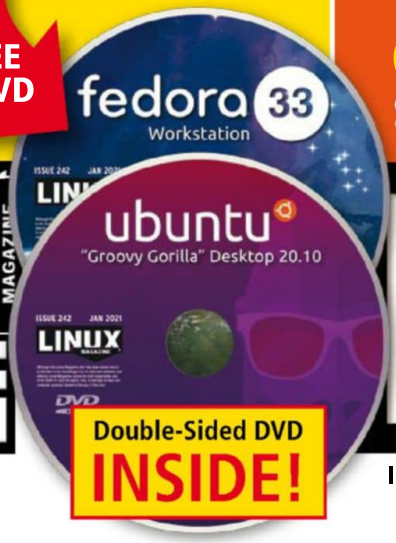
FREE DVD



GO PROGRAMMING
Search for files on Google Drive

3D PRINTING

LINUX MAGAZINE

ISSUE 242 – JANUARY 2021

3D PRINTING

From first steps to a finished creation

MOFO Linux
Steer around censorship with this privacy-focused distro

Nyttig
Share files and stream Internet radio from a Rasp Pi

Netdata
Zero configuration monitoring tool

Nerf Dart Game
Score points for your next neighborhood party

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

LINUXVOICE

- Stacer: Clean up your system
- maddog: Morality and licensing
- JessyInk: Presentations in Inkscape

FOSSPicks
- QGIS spatial data environment
- PrettyEQ audio equalizer

Tutorial
- Track your investments with Portfolio Performance

LINUX NEW MEDIA
The Pulse of Open Source

KEEP SQUAWKING

Dear Reader,

Today I'm remembering an episode that happened a few years ago. We are still a proud print publishing company, but, like most publishers, we deliver some of our copies in electronic form through content platforms available for personal computers and mobile devices. One of those platforms at the time was Apple Newsstand, a virtual newsstand for iPhone and iPad devices. The user interface for Apple Newsstand looked just like a magazine shelf at a bookstore, with a picture of a bunch of magazine shelves. Then, if you purchased a magazine, an icon with the cover of the magazine appeared on the virtual magazine shelf.

The goal of Apple Newsstand was to be a perfect little replacement of a real neighborhood newsstand – you buy any magazine you want (from Apple), and all the magazines you buy line up along your own personal virtual newsstand. The international company I worked for back then had several magazines, and they submitted applications for their magazines to be on Apple Newsstand. Linux magazines? No problem. System administration magazines? Sure. Raspberry Pi and Drupal magazines? Of course. But when we submitted a request to sell an Android magazine, the application was quickly rejected. It turns out that Apple was banning anything that mentioned Android from their newsstand because Android was competing against the iPhone and (according to them), it was an inferior, copycat technology. In other words, Apple Newsstand was a perfect little replacement for a real neighborhood newsstand, except that it inhabited an imaginary universe in which Apple's enemies did not exist.

Apple could have argued that it was their store and they could sell whatever they wanted, but there is something a little disingenuous about that argument. Apple Newsstand wasn't just a store – it was *the way* to consume magazines on an Apple system. It was an integral part of an ecosystem that purported to present a virtual version of everyday life. Since then, several other browser-based magazine platforms have appeared, and Apple Newsstand has itself been retired in favor of newer tech, so it is difficult to compare it to the situation today, but at the time, it seemed a lot like the monopolistic practices of Microsoft back in the browser war days, when "control of the desktop" meant control of the complete user experience, only Apple wielded a much more powerful form of control. The episode reflects a common theme in the evolution of the electronic marketplace, in which the user voluntarily surrenders freedom and privacy for the chance to be the one who is seen dabbling with smooth and shiny technology.

Fast forward to this month, and a new chapter in this saga is playing out with Apple's new macOS 11.0 "Big Sur" system. A blog post by security expert Jeffrey Paul [1] outlines some of activities that Big Sur attends to in secret, including the fact that it "…sends to Apple a hash (unique identifier) of each and every program you run, when you run it." Paul adds that "Because it does this using the Internet, the server sees your IP, of course, and knows what time the request came in. An IP address allows for coarse, city-level and ISP-level geolocation, which allows for a table with the following headings: Date, Time, Computer, ISP, City, State, Application Hash… This means that Apple knows when you're at home. When you're at work. What apps you open there, and how often."

Of course, Apple and other vendors have been spying on their users for years, but the thing that is most striking about this, other than the sheer audacity of it, is that there doesn't seem to be any way to actually turn it off. (In the past, there was usually some way to suspend the surveillance if you really took the time and tried hard enough.)

Another weird thing that Paul and other security researchers have discovered is that even a VPN or user-controlled firewall can't stop your Mac from leaking this information to Apple. According to the report, if you intentionally configure your system to route all traffic through a VPN, it will use the CommCenter component (used for making phone calls) to maintain its own connection for sharing reports on your behavior.

Sometime after Paul's blog post appeared, Apple posted their own statement [2], saying that the app logging technology was a thing called Gatekeeper, which checks the developer ID signature to ensure that the app has not been altered by malware. Apple says "We have never combined data from these checks with information about Apple users and their devices." They don't offer any proof – you're just supposed to take their word for it. They also don't promise they *will never* combine the data in the future – just that (if you take their word for it) they aren't doing it now.

The way this privacy trampling works is that the vendor tries something, and if no one says anything, it becomes the new norm. If enough people squawk, they say "OK never mind…" and dial it back some. Perhaps in response to the negative feedback they have received so far, Apple has now stated that they will create a new preference setting sometime in the next year that will allow users to opt out of the Gatekeeper reporting feature. They still haven't explained why they thought it was reasonable that a computer configured to route all traffic through a VPN would continue to send information to the vendor through a secret backdoor – and what other information they might be sending in addition to the Gatekeeper data. We might have to keep squawking about that part.

Joe

Joe Casad, Editor in Chief

Info

[1] "Your Computer Isn't Yours": *https://sneak.berlin/20201112/your-computer-isnt-yours/*

[2] "Safely Open Apps on Your Mac": *https://support.apple.com/en-us/HT202491*

LINUX MAGAZINE

WHAT'S INSIDE

The weird, wonderful, futuristic world of 3D printing is waiting for you right now if you're willing to invest a little time and energy. This month we help you get started with practical 3D printing in Linux. Also in this month's issue:

- **TOPSIS** – Use Python and the TOPSIS urban planning technique to choose the best solution from a group of competing alternatives (page 40).
- **Nyttig** – Turn your Raspberry Pi into a personal micro server (page 50).

Look in MakerSpace for a tutorial on building an electronic Nerf ball game, and check out the Stacer system maintenance tool in this month's Linux Voice.

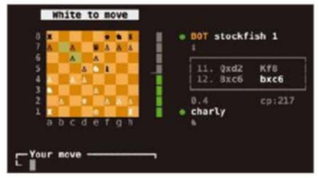
SERVICE

NEWS

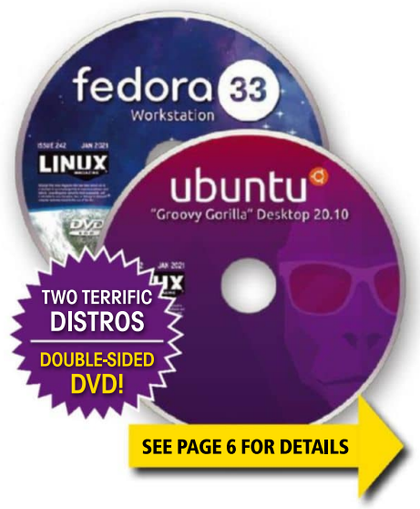
COVER STORIES

REVIEWS

IN-DEPTH



MAKERSPACE

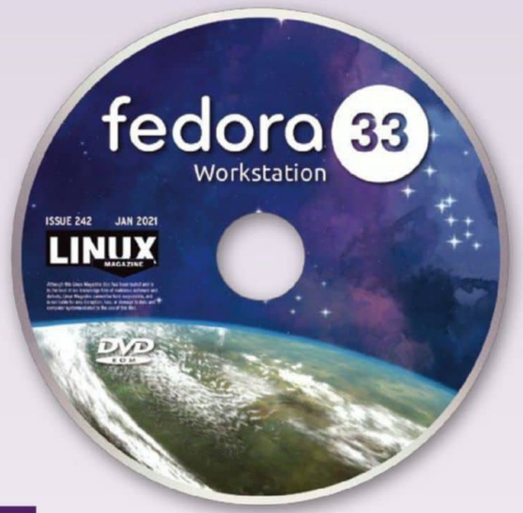


SEE PAGE 6 FOR DETAILS

LINUXVOICE

Ubuntu 20.10 and Fedora 33

Two Terrific Distros on a Double-Sided DVD!



Ubuntu 20.10
64-bit

Codenamed Groovy Gorilla, Ubuntu 20.10 is the latest release in the popular desktop distribution. A short-term release, it will be supported for nine months until July 2021, with an easy upgrade path to the next release. The next Long Term Support (LTS) release is not scheduled until April 2021.

Like all Ubuntu releases, Groovy Gorilla draws many of its packages from the Debian Unstable repository. In keeping with the tradition for short-term releases, it contains no major innovations. However, Groovy Gorilla does include one or two new features that might interest the curious. Chief among these are WiFi sharing through the use of QR codes. In addition, should you like what you see on the DVD, you might try Groovy Gorilla as a desktop on a 4MB or 8MB Raspberry Pi.

Other noteworthy features are included thanks to the shipping of Gnome 3.38. For instance, users can now manually drag and drop icons. When more than nine icons are present in a folder, pagination keeps them from taking up room. Several new features center on the message tray, where upcoming calendar events now display below the calendar widget in the message tray and a battery indicator and reboot option now appear.

Groovy Gorilla is ideal for all levels of users, but especially for Linux novices and the moderately experienced.

Fedora 33
64-bit

Fedora is both a popular community distribution and a testing ground for Red Hat Enterprise Linux and CentOS. However, even by Fedora's standards, Fedora 33 is one of the most innovative releases in years. Not only does Fedora 33 have numerous package updates and the enhancements of Gnome 3.38, but it also includes several modifications to key system components.

One of the most noticeable changes is a default to Btrfs filesystems. This change brings enhanced data integrity and compression, system snapshots, and support for multiple snapshots. Even more important is the increasing optimization and monitoring of the increasingly solid-state drives (SSDs). In addition, Fedora 33 uses zram for the swap partition, a move that frees up disk space and speeds up swapping. Another change is the use of systemd-resolved in the hopes of reducing differences between different distributions and providing network name resolution.

Note, too, that Fedora 33 will not install with Secure Boot. Thanks to the Boot Hole vulnerability, the certificate to sign Fedora's bootloader will be revoked soon and not replaced until the middle of 2021.

Fedora has provided a reliable user experience for years, and these innovations should not change that. However, the innovations do mean that Fedora 33 will have a special appeal to experts.

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

NEWS

Updates on technologies, trends, and tools

THIS MONTH'S NEWS

Dell to Enable Privacy Controls for Linux Hardware

Dell has completely embraced the Linux community. With one of the finest Linux-based laptops on the market (the XPS Developer Edition), Dell is definitely no stranger to open source and the needs of those who use it. Case in point, many Linux users are quite savvy in the realms of privacy and security. With most modern laptops equipped with webcams and microphones, which can be used (either by accident or via nefarious means) in less-than-desirable ways, it has become crucial for many users to have control over those features.

To that end, Dell has offered up code to the Linux kernel for the Dell privacy drivers. What this feature will do is allow users (via keyboard shortcuts) to disable the built-in microphone and/or webcam on supported devices. The applicable shortcuts will be Ctrl+F4 for the microphone and Ctrl+F9 for the webcam.

The Dell privacy drivers will go a long way to prevent malicious (or accidental) usage, but users will have to wait until 2021 for the new feature to find its way into Dell hardware.

It should also be noted that, tucked away in the code, there is mention of `PRIVACY_SCREEN_STATUS`, which could extend the privacy driver functionality to horizontal and vertical viewing angles of the screen.

For more information about Dell privacy drivers, check out the *kernel.org* entry (*https://lore.kernel.org/lkml/20201103125542.8572-1-Perry_Yuan@Dell.com/*).

Linux Mint Unveils New Packages

For those who prefer their Linux a bit mintier, but aren't terribly keen on everything installed via Snap, the developers of Linux Mint have announced they'll be bringing an official Chromium package to the next release of the distribution (20.1, aka Ulyssa). Unlike some Ubuntu-based distributions, Linux Mint users will be able to install Chromium from the traditional Apt repositories, instead of having to go with the Snap package (which is blocked by default). That installation is as simple as `sudo apt-get install chromium -y`.

© belchonock, 123RF

This decision wasn't just made because of Snap. According to the Linux developers, this was about release delays. To that, their official take is, "We noticed significant delays between official releases and the versions available in almost all Linux distributions. For this reason we set up our own packaging and we're building directly from upstream."

Chromium isn't the only package getting attention in the upcoming release. The Linux Mint developers have also created an M3U IPTV player, called the Hypnotix IPTV player. This new media player will connect with FreeIPTV to stream a variety of television shows. Hypnotix IPTV player is very much in the developmental stage, but the prototype can be downloaded and installed already (*https://linuxmint.com/tmp/blog/3978/hypnotix_1.0.0_all.deb*).

For more information about the development of Linux Mint 20.1, check out the official announcement (*https://blog.linuxmint.com/?p=3969*).

Pop!_OS 20.10 Now Supports DEB822 Format

Most users of Debian-based distributions are familiar with the single line Apt repository format that includes all of the information for a repository. With the new DEB822 format, those single lines are converted to multi-line entries, which allow more flexibility and extensibility over the standard for Debian software repositories.

In fact, the single-line format standard is planned for depreciation, so most distributions will eventually convert over. For Pop!_OS users, that time is now. Pop!_OS is the Linux distribution created by System76 as both a developer and general usage desktop operating system. Up until 20.10, Pop!_OS defaulted to the single-line style of repository entries. Now, instead of the `/etc/apt/sources.list` file, you'll find those repository entries in `/etc/apt/sources.list.d/system.sources`. As well, instead of a number of entries, (at least as of the initial release) you'll find only one in that file, which follows the new DEB822 format of:

- URIs – repository address
- Suites – various release-related repositories (such as `groovy`, `groovy-security`, `groovy-updates`, and `groovy-backports`)
- Components – main and other repositories (such as universe and multiverse)

The idea behind DEB822 is that it should make it easier for users, developers, and even machines to create, extend, and modify Apt entries, especially if a larger number of sources and/or options are involved.

Although not all repositories work with the new format yet, developers are being encouraged to switch to the new multi-line style.

For more information on what's new with Pop!_OS 20.10, check out the official System76 blog (*https://blog.system76.com/*).

Ubuntu 20.10 with Raspberry Pi Support

For any Linux admin who's been looking to deploy single board computers for various purposes, there's a new (while at the same time old) player in the Raspberry Pi mix – Ubuntu 20.10. Groovy Gorilla is the first official Ubuntu release to not only be optimized for the Raspberry Pi as a server distro, but as a full-blown desktop as well.

To make this even more appealing, Ubuntu 20.10 will include the likes of LXD 4.6 and MicroK8s for the easy deployment of resilient micro clouds, small clusters of servers providing virtual machines, and Kubernetes on demand at the edge.

Any Raspberry Pi 4 board with 4GB or 8GB of RAM can be deployed with Ubuntu Desktop or Server. And this isn't a stripped-down version of the platform; it's the full monty. Canonical has put in a ton of work to optimize Ubuntu for Raspberry Pi. According to the Ubuntu PR machine, "With this release, Ubuntu is optimized for Raspberry Pi, giving users of all levels and capabilities the access to Linux and microcloud technologies."

Mark Shuttleworth, CEO of Canonical, said of this iteration, "In this release, we celebrate the Raspberry Pi Foundation's commitment to put open computing in the hands of people all over the world." Shuttleworth continues, "We are honoured to support that initiative by optimising Ubuntu on the Raspberry Pi, whether for personal use, educational purposes or as a foundation for their next business venture."

Downloads will be made available starting October 22, 2020 from the official Ubuntu Raspberry Pi download page.

MORE ONLINE

Linux Magazine
www.linux-magazine.com

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

mpi4py – High-Performance Distributed Python
• Jeff Layton
Python, I think it's safe to say, is the most prevalent language for machine learning and deep learning and has become a popular language for technical computation, as well.

Why Good Applications Don't Scale
• Jeff Layton
You just bought a new system with lots and lots of cores (e.g., a desktop with 64 cores or a server with 128 cores). Now that you have all of these cores, why not take advantage of them by parallelizing your code?

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

Backup and Restore Disks
• Erik Bärwaldt
Not every backup tool is useful for backing up entire data media. Rescuezilla makes perfect bit-level copies of mass media in a matter of minutes, so you can completely reconstruct the system if disaster strikes.

Visualizing Containers with Clarity
• Chris Binnie
The Dockly terminal tool helps you troubleshoot and manage multiple Docker containers.

Seven Free Blocking Filters for Ads
• Markus Stubbig
We compare seven free blocking filters for advertisements and look at how they integrate into the network.

