 123RF.com

Advertising
Brian Osborn, bosborn@linuxnewmedia.com
phone +49 8093 7679420

Marketing Communications
Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
4840 Bob Billings Parkway, Ste 104
Lawrence, KS 66049 USA

Publisher
Brian Osborn

Customer Service / Subscription
For USA and Canada:
Email: cs@linuxpromagazine.com
Phone: 1-866-247-2802
(Toll Free from the US and Canada)

For all other countries:
Email: subs@linux-magazine.com

www.linuxpromagazine.com – North America
www.linux-magazine.com – Worldwide

Printed in Nuremberg, Germany by hofmann infocom GmbH.

Distributed by Seymour Distribution Ltd, United Kingdom

LINUX PRO MAGAZINE (ISSN 1752-9050) is published monthly by Linux New Media USA, LLC, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA. Periodicals Postage paid at Lawrence, KS and additional mailing offices. Ride-Along Enclosed. POSTMASTER: Please send address changes to Linux Pro Magazine, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA.

Published monthly in Europe as Linux Magazine (ISSN 1471-5678) by: Sparkhaus Media GmbH, Bialasstr. 1a, 85625 Glonn, Germany.

Issue 255 / February 2022

Fuzzing

Experts use this automating testing technique to build better and safer software. Next month we help you get started with Fuzzing.

Approximate On Sale Date

UK / Europe	Jan 01
USA / Canada	Jan 28
Australia	Feb 28

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***

Custom Tailored Suit

Select essential parts for your mobile computing needs

Processor
Intel or AMD CPU

Graphics Card
Integrated or dedicated

System Memory
Upgrade any time

Data Storage
Fast, large and switchable

Network
Wireless, Cellular or Gigabit LAN

100% Linux

5 Year Warranty

Lifetime Support

Built in Germany

German Privacy

Local Support

TUXEDO COMPUTERS

tuxedocomputers.com

HETZNER

NEW LOCATION IN THE USA

CLOUD SERVER

STARTING AT

$4.61

monthly

HETZNER CLOUD SERVER CPX11

- ✔ AMD EPYC™ 2nd Gen
- ✔ 2 vCPU
- ✔ 2 GB RAM
- ✔ 40 GB NVMe SSD
- ✔ 20 TB traffic inclusive
- ✔ Intuitive Cloud Console
- ✔ Located in Germany, Finland or USA

EPYC AMD

HIGH QUALITY - UNBEATABLE PRICES

DEPLOY YOUR HETZNER CLOUD IN UNDER **10 SECONDS!**

MANAGE YOUR CLOUD QUICK AND EASY WITH FEATURES LIKE

LOAD BALANCER, FIREWALLS, ONE CLICK APPS AND MANY MORE!

GET YOUR CLOUD NOW

+1 646 6858477

CLOUD.HETZNER.COM