SaltStack Acquisition Brings More Automation to VMware

After reading about the VMware purchase of SaltStack, a thought occurred to me that this acquisition might have well been pursued for a single purpose – to compete with Kubernetes. Although VMware is the ruler of the virtual machine space, as is, they didn't have the technology to compete with the cluster management found in Kubernetes.

© iofoto, Fotolia

So we asked Purnima Padmanabhan, VP and GM, Cloud Automation, Cloud Management Business Unit, VMware, about this very thing. Let's take a look at some of her responses.

Padmanabhan said, "First off, instead of competing with Kubernetes, VMware is committed to helping customers build, run, and manage their software on Kubernetes. For years, VMware has contributed to the Kubernetes community and related open source projects."

Padmanabhan continued, "Our goal is to help our customers accelerate their journey to modern, containerized applications, and, with our Tanzu portfolio, we are well positioned to make that happen." She then gave us the key to unlock the door to this mystery when she said, "In addition, VMware's acquisition of SaltStack furthers the goal, by helping our customers automate their complete application to infrastructure stack." She added that, "Through automation, we help customers shrink the time it takes them to release new applications to the market; continuously expand their application scale, scope and business impact; and dynamically adapt their application and cloud stack to meet their changing needs."

So it looks as though VMware plans on leveraging the power of SaltStack to help empower automation and (probably) CI/CD.

New Storage Model Could Replace POSIX

POSIX has been around for a long time, but within the realm of enterprise computing, it has its issues which limit the scalability of compliant systems. This is especially true for systems that make use of deep learning, artificial intelligence, and other data-intensive use cases.

That's where object storage comes in. This particular model doesn't require a hierarchical data structure. Instead, object storage makes use of flat pool data (where each piece of data is defined by its associated metadata). This type of storage has no scalability limitations, which could be a boon for large-scale applications.

A group of engineers (including Henry Scott Newman, Chief Technology Officer at Seagate Government Solutions) created mmap_obj(), which makes it possible for object storage systems to process data within memory. The `mmap_obj()` protocol is a software abstraction for low-latency access to files. Although not yet in production, this breakthrough could mean accessing data in filesystem storage would no longer be a challenge for systems that must work with very large amounts of data.

The `mmap_obj()` model creates a new mapping in the virtual address space and would greatly benefit low-latency hardware storage hardware, such as NVMe over Fabrics (NVMeoF) and Storage Class Memory (SCM).

To read more about `mmap_obj()`, check out the paper written by John Michael Bent, Ujjwal Lanjewar, Nikita Danilov, Scott Hoot, and Henry Scott Newman, entitled "User-level low-latency access via memory semantics to objects in object storage" (*https://priorart.ip.com/IPCOM/000263656*).

Zack's Kernel News

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

By Zack Brown

Dealing with Older GCC Versions

This past year, Linus Torvalds upped the supported version of GCC to version 4.9 or newer. The idea is that it's good to support older as well as newer versions, because then people running on old systems can still compile the kernel. At the same time, if no one in the world is using a particular older compiler, you might be able to simplify some kernel code by removing support for that compiler. So there's not only an incentive for the kernel developers to support older compilers, there's also an incentive for them to abandon very old compilers when they can safely get away with it.

It's an ever-advancing debate. Recently for example, Thomas Gleixner threw up his hands in disgust at some compiler behavior. Arnd Bergmann had posted some kernel patches to work around some ugly compiler behavior in GCC 4.9 that wasn't necessary in GCC 5 and newer.

Specifically, Arnd's code added some symbol references that were never actually used in the code. The symbols were then also referenced in unused inline functions, so as not to trigger a warning. Then, as Thomas pointed out, the symbols and the inline functions were all optimized out of the final compiled binary.

Thomas retched into his hand and said, "Bah, we really should have moved straight to GCC5 instead of upping it just to 4.9."

Nick Desaulniers and David Laight also retched into their hands, agreeing that this was a disgusting hack. David remarked, "It could be changed to use a symbol that the linker script already defines. However it does look like a workaround for a broken version of gcc."

A bunch of folks dove into GCC's documentation, trying to identify the intended behavior. But to some extent it was just an exercise, since the existing hack in the kernel code actually worked. A good thing too – because as Nick put it at one point, "From the link to GCC docs, the code in question doesn't even follow the pattern from the doc from informing the compiler of any dependency, it just looks like !@#$." And at another point Thomas remarked, "I clearly want a statement from the GCC people that this won't happen on pre gcc6 compilers and not just some 'works for me' statement based on a randomly picked compiler build."

At some point, Sedat Dilek said:

"There exist gcc-4.8 and gcc-4.9 for Debian/jessie where EOL was June 30, 2020 (see [https://packages.debian.org/search?keywords=gcc-4] and [https://wiki.debian.org/LTS])."

"In the latest available version '4.9.2-10+deb8u1' I see no PR82602 was backported (see [https://sources.debian.org/src/gcc-4.9/] and [https://sources.debian.org/src/gcc-4.9/4.9.2-10+deb8u1/debian/patches/])."

"I am asking myself who is using such ancient compilers? Recently, I threw away GCC-8 from my Debian system.

"If this is a real problem with GCC version < = 5, so can this be moved to a GCC specific include header-file? Thinking of include/linux/compiler-gcc.h or include/linux/compiler_types.h with a GCC-VERSION check?"

Segher Boessenkool, an actual GCC developer, replied, "There is GCC 4.9.4; no one should use an older 4.9." And he went on, "Some distros have a GCC 4.8 as system compiler. We allow building GCC itself with a compiler that far back, for various reasons as well (and this is very sharp already, the last mainline GCC 4.8 release is from June 2015, not all that long ago at all)."

Segher also remarked, "Does anyone actually test the kernel with old compilers? It isn't hard to build a new compiler." To which Arnd replied (with a non-zero quantity of snark), "there are a number of notable kernel developers that intentionally stick to old versions because of compile speed. Each major compiler release adds about 4% overhead in total time to compile a kernel, so between gcc-4.6 and gcc-11 you add over 50% in build time."

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.**

Miguel Ojeda said to Segher, "I am usually in favor of raising the minimum whenever new hacks are required to be added. On the other hand, we already raised the version twice this year and it is not clear to me what is the minimum version we would need to go for to ensure this does not bite us."

Linus Torvalds agreed with that basic idea. And he went on to say:

"The good news is that I haven't seen a lot of pushback on the gcc version updates so far. I was expecting some complaints. I haven't seen a single one.

"That may be because people did end up finding it very onerous and complained internally on channels I haven't seen, but it might also be indicative of us having perhaps been a bit too timid about compiler version updates.

"However, in this case, can we just leave that old '__force_order' hack alone, and to work around the clang thing, just make a dummy definition of it anyway.

"Alternatively, just use the memory clobber. We use memory clobbers elsewhere in inline asms to make sure they are serialized; it's not normally a huge problem. Both clang and gcc should be smart enough to know that a memory clobber doesn't matter for things like local variables etc that might be on stack but have never had their address taken.

"Or are there other cases than that particular __force_order thing that people now worry about?"

On the particular question of Arnd's original patch, and how to make a proper fix, Arvind Sankar replied to Linus, "Actually, is a memory clobber required for correctness? Memory accesses probably shouldn't be reordered across a CRn write. Is asm volatile enough to stop that or do you need a memory clobber?"

A "memory clobber" is when you want to do an operation that might affect more than just the input and output variables, so you need to save some state during the operation and then restore that state afterwards. Doing the copy involves a tiny performance hit, so it's avoided where possible.

In this case, Linus said to Arvind:

"You do need a memory clobber if you really care about ordering wrt normal memory references.

"That said, I'm not convinced we do care here. Normal memory accesses (as seen by the compiler) should be entirely immune to any changes we do [to] wrt CRx registers.

"Because code that really fundamentally changes kernel mappings or access rules is already written in low-level assembler (eg the entry routines or bootup).

"Anything that relies on the more subtle changes (ie user space accesses etc) should already be ordered by other things – usually by the fact that they are also 'asm volatile'.

"But hey, maybe somebody can come up with an exception to that."

The discussion petered out around here, with no clear conclusions. Pavel Machek had the final word in the thread, remarking, "I do have [GCC] 4.9.2 on some systems. They work well, and are likely to compile significantly faster than newer ones. Please don't break them."

On-boarding New Kernel Hackers

John Wood recently made his first security patch for the kernel. But before we get into that, I'll say that there is no real "on-boarding" process for new kernel hackers. If you're a coder and you ask anyone how to join, they'll probably tell you to subscribe to the mailing list, start sending in patches, and you'll get the hang of things as you go.

So John wrote a patch, and Kees Cook sent it to the mailing list for reasons that will become clear in a moment. The basic idea of the patch, as John said in the commit message, was "to detect and mitigate a fork brute force attack." Later, Jann Horn would point out that John's code would protect vulnerable userspace code, rather than the kernel itself. And in particular, John said, "Attacks with the purpose to break ASLR or bypass canaries traditionally use some level of brute force with the help of the fork system call. This is possible since when creating a new process using fork its memory contents are the same as those of the parent process (the process that called the fork system call). So, the attacker can test the memory infinite times to find the correct memory values or the correct memory addresses without worrying about crashing the application."

Address Space Layout Randomization (ASLR) is a kernel feature that takes an eggbeater to RAM so that attackers can't tell which part of the egg to attack.

A canary is a little piece of data that you stick in memory after your variables and before the address you'll jump back to at the end of your function call. If an attacker tries to write beyond the end of a variable in order to change that return address to aim at their own malicious code, they'll overwrite the canary. Then, before returning from that function, you can check the canary. If it's been altered, you know the return address has been hacked, and you can avoid using it.

John offered a summary of another attempt to deal with this, the grsecurity project. Specifically, he said that grsecurity intentionally added delays to the kernel code as a way to frustrate the attack. However, since grsecurity only added those delays when a child process crashed (i.e., when the attacker tried a random attack that then failed), an attacker could avoid the delay by forking off tons of child processes first and then analyzing them all in turn. Then they could crash all they wanted, because the attacker would already have gotten access to all those child processes.

A second way to avoid grsecurity's delay, John said, was to fork off a child process, but then attack the parent process instead of the child. After the parent process crashed, the child would still be available to do another fork and repeat the process. The delay would not affect the attacker, because the child process would already exist and be ready for more forking.

John's patch addressed both of those attack vectors. He said, "the solution for the first bypass method is to detect a fast crash rate instead of only one simple crash. For the second bypass method the solution is to detect both the crash of parent and child processes. Moreover, as a mitigation method it is better to kill all the offending tasks involve[d] in the attack instead of use delays."

To detect a fast crash rate, John wanted to share some statistical data across all the processes forked off of the original parent process. Specifically, he said, "these statistics hold the timestamp for the first fork (case of a fork of task zero) or the timestamp for the execve system call (the other case). Also, hold the number of faults of all the tasks that share the same statistical data since the commented timestamp."

If an ungodly number of processes crash really quickly, John said, it's a sure sign of shenanigans. At that point, the kernel would simply kill all the processes in that particular statistical group (i.e., in the process being attacked). Boom! No more attack.

Kees, although posting the patch on John's behalf, also replied to the post, saying, "Thanks for this RFC! I'm excited to get this problem finally handled in the kernel."

At this point, James Morris asked Kees, "Why are you resending this? The author of the code [John] needs to be able to send and receive emails directly as part of development and maintenance."

And John explained, "I tried to send the full patch serie[s] by myself but my email got blocked. After get support from my email provider it told to me that my account is young, and due to its spam policie I am not allow, for now, to send a big amount of mails in a short period. They also informed me that soon I will be able to send more mails. The quantity increase with the age of the account."

Kees and Jann had a bunch of comments about John's implementation. In particular, Jann felt that gathering all that statistical data might not be necessary, and that maybe the most recent five forks were enough to know if someone was killing them at a rapid rate. Specifically Jann suggested, "You could keep a list of the timestamps of the last five crashes or so, and then take action if the last five crashes happened within (5-1)*crash_period_limit time."

John liked that idea and said he planned to change his implementation to match.

For the moment at least, John's patch – in whatever form it may morph into – doesn't seem particularly controversial. It solves a known problem or at least takes a hefty bite out of it. But at the same time, there are many problems that can come up. Especially with security patches, you never know when someone will stick their head up at the last minute and say, "hang on a sec, I've got a problem with this." And then maybe the patch is accepted, or maybe you go back to the drawing board. ■■■

Getting started with the Ender 5 Pro 3D printer

Get Printing

Just unpack and get started? It's not so easy with hobby 3D printing. *By Oliver Nickel*

Whether it's decorative elements for a new case-modding project, custom trees to add a flourish to the Gloomhaven board game, or homemade gears for quirky gadgets, 3D printers make craft projects much easier – or so I thought. I had been struggling with my conscience for a long time about buying a 3D printer and weighing the arguments. Do I really need this? Isn't there a lot of work involved? Could I maybe invest the money in a better way?

Eventually, I gave in to my cravings when Chinese manufacturer Creality3D offered the Ender 5 Pro [1] for just EUR350 (~$414). Two weeks later, the package reached my home. After I unpacked, one of my first fears disappeared. I thought the DIY assembly would be complicated, but it was actually relatively easy (Figure 1). However, I soon discovered that the initial assembly was only the beginning of my quest. Of course, this story is only about one man and one 3D printer model, but many of the issues I faced are similar to problems you might see with other hobby printers. This article is intended as a practical case study on what it takes to get a 3D printer assembled and working in the real world.

Successful Build

The Ender 5 Pro arrives with some pre-assembled parts. You won't need to assemble the printing bed, the extruder, or the microcontroller (including the display). An easily understandable set of build instructions and the right tools for the bolting work are also included, and everything is packaged between soft foam pads.

The first thing to do is to connect various aluminum rods together and join the already-marked connectors for the axis motors, the distance sensors, the mainboard,

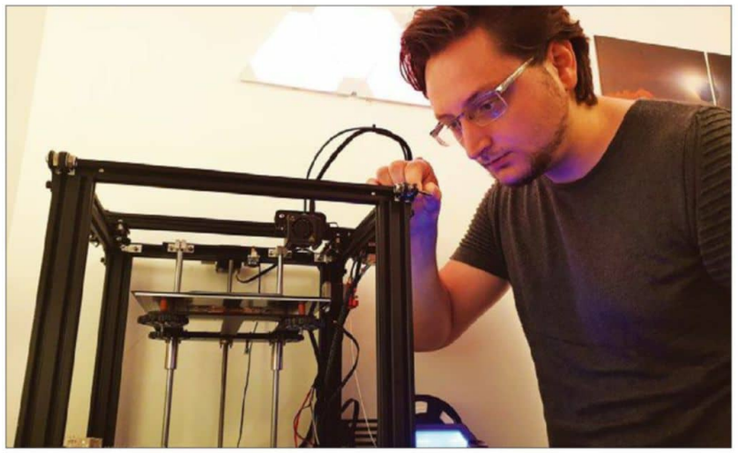
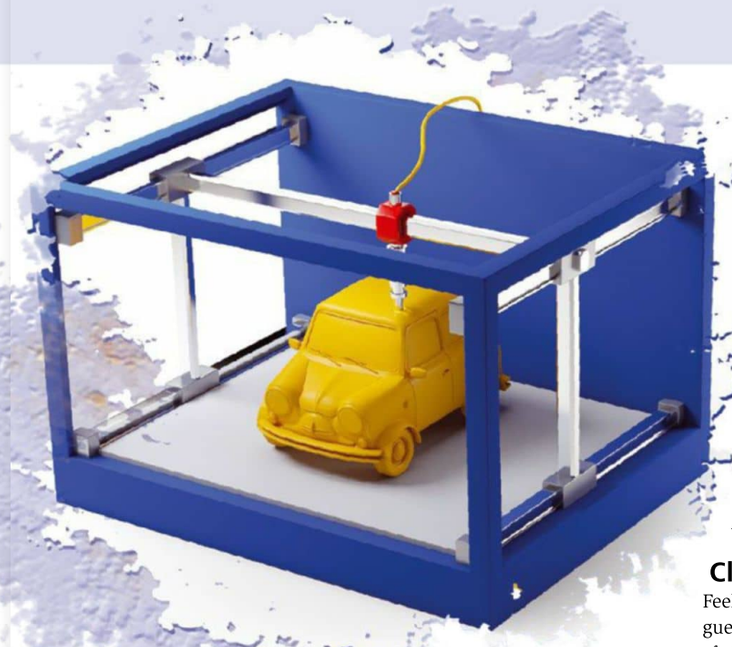


Figure 1: The easy-to-assemble Ender 5 Pro. © Oliver Nickel



and other parts. You can't really go wrong with these simple steps; however, the final build might look a bit improvised if you forget about the cable routing – like I did. For example, the ribbon cable leading to the nozzle was hanging loose in the air (Figure 2). At least everything worked after a short test.

Turning the Wheel: Calibration

If you think you can simply start printing after the build, forget it. It takes a huge amount of preparation to achieve good printing results: starting with calibrating the printing bed.

The Ender 5 Pro does not have an automated calibration function. Instead, you have to carefully push the printhead by hand into the four corners of the print bed. You can adjust the tilt and height with the four large adjustment screws underneath (Figure 3). Calibration is complete when a sheet of paper fits between the printing bed and the printing nozzle at all four corners.

The fact that the Ender 5 does not offer an automatic calibration function, even in the Pro version, makes the process somewhat frustrating. In my case, however, I first had to solve another problem that seemed less obvious: No filament came out of the nozzle, and the stepper motor, which is supposed to push the Polylactic Acid (PLA) filament material through a tube into the extruder, seized.

At first, I suspected faulty wiring of the motor causing it to turn in the wrong direction. I swapped the cables at the connector, which reversed the rotation direction of the motor as expected. By the way, this seems to be a fairly common problem with 3D printers.

In the case of the Ender 5 Pro, the direction of rotation can also be set via a firmware update and by changing a value in the configuration file. However, this option requires an Arduino Uno, which provides the bootloader for flashing the Marlin-based software [2] via a USB connection. I didn't have a suitable Arduino at the time of installation, but that didn't matter: A wrongly adjusted motor was not the problem.

Clogged Nozzles

Feeling significantly more frustrated, I turned to my second guess – that the nozzle itself might be clogged. With a pair of pliers, and with some effort, I loosened the feed tube of

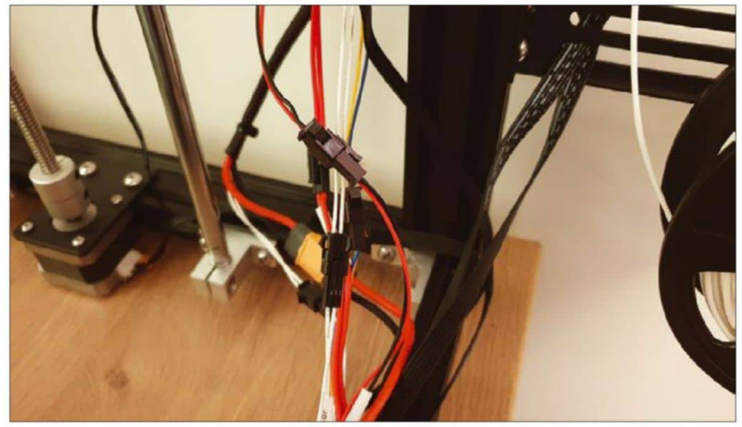


Figure 2: The cable routing has some room for improvement. © Oliver Nickel

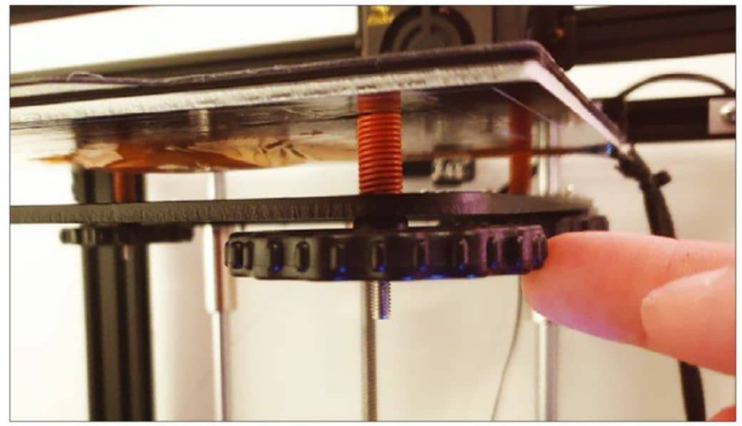


Figure 3: Calibrating the bed is a painstaking task. © Oliver Nickel

the extruder and pressed an excess piece of filament into the opening.

In fact, some black residue now flowed out of the bottom; this residue was what had been clogging the nozzle. After a few minutes of squeezing and pressing, the path seemed to be clear. Full of anticipation, I loaded a test print from the SD card and started the printing process – only to find that the filament was still not sticking to the bed. The calibration screws of the Ender 5 Pro had become misaligned again with all of my tinkering, and I had to repeat the tedious manual calibration process.

Five test prints, 30 minutes, and a few dozen bouts of hair tugging later, the filament was sticking as intended. But I already had an idea for improving the calibration step. In fact, a calibration sensor is available for those who are willing to put in the manual work – which should safely include any 3D printing enthusiast. The BLTouch [3] automatically levels the printing bed.

I immediately added the BLTouch to my list of potential upgrades. Another thing that was a pity: The Ender 5 Pro cannot connect to a host via WLAN or Ethernet out of the box. Printing is only possible via a USB cable or an SD card. Since I wanted to put the printer in a broom closet or a little-used room, I started looking for a solution.

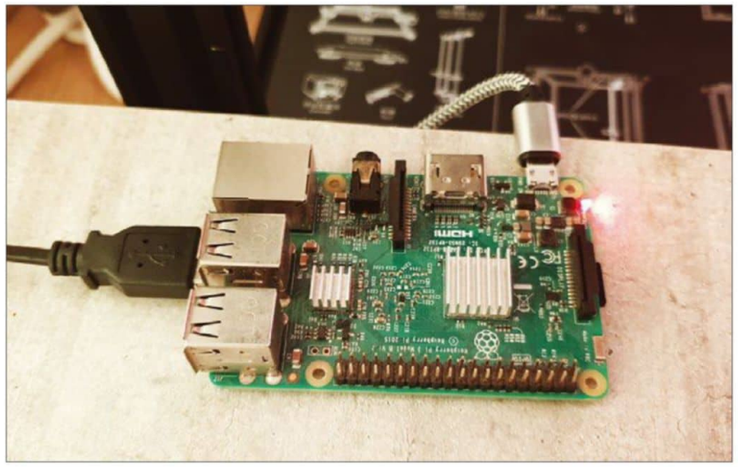


Figure 4: The OctoPrint server runs on an old Raspberry Pi 3. © Oliver Nickel

Connectivity Thanks to Rasp Pi

I was not alone with my wish. I soon discovered that there is an operating system for the Raspberry Pi that offers network connectivity for 3D printers: OctoPi. This operating system and an application package, based on Raspbian, make the 3D printer connected to the Raspberry Pi available via a web interface. You could also use OctoPi to control a webcam that monitors the printing process and makes the system visible outside the local network.

But that was not my intention at first. Instead, I wanted to send jobs from my desktop PC to the 3D printer, which is now in a quiet place where it doesn't bother me. I found a Raspberry Pi 3 model B, which had been lying around unused in a drawer for some time, and connected it to the printer. Installing OctoPi and the OctoPrint user interface onto the printer was easy with the Etcher SD flasher app and a step-by-step wizard. The tool copied all the required files to an SD card, which I then slotted into the Rasp Pi (Figure 4).

Then I logged onto the device locally and connected the Rasp Pi to the local WLAN. I did this in the Raspbian configuration menu, which can be called in the shell. Alternatively, a text file in the operating system image can be customized. Instead of logging on locally with a monitor and keyboard, you can also use an SSH connection, provided you have entered the WLAN information correctly in the text file.

After successfully completing the setup, I was able to login to my new OctoPrint print server by entering the local IP address in my web browser, and from there I was able to set up my printer. By the way, the Ender 5 Pro was immediately detected after I connected it to the Rasp Pi. I took the print space, the nozzle width, and other parameters from the printer's online data sheet.

The web interface displayed information about my printer, including how hot the bed and extruder were at the time (Figure 5). From an OctoPrint plugin repository [4], you can download and install various add-ons. For example, you could add a slicer into the web interface (see the box entitled "What Is a Slicer?"). However, the slicer available through the repository is an old and no longer supported version of Ultimaker Cura, a free and actually very good 3D printing program available for

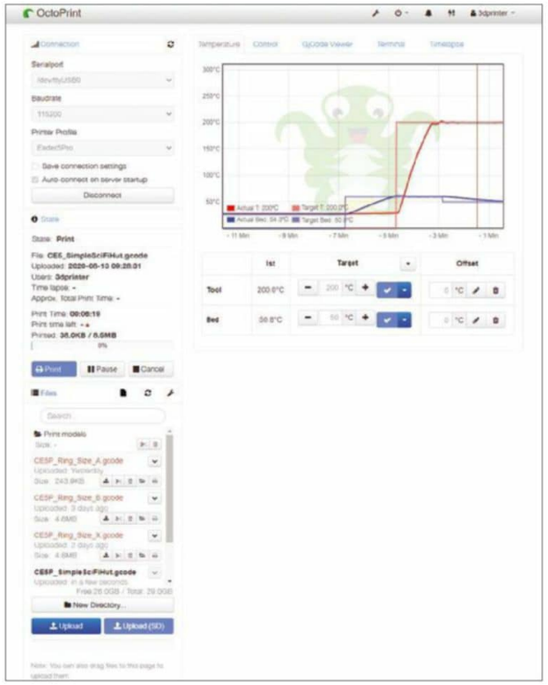


Figure 5: The web interface shows data for the printer – the temperature in this case. © Oliver Nickel

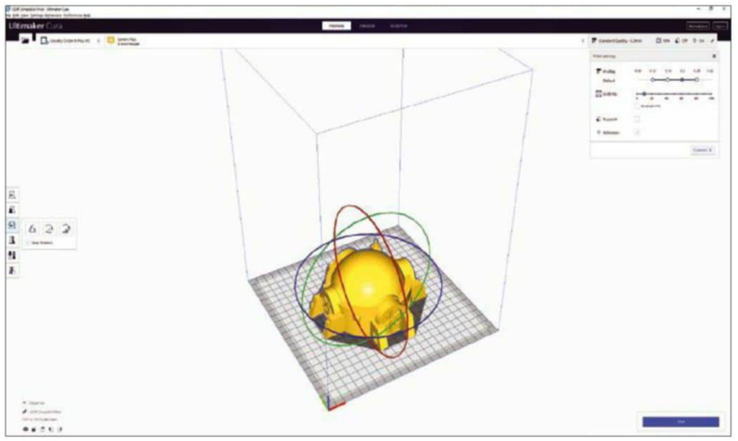


Figure 6: The Ultimaker Cura serves as a slicer. © Oliver Nickel

What Is a Slicer?

3D printers add print material in layers. A *slicer* is a program that converts a three-dimensional graphic image into instructions for a 3D printer. In other words, the software "slices" the image into layers and sends the layer-by-layer information to the printer.

Linux, Windows, and macOS. I decided to download the application directly to my desktop computer.

Ultimaker Cura Provides a Remedy

Because the Ender 5 Pro uses a modification of the Marlin firmware, it can only print files in G-code format. This code contains coordinates and information about different layers of a 3D model. The printer needs this information to know where the printhead has to apply new filament.

I downloaded the latest version of Ultimaker Cura. Cura offers a slicer that converts STL files, probably the most common format for 3D printing, into G-code and divides the model into different layers (Figure 6).

However, in order for Cura to find an OctoPrint-enabled 3D printer on a TCP port and transfer G-code files over the local network, it first needs another add-on, which you can obtain directly from the Cura marketplace. OctoPrint Connection [5] allows a previously created printer profile to be connected to the print server in the software settings.

Cura also offers a prebuilt Ender 5 profile, which sets up the printing chamber and extruder parameters as per the Pro version in advance. This saves annoying manual typing of the printer data. I took a chance and bought the Cura profile, because I had already had enough trouble in the previous steps.

Cura was able to transfer a test model at the first attempt, and my printer started to output a terrain piece for tabletop games. Now all I had to do was wait for nine hours – a complete success.

Conclusions

I have approached 3D printing as a hobby in a very euphoric way. Some tests and the many great models that are exhibited on platforms like Reddit had awakened my desire to get started. However, I quickly realized that 3D printers are also a lot of work.

It's little wonder that the hype surrounding this hobby that started in the early 2010s has quickly become a niche market again. The fiddling with things that should be obvious, and should not take too much time, adds a large dollop of frustration to the mix. In the case of the Ender 5 Pro, I noticed this especially with its printing bed that requires manual calibration. And I would not have imagined beforehand that the extruder of a completely new printer might be clogged (Figure 7).

On the other hand, 3D printing as a hobby offers massive opportunities to develop new skills. I am now into 3D modeling, and my adventures with 3D printing have given my dusty Raspberry Pi a reason to exist again. Trying out and installing components and software is great fun, especially when the results finally pay off.

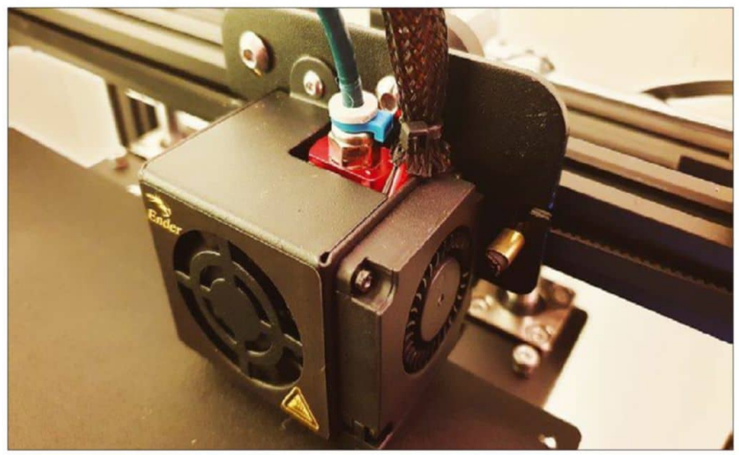


Figure 7: The Ender 5 Pro printhead was initially clogged. © Oliver Nickel

The 3D printing community is very vibrant. On many forums, you are likely to find quick help for your problems. I am looking forward to converting my printer, which seemed improvised and crude when I first bolted it together, into a useful tool for my handicrafting endeavors. ■■■

Info

[1] Ender 5 Pro: *https://www.creality3dofficial.com/products/ender-5-pro-3d-printer*
[2] Marlin firmware: *https://marlinfw.org/*
[3] BLTouch: *https://www.antclabs.com/bltouch-v3*
[4] OctoPrint Plugin Repository: *https://plugins.octoprint.org/*
[5] OctoPrint Connection: *https://github.com/fieldOfView/Cura-OctoPrintPlugin*

COVER STORY
3D Printing: Idea to Object

Going from a sketch to a printed object

Emergence

How do you get from an idea to a finished printed object? We'll take you through the steps with a glamorous example: a pair of 3D-printed earrings. *By Thomas Kaffka*

Every 3D-printed project begins with an idea. Where does the idea come from? The simplest solution is to use a model that already exists. Several Internet platforms collect finished 3D models, including MyMiniFactory [1], Thingiverse [2], or even a special NASA [3] website. From these sites, you can download finished 3D models for printing, and in some cases, you can even publish your own 3D models for the community.

Another way to create a 3D model is to scan or photograph real-world objects and then use software, such as a photogrammetry application, to generate a model. For instance, you could photograph an object from different perspectives and from all sides; when you have created 50 or more photos, you can then convert the photos into a 3D model using specialized software. See the 3Dnatives site [4] for an overview of possible software tools, including some open source programs.

A third approach is to design a 3D model directly and then print the design. This article demonstrates this approach with a pair of earrings. Their shape was first sketched the old-fashioned way using a pencil and paper (Figure 1). You can see two rectangles pushed into each other. The next step is to transfer this project idea to a 3D model.

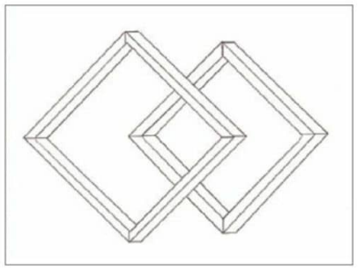


Figure 1: Manual sketch of the planned earring.

Create a 3D Model

Linux supports a number of free programs that can assist with developing a 3D model. For instance, Blender [5] is a good choice that was actually developed to create 3D scenes, render videos, and develop 3D models for computer games. Blender shows its strengths when it comes to organic forms.

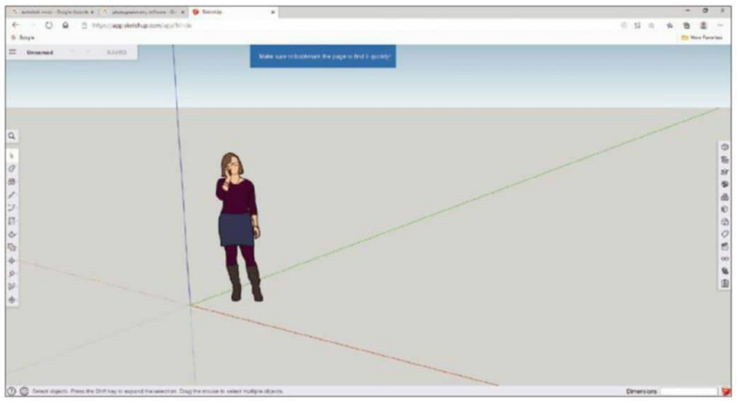


Figure 2: The SketchUp Free design page.

But since the earring sketched in Figure 1 is more of a geometric figure, a simpler tool such as SketchUp Free might be more suitable [6]. The browser-based SketchUp is often used in architecture, for example.

From the SketchUp home page, click on *Try SketchUp*, select the category *Personal* and, if you haven't already done so, create an account. After that, fill out the form, and you are taken to the page shown in Figure 2.

You will probably not want to print the lady shown in the workspace, so click on the figure and press Delete. Then use the menu in the top left corner below *App Settings* to specify the units for the project; in my case, I will use millimeters (*Regional/Simple Template – Millimeters*).

The toolbar on the left side of the screen contains some tools that help you with the construction work. To get the basic shape of the earring (you only need to design one and then print two objects based on the 3D model), click on the rectangle symbol and select *Rectangle* from the pop-up menu.

Now you can draw a square on the base area. You can specify the desired dimensions of 40x40mm, and the software then adjusts the rectangle accordingly. In the next step, use the *Push/Pull* tool to raise an area that is 5mm tall out of the rectangle, creating a cuboid (Figure 3).

To create the first shape of the earring, use the ruler tool to create measurement points at the edges of the box 5mm from the corners. Connect the lines using the *Lines* tool. A smaller rectangle is created on the surface of the box. The *Eraser* deletes lines that protrude at its corners. The results are shown in Figure 4.

To obtain the basic shape of the earring, lower the area of the enclosed, smaller rectangle and remove it from the 3D model. This is where the already-familiar *Push/Pull* tool comes into play. The inner rectangle is pressed in a little bit; you then enter *5mm*, and it disappears. Using the principle just described, now create another figure with the external dimensions of 25x25mm. If you now draw a frame around this design with the pointer tool, the software displays all areas of the small figure as dots. The *Move* tool then moves the small figure to the large one. The result is shown in Figure 5.

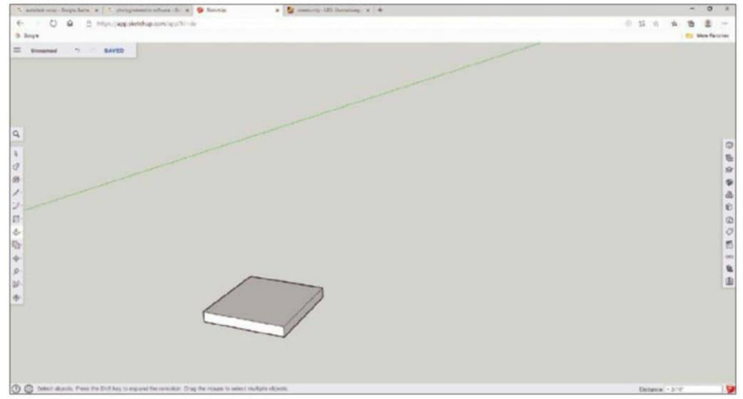


Figure 3: Lifting an area turns the rectangle into a cuboid.

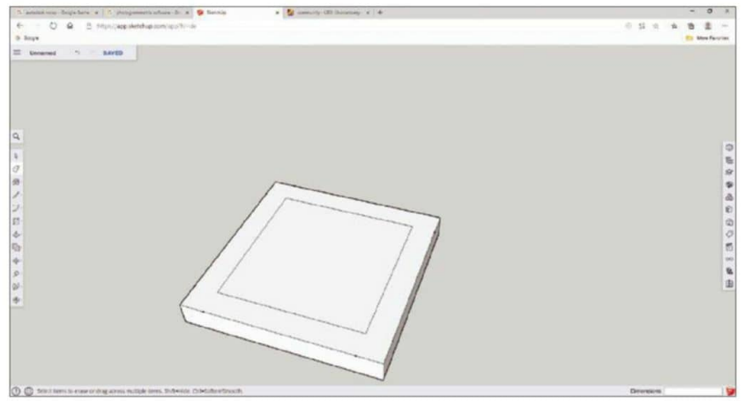


Figure 4: Drawing a helper rectangle on the box.

The last thing to do now is to integrate the eyelet into the earring using the *Rectangle/Circle* tool. First you need a center for the circle. Draw two measurement points on the upper corner of the earring using the ruler tool; each of the points needs to be 2.5mm from the edge. Now you can use the *Circle* tool to draw a circle with a radius of 2mm on the resulting center point in the corner of the design. Afterwards, delete the guides again with the *Eraser*. The *Push/Pull* tool then lowers the eyelet as before. This completes the 3D model of the earring (Figure 6).

Next, save the 3D model by clicking on *Save* at the top and enter *Earring* as the name. Finally, export the 3D model and select STL as the file format. You can then download the `earring.stl` file. The 3D model is now available in a form that you can use for further work. Exit the program via *Home* | *Account* | *Logout*.

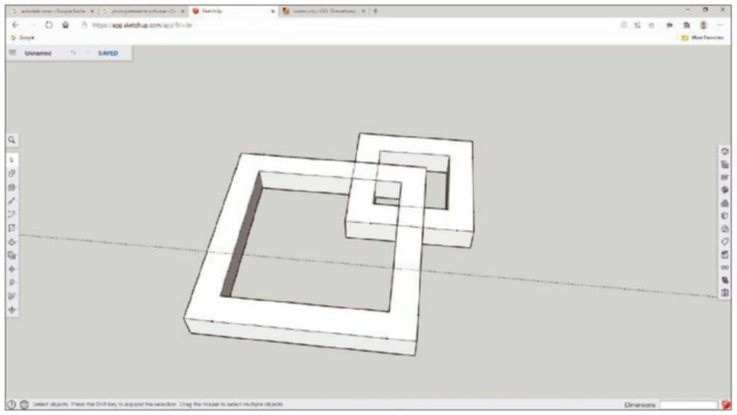


Figure 5: The smaller figure is pushed into the larger figure.

Slicing the 3D Model

The next step is to generate a file that allows a 3D printer to print the model.

In practice, a quasi-standard has emerged in the form of Ultimaker Cura [7]. Cura is available in versions for Linux, macOS, and Windows. Alternatively, you could use the slicer program that came with your 3D printer. For example, I own a Renkforce RF1000, which includes the Renkforce Repetier-Host slicer program.

After starting Cura, the first step is to configure the printer you are using. Press the second button in the menubar. After you click on the combobox and then on *Add Printer*, a window appears in which you can select your own printer.

If the model you use does not appear, select a similar one. Since I am printing the earrings on an RF1000 that Cura does not support, I chose the Ultimaker 2+. In the combobox, you have to set the printing material and maybe the diameter of the printing nozzle. In this example, the earrings will to be printed with Polyactic Acid (PLA) filament.

Now you have to make some settings for the profile in the right window. A click on the menu item *Show all settings* lists all print parameters. Since 3D printers differ greatly, Table 1 shows a few parameters that apply to the RF1000 3D printer used in this article.

Which filament to choose depends on the model and the 3D printer. For printers without headed print beds, only PLA is suitable. For printers with heated print bed, you can choose between different plastics. The height of the individual layers depends on how detailed you want the print results to be. If you select a layer thickness of 0.1mm, you'll create a finer structure.

An infill is the inner structure of the 3D object. Usually, a box structure is printed with a density between 15 and 20 percent. A loose structure inside the object is particularly suitable, because it saves material and makes the model less heavy.

You'll need support structures, or support material, if the 3D printer would otherwise print parts of the object in the air. For this reason, a slicer program automatically generates support structures where they are needed; once the object is complete, you can remove the support structure by pressing it out with your fingers or a modeling tool. (The earrings presented in this article do not have any parts that need to be supported by supporting structures.)

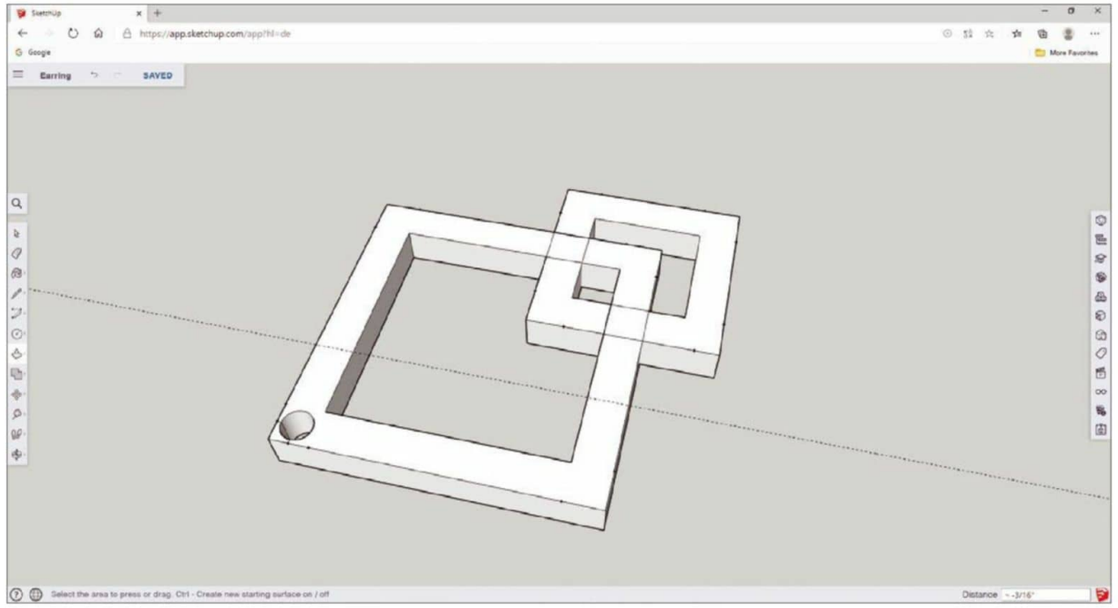


Figure 6: Lowering and removing the circle.

The parameters *Print Speed* and *Travel Speed* (of the printhead) will also depend on the 3D printer. You must specify the diameter of the filament separately. The temperature of the extruder depends on the filament. The recommended temperature of the extruder (printhead) is usually stated on the filament packaging. If the 3D printer has a heated print bed, you can adjust its temperature.

If you now load the 3D model you created with SketchUp into the Cura program, you will not see anything at first. The metric specifications of the SketchUp Free and Cura programs do not match. Therefore you have to use the middle mouse button to zoom in on the 3D model, click on it, and then enlarge it. Use the arrows to move the model on the virtual printing bed.

Some tools appear on the left side of the window. If you select *Scale* and enter 2,000 percent as a factor, the earring will appear on the printing bed. Afterwards, move the earring to the side with the *Move* tool and load the 3D model a second time – because you want to create two earrings. Again, this model is scaled to 2,000 percent (Figure 7).

Table 1: RF1000 Print Parameters

Parameter	Value
Filament	*PLA*
Layer Height	*0.2mm*
Infill Density	*15%*
Generate Support	disabled
Print Speed	*50mm/s*
Travel Speed	*130mm/s*
Diameter	*2.85mm*
Printing Temperature	*230°C*
Build Plate Temperature	*60°C*

In a window in the lower right corner, you can see the expected printing time, as well as the required amount and length of the PLA strand. *Save to File* saves the processing status to the G-code file `UM2_earring.gcode`, which is used for printing later on.

Click on *Preview* to see how the 3D printer will print the objects. You can see a collar surrounding the model. The collar is intended to provide more stability during printing and is removed from the objects afterwards. There is also a slider in the middle of the window that you can move to the left. Then you can see how the printhead moves over the objects and slowly creates or removes them.

Before you can start printing, you need to prepare the 3D printer. First, save the G-code file on the SD card for your 3D printer and insert it into the 3D printer.

It is better to choose a 3D printer that has an SD card reader. Otherwise, you have to control the 3D printer from your computer via a USB cable, which has several disadvantages: First, you cannot have full control of the computer during printing, and second, a program abort of the control program would spoil the print. Large, complex objects often require 6 to 10 hours of printing time.

Next, insert a suitable filament spool. It is a good idea to remove the used filament spool from the 3D printer after printing, as the plastic absorbs humidity over time. The material feeder can then no longer transport it correctly. After inserting the filament, let a little extrusion take place so that it is sitting flush in the extruder nozzle.

One useful trick is to paint the printing bed with a Pritt glue pen, even if you have a heatable printing bed. The glue stick has proved its value to this author; after printing, you can wash the glue off easily with a sponge and some cleaning liquid.

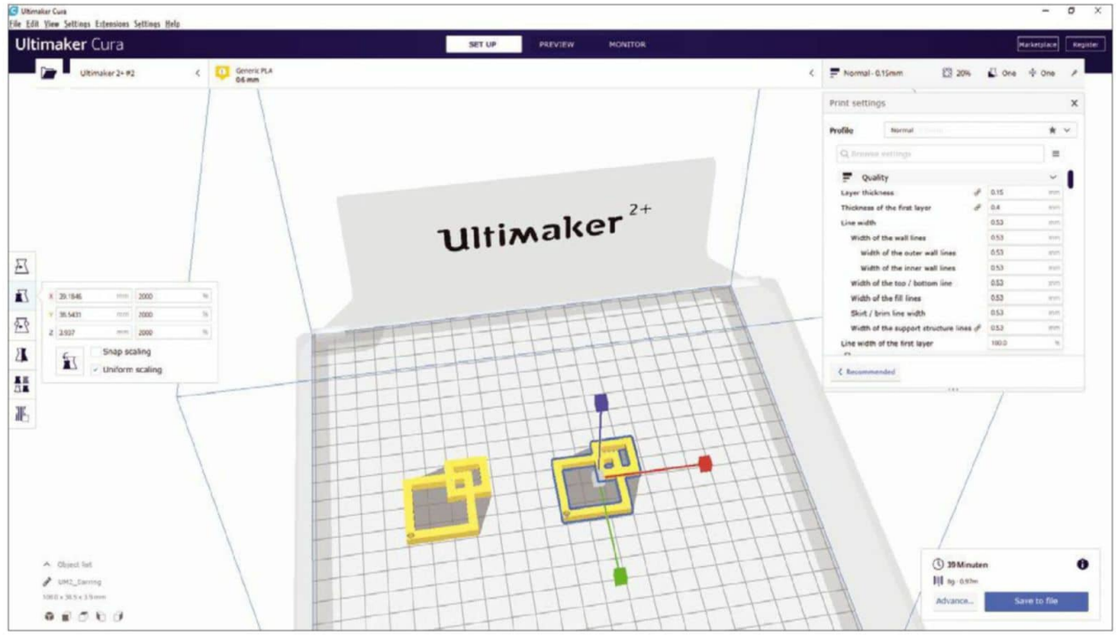


Figure 7: Both earrings are scaled and placed on the Cura virtual printing bed.

Before printing, you'll need to condition the printing bed and the pressure nozzle to the target temperature; wait at least 10 minutes until both are fully heated. A heating coil runs through the print bed. The packaging of the PLA filament used in this example gives 60°C as the printing bed temperature and 230°C as the temperature for the printing nozzle.

Now it's finally time for the actual printing, where the 3D printer automatically unwinds all steps. Those who want to can watch the not-really-exciting process. You should not leave the house in the meantime. Sometimes the object detaches from the printing bed during printing, which requires an immediate stop to prevent possible hardware defects. Keep an eye on the process and check back regularly. While printing, you can see the inside box pattern of the infill created by the slicer program. The blue color in Figure 8 is due to the fact that the interior of my 3D printer is illuminated by two blue LED light strips.

When printing is complete, you can remove the object from the 3D printer. Heatable printing beds should be allowed to cool down a little at first. The plastic used and the printing bed have different expansion coefficients, so that the object can be easily removed after printing. If it still sticks to the printing bed, a razor blade will help.

You can expect some rework after printing. The first thing to do is to remove the printed collar. A modeling tool has proven to be a good choice for difficult areas. The sharp edges that remain on the objects after removing the collars can be smoothed with a disposable file from the drugstore. Be sure you dispose of all plastic leftovers responsibly.

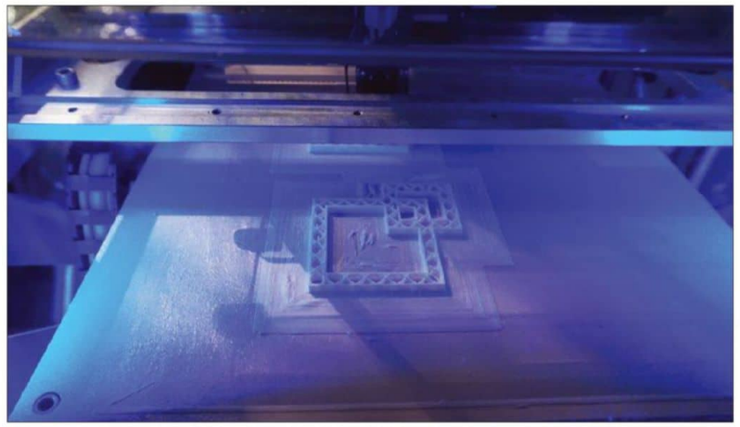


Figure 8: Clearly visible: the box pattern for filling the object.

Figure 9: The finished earrings are ready to wear.

The budding jewelry designer can also use color to further enhance the earrings. Golden spray paint from a well-stocked hardware store is a good choice. Last but not least, install a ring or hook on each earring so that you can actually attach the earrings to ears (Figure 9).

Conclusions

This small example shows the potential of a 3D printer. Once you become accustomed to the equipment and software tools, your options are nearly unlimited, and the results are often impressive. 3D printing is a hobby that continually rewards the user with a feeling of success. ■■■

Info

[1] MyMiniFactory: *https://www.myminifactory.com*
[2] Thingiverse: *https://www.thingiverse.com/*
[3] NASA 3D models: *https://nasa3d.arc.nasa.gov/models/printable*
[4] Photogrammetry software: *https://www.3dnatives.com/en/photogrammetry-software-190920194/*
[5] Blender: *https://www.blender.org/download*
[6] SketchUp Free: *https://www.sketchup.com/plans-and-pricing/sketchup-free*
[7] Ultimaker Cura: *https://ultimaker.com/software/ultimaker-cura*

Alternative Linux desktop environments

Go Your Own Way

If you are looking for an alternative desktop interface, Bruce gives a rundown of seven Linux distros with unique desktops worth exploring. *By Bruce Byfield*

Two decades ago, Linux desktops were limited to Gnome and KDE, with Xfce a distant third. All three are thriving today and have been joined by Linux Mint's MATE and Cinnamon, as well as LXDE. Together, these six provide the interface for the majority of Linux distros. Many distros support more than one of them as well.

However, not every distribution is content with the Big Six. Under the interface, these distributions may share the same applications and technologies, but they also want something more – minimalism, speed, or aesthetics. Here are seven desktop distributions that have chosen to go their own route that are worth exploring if you are looking for something different. Some of these distros' desktops have found their way into another distribution, but it is always worth seeing them in their original settings, the way they were intended to be seen.

Figure 1: With Moksha, the once-popular Enlightenment window manager became a desktop environment at last.

Bodhi

Around the turn of the millennium, Enlightenment was a popular interface somewhere between a window manager and a desktop. In 2015, Bodhi Linux [1] took Enlightenment 17, removed half-finished and broken code, and applied bug fixes from Enlightenment 18 and 19 to produce Moksha (Figure 1), which it has offered ever since.

The result is one of the most lightweight, minimalist desktops available – and, according to my observations, the fastest. Not only is the desktop no more than a dock, but fewer than a dozen utilities and apps are installed, leaving users to add whatever else they want. This approach not only makes Moksha ideal for older systems, but it is in keeping with the basic security principle to install only what is necessary. From Bodhi, Moksha spread to most major distributions.

Deepin

Deepin [2] is a distribution based on Debian's Stable repository. The Deepin Desktop Environment (DDE) is written in Qt (Figure 2), the same framework as KDE's Plasma desktop. Its window manager, dde-kwin, is a modified version of KDE's KWin, which makes for a fast and flexible desktop

DDE is a generic desktop in its layout, with a fully-populated bottom panel. Its customization features, however, almost rival Plasma's in many ways with a well laid out organization that makes it exceptionally similar to use. In addition, DDE has many custom-built components. Its installer is often praised for its simplicity, and tools like Deepin Repair give the distro a usability that is unusual even today. With these advantages, deepin is suited to regular users, especially those who appreciate Debian technology but are looking for something easier to use.

Photo by Caleb Jones on Unsplash

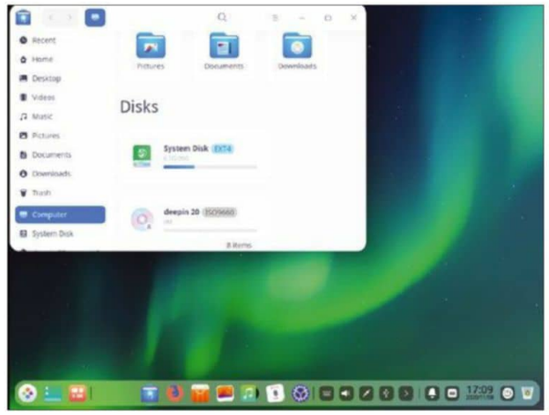


Figure 2: Written in Qt, DDE is fast and highly customizable.

elementary OS

First released in 2011, elementary OS [3] has always emphasized aesthetics in its desktop Pantheon (Figure 3). Pantheon's three main design guidelines are minimal redundancy, accessible configuration, and minimal documentation [4]. The result is an interface that is often compared favorably to Apple, although advanced users sometimes complain about a lack of choice in customization.

Pantheon is built on top of Gnome. However, over several releases, much of Pantheon has been rebuilt from scratch, including its email client, music player, panel, and application launcher. Probably its best known application is Plank, a Linux dock that is the inspiration for Planky in other distributions. The first icon on the left of Plank is always a multitasking overview. Despite all these innovations, the overall look and feel remains Gnome-like.

Pantheon is best-suited for new users, although users of all experience levels have praised it.

Manjaro

Although Manjaro [5] primarily uses Xfce and KDE, in 2019 the distribution introduced Just Another Desktop Environment (JADE, Figure 4). Originally written as a learning exercise in Python, it is a minimalist desktop, with many features hidden until required. It opens, for instance, on a few basic items like Help, Settings, and Exit arranged around a sticky-note utility. The main menu opens horizontally rather than vertically, giving plenty of room for details about each item. Each top-level item is identified by its own wallpaper.

The desktop also features audio notifications, as well as a written list of recent ones. Another small innovation is a list of recent events, a kind of undo feature for the entire desktop that seems such an obvious feature that you might wonder why it did not appear years ago.

JADE is only starting to mature, but those in search of something new may appreciate its fresh take on the desktop.

Solus/Ubuntu Budgie

Solus's [6] desktop environment is called Budgie (Figure 5). Budgie received a boost when it became an official Ubuntu flavor; it is now widely available in major distributions, including Ubuntu Budgie. Like many alternative desktops, Solus's goal is a clean, minimalist look, and it is perhaps second only to Moksha for speed.

Budgie's popularity has led to the creation of a small ecosystem of applets, including ones for countdowns, external drives, hot corners, separate keyboard bindings for each app, and menus that place favorite items at the top. By choosing which of these applets to work with, users can customize the basic desktop environment according to their own preferences, often using choices that are unavailable elsewhere.

Sugar Labs

Sugar Labs [7] contributed the desktop for One Laptop Per Child Project, a popular cause in the early years of the century.

Figure 3: Pantheon is a desktop aimed at simplicity and aesthetics.

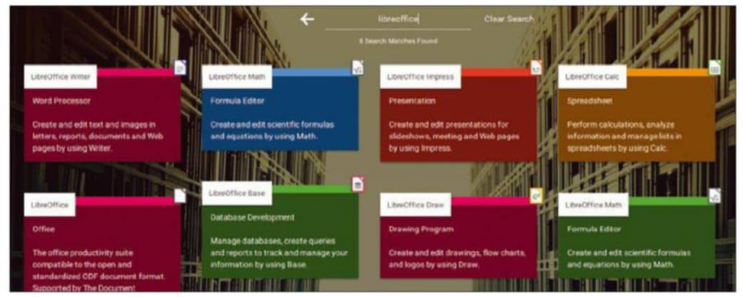


Figure 4: From its inception in 2019, JADE has shown that something new is still possible in desktops.

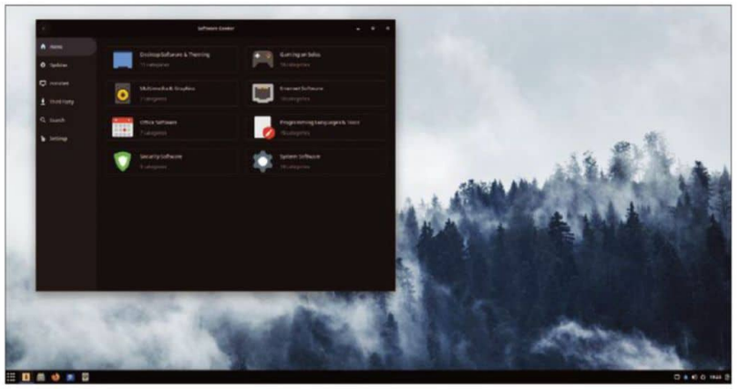


Figure 5: Budgie is a recent desktop that has spread to many distributions.

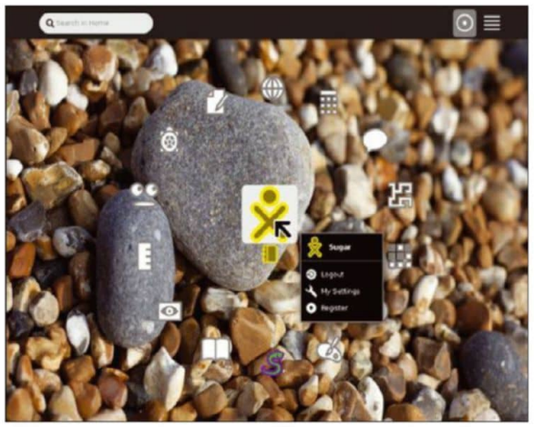


Figure 6: Sugar is a rarity: a desktop organized by task and intended for children.

Called Sugar (Figure 6), the result is still available as a desktop for children in many distributions. It is radically different from most interfaces, running full-screen and one program at a time. Moreover, it is designed to be entirely task-oriented and primarily to run its own programs – or activities, as Sugar calls them.

Sugar boots to a screen showing the current user surrounded by favorite activities. Open activities, as well as the user's journal and log, appear beneath the user's icon. Also available at boot time is a list of installed activities, with the most recently used ones at the top, accessible from an icon or via function keys. Activities are heavily documented with mouse overs or with text embedded in the activities. Available activities include Calculate, Image Viewer, Terminal, Write, Browse, and introductory programming tools.

Sugar is not for everybody, but it offers a different desktop view in an era in which innovation is frequently lacking. You can try it from a flash drive using Sugar on a Stick.

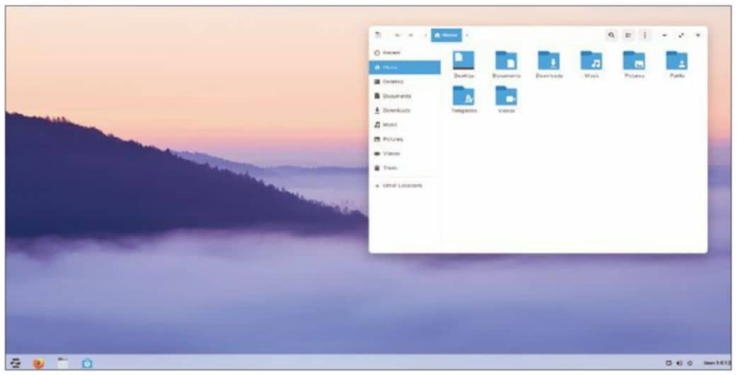


Figure 7: Zorin is full of tools not found in other distributions.

Zorin OS

Like deepin and elementary OS, one of the goals of Zorin OS [8] is aesthetics. Its desktop (Figure 7) is simple and uncluttered and aimed primarily at new users. What makes it stand out is its unusual feature set, such as the activities overview, universal search, interactive notifications, and its integration of online tools into the interface. More mundane but just as useful, Zorin OS includes a full set of accessibility settings. Its best-known feature is undoubtedly Zorin Appearance, which allows the desktop to emulate Gnome, macOS, and Windows.

Window Managers

While these seven desktop environments should be enough for anyone, you have at least four dozen alternatives if you move beyond the emphasis on distributions. Before Gnome, before KDE, Linux had other window managers – applications that set how windows were positioned, how they acted, and how they looked as well. Many window managers continue to have some popularity today, and most distributions package at least a few.

Among today's workhorse window managers are the lightweight OpenBox, dwm, Fluxbox, Window Maker, and IceWm. For those who like 3D displays, there is always Compiz. For a tiling desktop, Ratpoison remains a popular choice. How many window managers will make the transition to Wayland if it ever becomes universal is uncertain, but for the time being, there's no shortage of choices before you fall back on the best-known desktop environments or – all else failing – on the command line. ■■■

Info

[1] Bodhi Linux: *https://www.bodhilinux.com/*
[2] Deepin: *https://www.deepin.org/en/dde/*
[3] elementary OS: *https://elementary.io*
[4] elementary OS human interface guidelines: *https://elementary.io/docs/human-interface-guidelines#human-interface-guidelines*
[5] Manjaro: *https://manjaro.org*
[6] Solus: *https://getsol.us/home/*
[7] Sugar Labs: *https://www.sugarlabs.org/*
[8] Zorin OS: *https://zorinos.com/*

Secure online communication with MOFO Linux

Speak Freely

Controls, surveillance, and censorship are increasing rapidly on the Internet. MOFO Linux lets you anonymize your communication on the web with an easy-to-use live system. *By Erik Bärwaldt*

It is not only repressive political regimes, such as those in China, Iran, or Turkey, that are purposefully expanding their control and surveillance machinery on the Internet. In many Western countries, too, the Internet is increasingly becoming an object of surveillance and censorship. This is of particular concern to professional groups, such as journalists and lawyers, that need to be able to count on secure private communication to do their work.

With the help of innovative free software, however, these concerns can be readily addressed. There are many mature tools to secure free communication on Linux. However, with conventional distributions, you often have to painstakingly gather the individual tools and sometimes even install them painstakingly by hand.

Not so with MOFO Linux [1], a live system based on Ubuntu, which has been under continuous development for years. Its main focus is to ensure secure communication that is shielded from external influences.

Installation

You can pick up the hybrid ISO image, weighing in at about 2.3GB, from the project site via torrent or SourceForge. After transferring the image to an optical data carrier or USB memory stick, boot the system in live mode. Alternatively, MOFO can also be used on a virtual machine such as VirtualBox.

The system comes up without a GRUB boot menu. Instead, a login prompt appears after the boot process, but it closes automatically after a few moments. After this, an input box appears again; this also logs in the preset user *mofo* with administrative privileges without you needing to take any further action.

You are taken to an unspectacular MATE desktop with a panel at the top of the screen that lets you switch between two virtual desktops. A system tray top right in the panel provides information about the current system status. There are no icons or launchers on the desktop.

Secure Data Transmission

A glance at the menus reveals MOFO's two main focuses: On the one hand, the Ubuntu derivative contains numerous applications for securing and anonymizing Internet access using tunneled connections, while on the other it includes many programs from the realm of secure communication. The number of office or graphic applications, on the other hand, is kept within narrow limits.

The *Internet* section contains the Firefox web browser, the Thunderbird email client, and the Tor browser, which automatically connects to the network of the same name when it starts. The Tor controller independently provides convenient one-click access to the Tor network. The MOFO developers extended Firefox with add-ons like HTTPS Everywhere and the Video Downloader for downloading videos. Thunderbird comes with the Enigmail add-on preinstalled.

Numerous other applications let you set up your own VPN server and use VPN services. These include the Algo [2] VPN server manager and the Streisand [3] server. The distribution also comes with several secure messaging services, for which you only need to enter your personal access data.

Next to it, you will find the Telegram desktop short message service in the menu. The Invisible Internet Project (I2P) controller in the Internet menu lets you use the anonymized I2P network [4]. Unlike Tor, this is not a network with different nodes, but a closed overlay network that routes encrypted data traffic via the individual nodes.

Lead Image © Zentilia, 123RF.com

Central servers are missing in the I2P network. Since the data traffic runs over the network participants' I2P routers, the connection paths change permanently. In combination with end-to-end encryption, data streams in the I2P network are practically impossible to trace. The network allows access to various services that are common on the World Wide Web, but which cannot be localized due to the design of the network.

However, I2P runs with very low data throughput, which cannot be compared to today's typical bandwidths of 1MB and more. The I2P controller in MOFO prepares Firefox for the use of the I2P network and thus enables the use of the different I2P services (Figure 1).

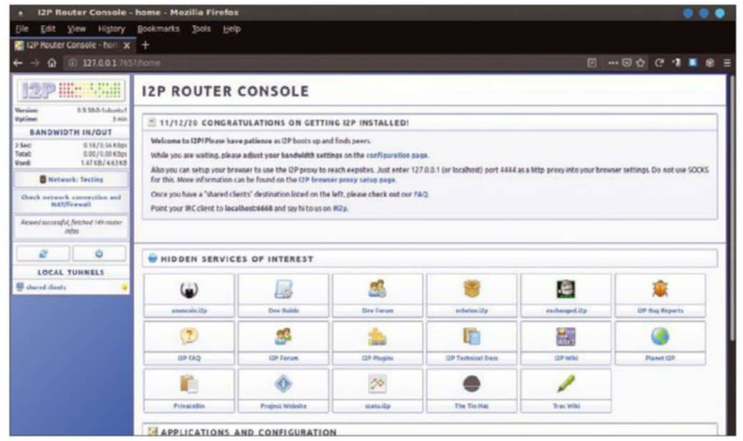


Figure 1: The I2P controller lets you access hidden services via Firefox.

Tor and Freenet

As another tool for secure data communication, MOFO also integrates OnionShare. This is a tool for transferring files over the Tor network. The contents to be sent can be dragged and dropped into the program window. When the software is called, it automatically connects to the Tor network, so it takes a few seconds for the OnionShare client to work properly.

The Freenet installer, which you will also find in the Internet menu, lets you access the Java-based installation wizard for the Freenet network [5]. This is also an overlay network for closed groups without servers or nodes that allows data to be distributed and stored redundantly. The wizard installs the Java client for use with a locally configured proxy server that supports various protocols.

The setup wizard in the browser also supports the configuration of different security levels, such as the encryption of temporarily stored data. Like on the I2P network, the data transfer rate on Freenet is significantly lower than on the open Internet. You can adjust the available bandwidth individually in the setup wizard.

In the form of Lantern, MOFO comes with another proxy service that uses peer-to-peer technologies to reach regionally blocked websites. For this purpose, the tool uses its own servers and users' computers, which help to grant access to the affected websites.

The Psiphon proxy service, which is also free, serves the same purpose and relies on peer-to-peer connections beyond central servers to bypass Internet censorship and repressive measures.

Outline Manager is an additional tool for convenient use of VPNs. This is a still quite new but free project primarily aimed at journalists and news agencies.

Outline supports the convenient installation and configuration of a VPN on local servers, but also supports the use

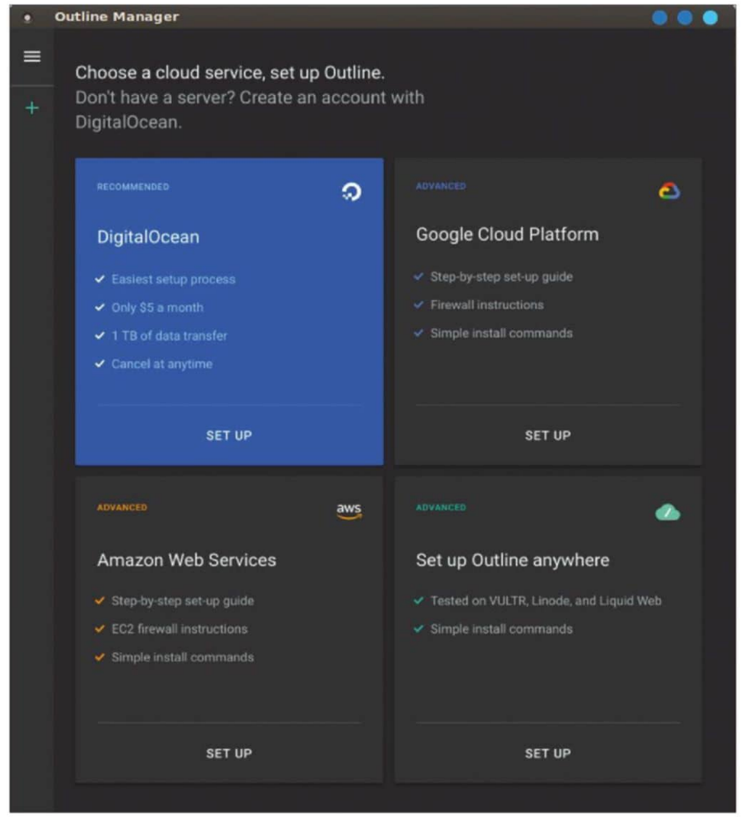


Figure 2: Outline lets you set up a VPN in no time at all.

of cloud services. Outline Manager installs the server: The project supports any number of clients. To do this, MOFO loads the corresponding AppImage off the web. You can connect the clients to the VPN via the desktop app on the workstations (Figure 2).

Communication

There are several applications for direct communication in the distribution. Besides the Signal messaging client, which supports telephony and messaging over the Internet and excels with end-to-end encryption using the free Signal protocol, Riot Messenger (recently renamed Element) is also in place. It also offers end-to-end encryption and supports various types of computer-supported communication using the Matrix protocol.

The still quite new Jami instant messenger takes a decentralized approach and does not require either a server or registration with a provider. Thanks to support for the SIP protocol, the software not only replaces messengers like WhatsApp, but it is also suitable as a VoIP telephony application à la Skype. The program works with end-to-end encryption and stores data only on the participating computers, which rules out spying. For remote desktop applications, MOFO also comes with the Remmina RDP client, which also supports VNC or NX connections.

Filesystem

Through its use of InterPlanetary File System (IPFS) [6], MOFO Linux supports an innovative peer-to-peer filesystem where the available peers receive and pass on data in a similar way to BitTorrent-based protocols. This is why DDoS attacks, which are often used to force conventional websites off the Internet do not affect IPFS connections. Distributed content delivery also saves transmission volume.

You can launch IPFS via the *IPFS Desktop* entry in the Internet menu. The routine first downloads an AppImage off the web and integrates it into the operating system. The node activity is then visualized by a blue cube symbol in the panel.

Since installing the software also sets up a corresponding Firefox add-on, the node's activities can be monitored in your browser. This is also where you can access graphics for node utilization and the approximate locations of the currently active nodes on a world map via the *Open WebUI* option (Figure 3).

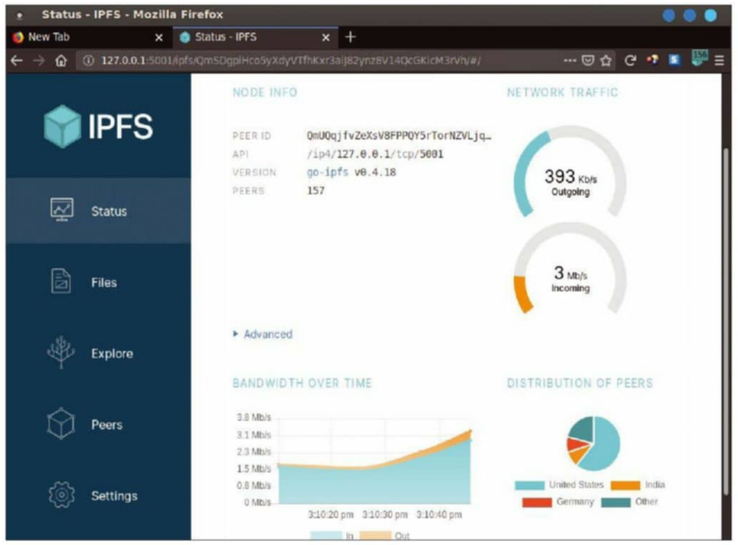


Figure 3: The IPFS peer-to-peer filesystem distributes files worldwide and thus protects them against DDoS attacks, among other things.

Other Features

MOFO offers additional applications for analyzing and configuring local data media. This includes, above all, tools for the use of data carriers and mass storage devices. In addition to GParted, which is used for partitioning mass storage devices, the graphical Disk Usage Analyzer tool provides detailed information about their allocation.

BleachBit (Figure 4), which is preinstalled in one version for users and one for administrators, helps you to safely dispose of obsolete data such as temporary files or configuration directories of deleted applications. The MATE system monitor and a logfile viewer are used to display logfiles, which in case of irregularities or problems often provide clues as to their cause.

MOFO's Accessories menu also contains a GUI tool for convenient encryption

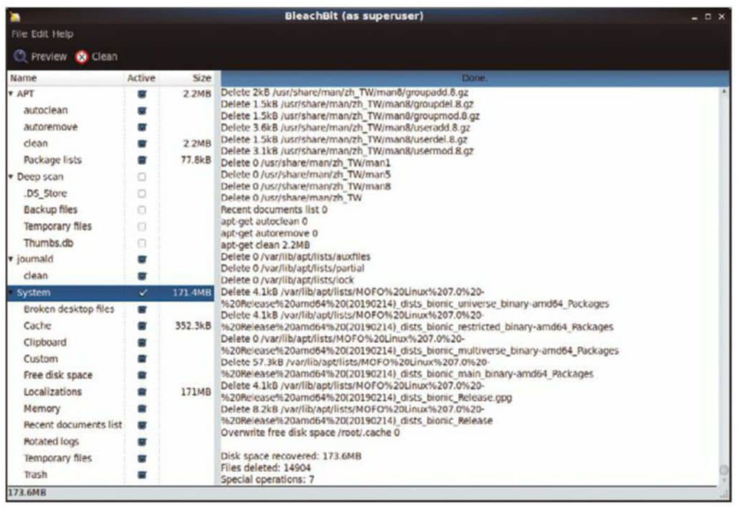


Figure 4: BleachBit clears obsolete data from mass storage devices.

of folders and files in the form of the TrueCrypt fork VeraCrypt. To help users with this, the software creates encrypted containers to which you copy the data to be backed up.

Workaround

The developers offer an elegant option to help users get around the limitations of the live system, such as the lack of persistence or the relatively slow speed.

The `sudo apt update` and `sudo apt install calamares` commands (alternatively: `sudo apt install ubiquity`) let you set up either the Calamares or the Ubiquity installation wizard on the live system. Both let you install the distribution on a mass storage device. However, doing so entails the risk of a forensic examination of the data carrier potentially revealing compromising data.

Alternatively, you can store the MOFO ISO image in a directory on an existing partition and start the image via the system's GRUB boot manager. The developers provide detailed instructions with the corresponding command syntax for the GRUB boot manager.

Conclusions

MOFO brings together numerous tools for anonymous Internet access and secure communication. For less frequently used applications, the distribution provides an installer that retroactively integrates the software into the system.

The range of tools covers just about every potential application scenario, so that you can chat, send and receive messages, and use blocked websites and services securely. This makes the live system a great choice as a secure communication platform for professional groups such as lawyers or journalists, but also offers significant benefits to the average consumer, whether traveling or at home. ■■■

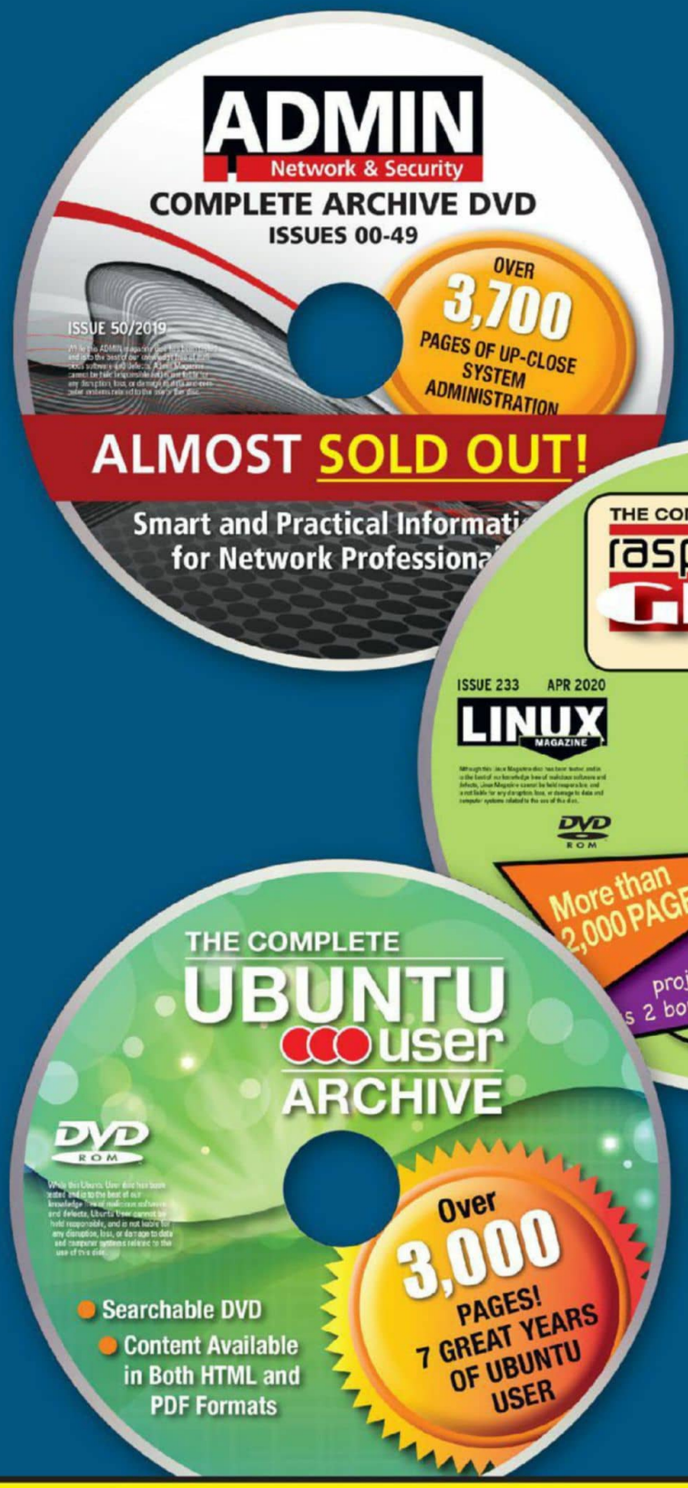
Info

[1] MOFO Linux: *https://mofolinux.com*
[2] Algo VPN: *https://github.com/trailofbits/algo*
[3] Streisand: *https://github.com/StreisandEffect/streisand*
[4] I2P: *https://geti2p.net/de/about/intro*
[5] Freenet: *https://freenetproject.org/pages/about.html*
[6] IPFS: *https://docs.ipfs.io/project/*

Complete Your Open Source Library with Archive DVDs!

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

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



ORDER YOUR DVDS NOW!

https://bit.ly/Archive-DVD

Lead Image © pixxart, 123RF.com

A command-line Twitter client

Terminal Tweeter

If you prefer to work from the command line, Rainbow Stream offers a quick and flexible Twitter client. *By Bruce Byfield*

Twitter and similar microblogging services are usually accessed from the desktop or a web browser. A rare exception is Rainbow Stream [1], which runs from the command line. Like most command-line interface (CLI) applications, it requires some learning, but once you are familiar with it, Rainbow Stream proves surprisingly quick and flexible, as well as a useful way to keep your hands on the keyboard to reduce repetitive stress injuries. With tweeting, navigation, filtering, and other functions accessible through a brief command, Rainbow Stream is a mostly complete Twitter client that is in some ways more efficient than many desktop alternatives. Overall, it is reminiscent of Alpine, the email client that was common on mainframes a couple of decades ago.

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/*.

You will not find Rainbow Stream in the repositories of most distributions. The reason seems to be because it installs using `pipenv`, Python's dependency manager. You may already have most of the required packages installed, but to confirm enter the command:

```
sudo apt-get install python-dev libjpeg ⮐
  libjpeg-dev libfreetype6 ⮐
  libfreetype6-dev zlib1g-dev ⮐
  python-pip pythonX
```

Currently, you need Python 2.7 or higher (although in the next release, the oldest supported version will be 3.6). To install, enter

```
pip install rainbowstream
```

or

```
pip3 install rainbowstream
```

depending on the Python version you are using (Figure 1).

Alternatively, you can install Rainbow Stream using `virtualenv`, as described in the documentation [2].

To use Rainbow Stream, run a Twitter account and enter the command `rainbowstream`. If you have not already authorized Rainbow Stream to use that Twitter account, Rainbow Stream will open your web browser for you to obtain an authorization number to enter at the command line (Figure 2). Strangely, in Fedora, you are given a chance to choose the Twitter account to use, while in Debian Rainbow stream chooses one of your existing accounts apparently at random.

After authorization, Rainbow Stream notifies you about any development news, the version you are using, and any errors in setup (Figure 3). Your stream is immediately available. The default theme is brightly color-coded, with the tweeter's name in orange, their Twitter handle in green, how long ago they posted in blue, and so on. Pay special attention to the ID, which you will need to interact with the tweet (Figure 4).

Basic Navigation

Rainbow Stream can quickly fill the screen. At times, you may want to press the *p* key to pause the stream if you want to focus on Rainbow Stream's help or the currently displayed tweets. When you are ready, pressing *r* will unpause the screen. You can also press *c* to clear the current display completely. Before you settle down to read, you may want to type *h* to look at the help (Figure 5). No prompt is available – just type the command in the terminal. To exit, press the *q* key.

Navigation and filtering are where Rainbow Stream excels. In Twitter's default web interface, navigating may involve scrolling, followed by selecting from the navigation pane on the left of the screen, and possibly more scrolling. By contrast, in Rainbow Stream, you can scroll using the arrow keys, but this is not the most efficient method of getting around. It is far easier to use Tab autocompletion. Better yet, you can navigate by typing brief commands and then pressing the Enter key. All commands can be entered alone, but you can ap-

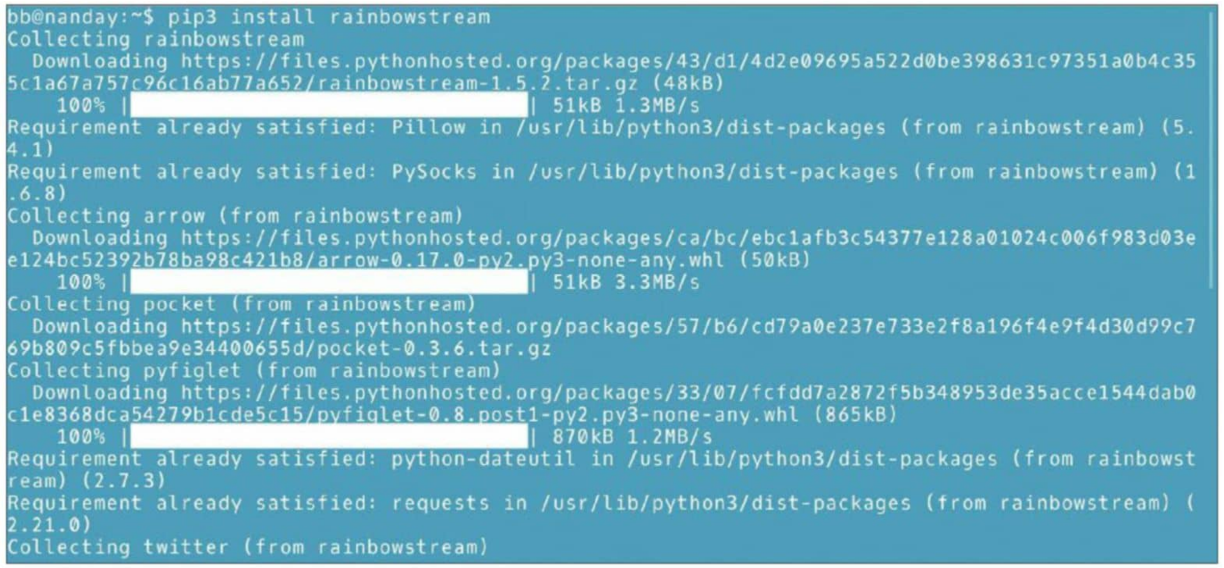

bb@nanday:~$ pip3 install rainbowstream
Collecting rainbowstream
Downloading https://files.pythonhosted.org/packages/43/d1/4d2e09695a522d0be398631c97351a0b4c355c1a67a757c96c16ab77a652/rainbowstream-1.5.2.tar.gz (48kB)
100% | 51kB 1.3MB/s
Requirement already satisfied: Pillow in /usr/lib/python3/dist-packages (from rainbowstream) (5.4.1)
Requirement already satisfied: PySocks in /usr/lib/python3/dist-packages (from rainbowstream) (1.6.8)
Collecting arrow (from rainbowstream)
Downloading https://files.pythonhosted.org/packages/ca/bc/ebc1afb3c54377e128a01024c006f983d03ee124bc52392b78ba98c421b8/arrow-0.17.0-py2.py3-none-any.whl (50kB)
100% | 51kB 3.3MB/s
Collecting pocket (from rainbowstream)
Downloading https://files.pythonhosted.org/packages/57/b6/cd79a0e237e733e2f8a196f4e9f4d30d99c769b809c5fbbea9e34400655d/pocket-0.3.6.tar.gz
Collecting pyfiglet (from rainbowstream)
Downloading https://files.pythonhosted.org/packages/33/07/fcfdd7a2872f5b348953de35acce1544dab0c1e8368dca54279b1cde5c15/pyfiglet-0.8.post1-py2.py3-none-any.whl (865kB)
100% | 870kB 1.2MB/s
Requirement already satisfied: python-dateutil in /usr/lib/python3/dist-packages (from rainbowstream) (2.7.3)
Requirement already satisfied: requests in /usr/lib/python3/dist-packages (from rainbowstream) (2.21.0)
Collecting twitter (from rainbowstream)

Figure 1: **Installing Rainbow Stream with Python's `pipenv`.**

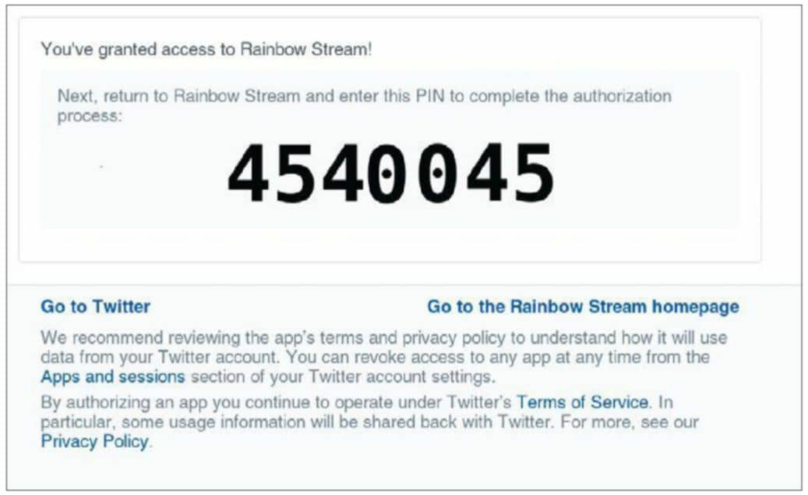

You've granted access to Rainbow Stream!
Next, return to Rainbow Stream and enter this PIN to complete the authorization process:
4540045
Go to Twitter
Go to the Rainbow Stream homepage
We recommend reviewing the app's terms and privacy policy to understand how it will use data from your Twitter account. You can revoke access to any app at any time from the Apps and sessions section of your Twitter account settings.
By authorizing an app you continue to operate under Twitter's Terms of Service. In particular, some usage information will be shared back with Twitter. For more, see our Privacy Policy.

Figure 2: **Authorization is required for encryption reasons. Copy the authorization number from the web browser and paste it into the terminal.**


bb@nanday:~$ rainbowstream
/usr/local/lib/python2.7/dist-packages/arrow/arrow.py:28: DeprecationWarning: Arrow will drop support for Python 2.7 and 3.5 in the upcoming v1.0.0 release. Please upgrade to Python 3.6+ to continue receiving updates for Arrow.
DeprecationWarning,
You are running latest version (1.5.2)
Need tips ? Type "h" and hit Enter key!
[@PrenticePieces]:

Figure 3: **Rainbow Stream notifies you of development news, the version number you are running, and any errors in setup.**

```
Prentice Pieces @PrenticePieces 3 hours ago
△:0 ♥:0 id:3 via Rainbow Stream
testing Rainbow Stream
```

Figure 4: Rainbow Stream's default theme is brightly color-coded.

pend some commands with a string of letters to narrow the topic, a specific Twitter tag to view, or a number to indicate the number of tweets to display. After being on Twitter for more than a few moments, you might want to use `me NUMBER` to reduce the amount of scrolling you have to do.

Rainbow Stream supports all of Twitter's basic features. With the command `trend`, you can view the popular tweets on Twitter, filtering the response with a single word. Alternatively, `s STRING` searches Twitter for people and topics. When you have located an account, `whois @ACCOUNT` shows the account's profile and `view @TAG NUMBER` the account's timeline. For those who are curious about their own Twitter presence, `mentions NUMBER` show tweets from others that mention you, while `notification` shows your notifications since you started Rainbow Stream. You also can refer back to your own tweets (last one first) with `me NUMBER`. And if you get lost, `home NUMBER` will return you to your timeline.

Basic Tweeting

To tweet, press *t*, followed by the text of the tweet. No quotation marks are required. Rainbow Stream will not post a duplicate tweet, nor will it tweet more than 144 characters – which is the old (not current) limit. Should you post a longer tweet, Rainbow Stream will post only 144 characters and drop the rest with no warning. The same limitations apply to any retweet or any other form of retweeting.

However, to retweet, you have to add the ID of the original tweet. For example, `rt 11` retweets the tweet in the current tweet with that ID. For your convenience, the ID is reprinted alongside your retweet. You can use `quote COMMENT ID` to retweet a message with your own mandatory comment, or you can see a list of all retweets with `rt ID`, followed by an optional number of retweets to display, or `conversation ID` to show the complete conversation in which the retweeted message appears.

Other basic commands include `fav ID` to make a tweet a favorite, or `ufav ID` to unfavorite a tweet. Similarly, with `share ID`, you can copy a link, and `image ID` uploads an image in your default image viewer. In addition, with `open ID`, you can open a tweet in your default browser – which can be a convenient alternative to searching or scrolling to find the tweet.

You can scan Direct Messages (DMs) with the `inbox` command, using `thread ID` to show a complete thread and `trash ID` to remove a message. The command `list` will show which DM lists you belong to, and `list home` will ask you for the name of a list whose timeline you wish to display. You can use `list sub` and `list unsub` to manage your subscriptions and `list add` and `list rm` to edit the people on lists that you run.

Curious about your followers? You can view a complete list with `ls fl`. Similarly, `ls fr` displays the people you are following. You can add a follower with `fl @ACCOUNT` or unfollow with `ufl @ACCOUNT`. Should someone become obnoxious, you can mute them (`mute @ACCOUNT`) or block them (`block @ACCOUNT`). Conversely, you can unmute (`unmute @ACCOUNT`) or unblock (`unblock @ACCOUNT`) them if you change your mind. Spam accounts can be reported with `report @ACCOUNT`.

None of these operations is unique. You can get the same functionality in Twitter's default web browser. However, these operations are faster in Rainbow Stream, requiring no copying and pasting or deleting with the mouse. The commands can take time to learn, but almost all are consistent

```
theme
    larapaste
    base16
    solarized
  * monokai
    tomorrow_night
```

Figure 6: Rainbow Stream offers five themes. An asterisk marks the current theme.

```
h

  Hi boss! I'm ready to serve you right now!
  ------------------------------------------------------------------------------
  You are already on your personal stream.
  Any update from Twitter will show up immediately.
  In addition, following commands are available right now:

  ♪ Twitter help
    h discover will show help for discover commands.
    h tweets will show help for tweets commands.
    h messages will show help for messages commands.
    h friends_and_followers will show help for friends and followers commands.
    h list will show help for list commands.
    h stream will show help for stream commands.

  ♪ Smart shell
    111111 * 9 / 7 or any math expression will be evaluate by Python interpreter.
    Even cal will show the calendar for current month.
```

Figure 5: Rainbow Stream's help displays a list of commands.


```
{
    // Turn to 'true' in order to disable extended tweets display (legacy mode)
    "DISABLE_EXTENDED_TWEETS" : false,
    // After 120 minutes, the stream will automatically hangup
    "HEARTBEAT_TIMEOUT" : 120,
    // Image on term
    "IMAGE_ON_TERM" : false,
    // Resize image to fit on terminal view
    "IMAGE_RESIZE_TO_FIT" : false,
    // Themes
    "THEME" : "monokai",
    // Ascii Art
    "ASCII_ART" : true,
    // Hide promt when receive a tweet
    "HIDE_PROMPT" : true,
    // Prefix
    "PREFIX" : "#owner#place#me#keyword",
    // 'search': search type ('mixed','recent','popular')
    "SEARCH_TYPE" : "mixed",
    // 'search': search max result, number over 100 will fallback to 100
    "SEARCH_MAX_RECORD" : 5,
```

Figure 7: You can customize Rainbow Stream by editing `~/.rainbow_config.json`.

in their structure, which makes them easier to learn than their sheer number might suggest.

Configuration and the Interactive Shell

Rainbow Stream installs with five multicolored themes. The command `theme` lists the theme, showing *monokai* as the current theme (Figure 6). To change the theme, enter `theme THEME`. For more detailed configuration, you can edit `~/.rainbow_config.json` (Figure 7). Most of the approximately 30 options in `.rainbow_config` can be set to *true* or *false*. Most options do things like set the number of tweets to display in a given circumstance, the order in which information for each tweet is displayed, or how long the app can be inactive before it hangs up. Other options set things such as the format for date and time. The `.rainbow_config` file overrides the app's `default.conf` file. Should the format become invalid because of edits, Rainbow Stream will still start, and you can overwrite `rainbow_config` to try adding customization again.

Since Rainbow Stream is a Python interactive shell, it can also run most other scripts designed for an interactive shell. For example, you can use it as a calculator by using `+`, `-`, `*`, and `/` for basic operands. You can also use `cal` to display a calendar for the current month (Figure 8). This functionality can sometimes save you the trouble of switching from the terminal where you are running Rainbow Stream to the desktop.

Overcoming Limitations

Besides retaining the old character limit, Rainbow Stream has at least two limitations. First, its commands do not include support for emojis. However, Rainbow Stream does support Unicode characters, which include a complete list of emojis [3]. For instance, to enter a basic smiley emoji, you would enter `U+1F600`. Given that Unicode supports 1,916 emojis, this isn't really a limitation after all.

Second, Rainbow Stream supports only one Twitter account per user. It would be inconvenient to be always deleting and replacing the authorization. However, you could install Boxer and one virtual operating system per account. This may pose a problem if you lack the disk space for virtualization.

What is not a problem – so far as I can tell – is the occasional message that Twitter is overloaded and that you should wait 15 minutes before doing anything. This warning appears to be left over from a streaming problem in earlier versions, since there is no interruption of service.

In the end, Rainbow Stream is a basic Twitter client. Unlike TweetDeck, for instance, it has no provision for scheduling tweets, although programmers might consider adding this feature. Still, its speed and efficiency might make it the Twitter client of choice for any who prefer to work from the command line. ■■■

```
cal
      October 2020
Su Mo Tu We Th Fr Sa
                1  2  3
 4  5  6  7  8  9 10
11 12 13 14 15 16 17
18 19 20 21 22 23 24
25 26 27 28 29 30 31
```

Figure 8: Rainbow Stream can run Python interactive shell scripts like `cal`.

Info

[1] Rainbow Stream: *https://github.com/orakaro/rainbowstream*

[2] Installing virtualenv: *https://rainbowstream.readthedocs.io/en/latest/*

[3] Unicode emojis: *https://unicode.org/emoji/charts/full-emoji-list.html*

Choosing the best alternative with topsis-python

Better Decisions

Complex decisions require the evaluation of multiple criteria. This article shows how to support such decisions with Linux, Python, and the TOPSIS Multi-Criteria Decision Model. *By Juan P. Tobar*

Zoning is the process by which a city's authorities define the type of land use that will be allowed in different areas of the city. Some of the zoning objectives include the promotion of tourism, employment, safety, and the general well-being of communities. As you might expect, the zoning process generates disputes due to different interests and visions.

For example, some residents might prefer to locate the industrial zone on the periphery, and others might want it in the center of the city. Which is better? The answer depends on multiple factors (Figure 1). For example, the city of Lyon, France, had its industries located in the periphery. When it suffered a strong economic collapse in the second half of the 20th century, it found itself with a deteriorated and dangerous area that needed a strong investment to be recovered. The opposite was true of Manchester, England, which had an industrial zone in the heart of the city. During the economic crisis of the 1930s, depressed industries degraded and devalued downtown properties.

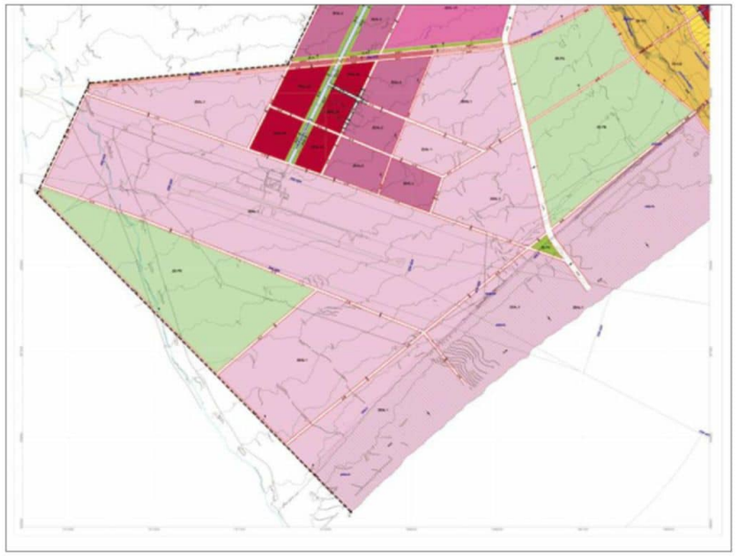


Photo by Jon Tyson on Unsplash

Figure 1: Atypical zoning proposal (Arica, Chile. Public domain).

Because zoning decisions affect cities for decades, an objective methodology is needed to evaluate both qualitative and quantitative factors. One evaluation option is Multi-Criteria Decision Models (MCDMs).

MCDMs are models that compare multiple decision alternatives, considering both qualitative and quantitative variables. These models provide an ordered list of alternatives, from the best/optimal to the least desirable.

Several MCDM models exist. In fact, planners often use more than one model and then compare the results. One example of an MCDM model that lends itself to easy computer analysis is Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS). A ready-made Python module called `topsis-python` makes it especially convenient to apply TOPSIS techniques.

TOPSIS is used every day to evaluate multi-million dollar civic improvement projects. However, if you like to code and are interested in the possibility of applying advanced analysis tools to everyday decisions, you can find a way to use TOPSIS for much simpler tasks, such as helping you determine where to buy a house or where to go on vacation. This article introduces you to the TOPSIS

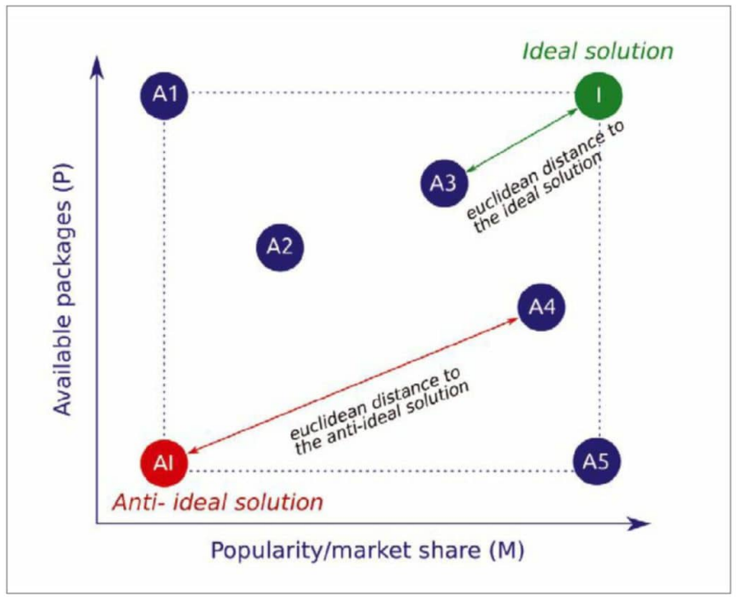


Figure 2: The TOPSIS technique.

Author

Juan Tobar has worked as an advisor to local governments and the Chilean Congress, as well as a university professor and FLOSS software developer. He is currently researching the relationship between public investment and citizen satisfaction through the use of Cellular Automata and MCDMs for Corporación de Desarrollo de Tarapacá and Consultores TyT in Chile. For more information: *juan.tobar@consultorestyt.com*.

TOPSIS Steps

Step 1: Calculate the normalized decision matrix

The first step is to bring the values of the criteria of each alternative to common units. Although TOPSIS does not require the use of a specific normalization procedure, *vector normalization* is generally used.

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{m} x_{ij}^2}}, \quad i = 1, \ldots, m, \quad j = 1, \ldots, n$$

Step 2: Calculate the weighted normalized decision matrix

In the second step, each normalized value is weighted by the importance given. This definition of weights or importance is subjective. (In real cases, these weights are defined by a commission of external experts.)

$$\sum_{j=1}^{n} w_j = 1, \quad j = 1, 2, \cdots, n$$

Step 3: Determine the ideal and anti-ideal solutions

The third step is the calculation of the ideal (*A**) and anti-ideal (*A-*) solution. For cases where the criterion is a benefit, the ideal solution is the maximum value of the normalized matrix, and the minimum value will be the anti-ideal. In cases where the criterion is a cost (or value to be minimized), the ideal solution will be the minimum value and the anti-ideal the maximum value.

$$A^* = \{v_1^*, v_2^*, \cdots, v_n^*\} = \left\{\left(\max_j v_{ij} | i \in I'\right), \left(\min_j v_{ij} | i \in I''\right)\right\}, \quad i = 1, 2, \cdots, m, \quad j = 1, \cdots, n$$

$$A^- = \{v_1^-, v_2^-, \cdots, v_n^-\} = \left\{\left(\min_j v_{ij} | i \in I'\right), \left(\max_j v_{ij} | i \in I''\right)\right\}, \quad i = 1, 2, \cdots, m, \quad j = 1, \cdots, n$$

Step 4: Calculate the separation measures

The fourth step is to calculate the distance of each alternative to the ideal solution (*Di**) and the distance of each alternative to the anti-ideal solution (*Di-*).

$$D_i^* = \sqrt{\sum_{j=1}^{n} \left(v_{ij} - v_j^*\right)^2}, \quad i = 1, 2, \cdots, m, \quad j = 1, 2, \cdots, n$$

$$D_i^- = \sqrt{\sum_{j=1}^{n} \left(v_{ij} - v_j^-\right)^2}, \quad i = 1, 2, \cdots, m, \quad j = 1, 2, \cdots, n$$

Step 5: Calculate the relative closeness to the ideal solution

The relative closeness (*Ci**) takes values between *0* and *1*. Alternatives closer to *1* are closer to the ideal solution, and alternatives closer to *0* are closer to the anti-ideal.

$$C_i^* = \frac{D_i^-}{D_i^* + D_i^-}, \quad i = 1, 2, \cdots, m$$

Step 6: Rank the alternatives

Finally, from the relative proximity, the alternatives are ranked; the most recommended is the one that is closest to the ideal solution and farthest from the anti-ideal solution.

Table 1: Alternatives and Evaluation Criteria

Alternatives	C1	C2	C3	C4
A1	3.0	100	10	7
A2	2.5	80	8	5
A3	1.8	50	20	1
A4	2.2	70	12	9
Seeking	Min	Max	Min	Max

analysis process and puts TOPSIS to work on a simple comparison problem.

About TOPSIS

The TOPSIS technique was developed by Hwang and Yoon in 1981 [1]. The goal of TOPSIS is to find the positive ideal solution (which maximizes benefits and minimizes costs) and the negative ideal solution (which minimizes benefits and maximizes costs). TOPSIS also calculates the alternative that simultaneously has the shortest distance from the positive ideal solution and the longest distance from the negative ideal solution.

To better explain the TOPSIS technique, take a look at Figure 2. Suppose you must select a Linux distribution to install on your new computer. There are five possible alternatives (*A1*, *A2*, *A3*, *A4*, and *A5*). These alternatives are evaluated with two criteria: *Popularity/market share (M)* and *Number of available packages (P)*.

You can see that the highest possible value for the criterion *Popularity/market share (M)* is in the alternative *A5*, and the best value for the criterion *Number of available packages (P)* is the alternative *A1*. Then an ideal alternative would be one that combines these two values, which I call the *Ideal solution* (*I* in Figure 2). In the same way, I will define the point with the lowest possible values of both criteria and will call it the *Anti-ideal solution* (*AI*).

Because in the example (and in most real cases) no alternative fulfills all the requirements of the ideal solution, TOPSIS tells us that the best alternative is the one that simultaneously has the shortest euclidean distance to the ideal solution and the longest to the anti-ideal solution. To reach this solution, TOPSIS makes calculations in 6 steps (see the box entitled "TOPSIS Steps") [2].

Listing 1: Inputting the Data

```
>>> from topsis import topsis
>>> a = [[3, 100, 10, 7], [2.5, 80, 8, 5], [1.8, 50, 20, 11], [2.2, 70, 12, 9]]
>>> w = [0.188, 0.266, 0.135, 0.411]
>>> I = [0, 1, 0, 1]
>>> decision = topsis(a, w, I)
```

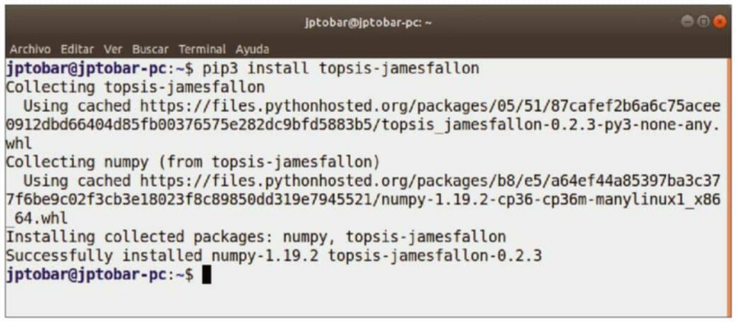


Figure 3: Installing `topsis-python`.

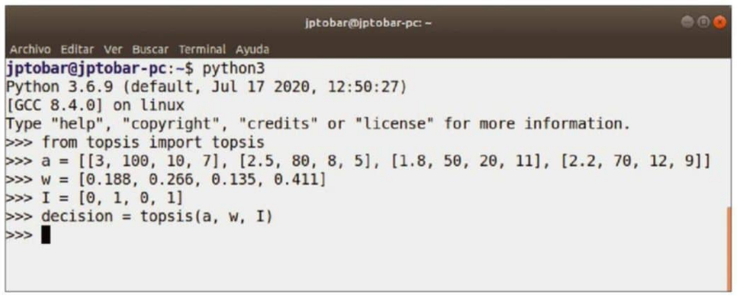


Figure 4: Use the Python interpreter to execute the steps from the command line.

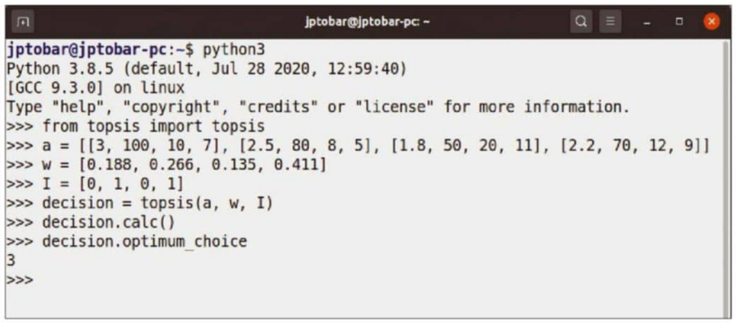


Figure 5: Finding the optimum choice.

An Example

Assume you wish to define the city's optimal zone to install industries. The possible alternatives are shown in Table 1. Table 1 also shows the criteria that will be used to evaluate the alternatives (including an indication of whether the criteria seeks to maximize or minimize). The criteria are:

- C1: Cost to relocate a company that is out of the area; minimization is sought (Min).
- C2: Expected level of employment if the zone is selected; maximization is sought (Max).
- C3: Negative externalities generated (e.g., noise); minimization is sought (Min).
- C4: Positive impact on the social development of the area; maximization is sought (Max).

The criteria are not all equally important, so I will give them different weights. In real cases, a commission of independent experts will assign the weights, but for this example, I will define the weights of the criteria as follows (on a scale from *0* to *1*, where 0 means "unimportant" and *1* means "the most important"):

```
C1: 0.188
C2: 0.266
C3: 0.135
C4: 0.411
```

To find the solution to the problem, I will use Python 3 on Ubuntu with the `topsis-python` module.

The first step is to open the command line and install the module [3] (Figure 3):

```
pip3 install topsis-jamesfallon
```

After installing the module, use the Python interpreter to enter the problem data (Listing 1). Variable "`a`" represents the alternatives/criteria values (also knows as decision matrix). Variable "`w`" stores the assigned weights of the criteria. The information on minimization/maximization is provided in "`I`" (Figure 4).

Once you have input the data, the functions of the `topsis-python` module will perform the necessary calculations. If you want to skip directly to the final answer, calling `decision.calc()` executes all the steps at once, and `decision.optimum_choice` returns the optimal alternative (Figure 5). However, if you're interested in the intermediate calculations, `topsis-python` has separate functions that execute each of the TOPSIS steps (refer to the box entitled "TOPSIS Steps"). As shown in Listing 2, you can start by using the `decision.step1` function and the `decision.r` variable to calculate the normalized decision matrix (Table 2).

Then `decision.step2` multiplies the normalized decision matrix by the weight associated with each of the criteria to determine the weighted normalized decision matrix (Listing 3, Table 3). As shown in Listing 4, the `decision.step3` function puts the ideal solution in the `decision.ab` variable and the anti-ideal solution in `decision.aw` (Table 4). In Listing 5, `decision.step4` calculates the separation measure to the ideal solution (`decision.db`) and the negative-ideal solution (`decision.dw`) – the results appear in Table 5. Use `decision.step5` and

Table 2: Normalized Decision Matrix

Alternatives	C1	C2	C3	C4
A1	0.62110337	0.51758614	0.37266202	0.4554758
A2	0.64820372	0.51856298	0.32410186	0.45374261
A3	0.37582301	0.30065841	0.75164603	0.45098762
A4	0.42135049	0.30096463	0.66212219	0.54173634

Table 3: Weighted Normalized Decision Matrix

Alternatives	C1	C2	C3	C4
A1	0.11676743	0.09730619	0.07006046	0.08562945
A2	0.17242219	0.13793775	0.0862111	0.12069553
A3	0.05073611	0.04058889	0.10147221	0.06088333
A4	0.17317505	0.12369646	0.27213222	0.22265364

Table 4: Ideal and Anti-Ideal Solutions

	C1	C2	C3	C4
Positive ideal solution	0.07006046	0.17242219	0.04058889	0.27213222
Negative ideal solution	0.11676743	0.0862111	0.10147221	0.12369646

Table 5: Separation Measures

	A1	A2	A3	A4
Distance to the ideal solution	0.10989554	0.1548053	0.10554209	0.07601339
Distance to the anti-ideal solution	0.11160034	0.08222631	0.15561078	0.11661359

Listing 2: Normalized Decision Matrix

```
>>> decision.step1
>>> decision.r
array([[0.62110337, 0.51758614, 0.37266202, 0.4554758 ],
       [0.64820372, 0.51856298, 0.32410186, 0.45374261],
       [0.37582301, 0.30065841, 0.75164603, 0.45098762],
       [0.42135049, 0.30096463, 0.66212219, 0.54173634]])
```

Listing 3: Weighted Normalized Decision Matrix

```
>>> decision.step2
>>> decision.v
array([[0.11676743, 0.09730619, 0.07006046, 0.08562945],
       [0.17242219, 0.13793775, 0.0862111 , 0.12069553],
       [0.05073611, 0.04058889, 0.10147221, 0.06088333],
       [0.17317505, 0.12369646, 0.27213222, 0.22265364]])
>>>
```

Listing 4: Ideal and Anti-Ideal Solutions

```
>>> decision.step3
>>> decision.ab
array([0.07006046, 0.17242219, 0.04058889, 0.27213222])
>>> decision.aw
array([0.11676743, 0.0862111 , 0.10147221, 0.12369646])
>>>
```

Listing 5: Separation Measures

```
>>> decision.step4
>>> decision.db
array([0.10989554, 0.1548053 , 0.10554209, 0.07601339])
>>> decision.dw
06 array([0.11160034, 0.08222631, 0.15561078, 0.11661359])
>>>
```

`decision.C` to calculate the relative closeness to the ideal solution (Listing 6, Table 6), then finally call the `decision.C` variable to get the ranking of the alternatives (Listing 7, Table 7).

According to the result in Listing 7, the optimal alternative, that is, the one that maximizes the benefits and minimizes the negative externalities, is number 4 (array position 3). In addition, the result shows that the second best alternative is number 3, then number 1, and finally number 2.

Table 6: Relative Closeness to the Ideal Solution

	A1	A2	A3	A4
Relative Closeness	0.50384838	0.3469002	0.59586089	0.60538555

Listing 6: Relative Closeness to the Ideal Solution

```
>>> decision.step5
>>> decision.C
array([0.50384838, 0.3469002 , 0.59586089, 0.60538555])
>>> decision.optimum_choice
3
>>>
```

Listing 7: Ranking the Alternatives

```
>>> decision
Best alternative
a[3]: [ 2.2 70.  12.   9. ]
>>> decision.C
array([0.50384838, 0.3469002 , 0.59586089, 0.60538555])
```

Conclusion

In real-world scenarios, the number of criteria and alternatives is higher, but this example shows how to define complex problems and obtain an optimal decision in a simple way using Python, Linux, and TOPSIS. ■■■

Table 7: Ranking the Alternatives

Alternative	Result	Rank
A4	0.60538555	1
A3	0.59586089	2
A1	0.50384838	3
A2	0.3469002	4

Info

[1] Hwang, C.L. and K. Yoon. *Multiple Attribute Decision Making: Methods and Applications*. Springer-Verlag, New York, 1981

[2] Papathanasiou, J. and N. Ploskas. "TOPSIS". In: *Multiple Criteria Decision Aid*. Springer Optimization and Its Applications, vol. 136. Springer, Cham, 2018: *https://doi.org/10.1007/978-3-319-91648-4_1*

[3] topsis-python module: *https://pypi.org/project/topsis-jamesfallon/*

The sys admin's daily grind: Stockfish

Shallow Blue

In the absence of an IBM supercomputer at his data center in Germany's Lower Rhine region, Charly has to make do with a Linux desktop, Stockfish, and chs in order to follow in the footsteps of chess grandmaster Garry Kasparov. *By Charly Kühnast*

Writer Raymond Chandler called chess "as elaborate a waste of human intelligence as you can find outside an advertising agency." But that doesn't put us off, does it?

In 1996, the media frenzy was huge when the Deep Blue chess computer developed by IBM beat the reigning world champion Garry Kasparov [1]. However, Deep Blue was also a veritable wall cabinet with power consumption to match. Today there are powerful open source chess engines for the home Linux desktop. One of the most powerful engines goes by the name of Stockfish [2]. It has been in development for more than 10 years and has now reached version 12.

However, Stockfish only provides the artificial chess intelligence. You also need a user interface (i.e., a game board). As a chess aficionado, I decided to use chs, which is written in Python. Chs lets you play against the Stockfish engine in your terminal. First, you need to install Stockfish and the Python installer pip before calling `pip` to install chs (Listing 1).

Then you need to tell chs where to find the Stockfish engine. This involves editing the `.local/lib/python3.6/site-packages/chs/engine/stockfish.py` file. The line starting with `engine_path` needs to be edited as shown in Listing 2.

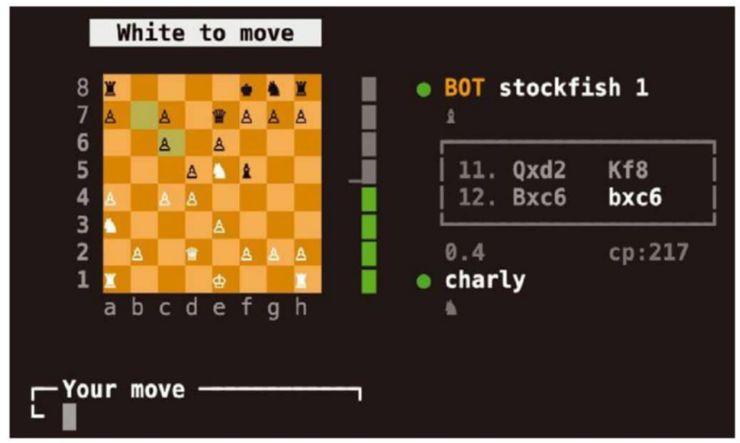


Figure 1: Chs notes all the moves in standardized PGN.

All done. I can now type `chs` to start a game. You enter the moves in a line of text. The letter indicates which figure you want to move:

- N = Knight
- R = Rook
- Q = Queen
- K = King
- B = Bishop

By the way, the word rook originally comes from Persian, where "rukh" means a chariot. There is no letter for the pawn, which is not necessary due to the limited move paths. To move the left white pawn two squares forward, simply enter `a4`. For the other chess pieces, you enter the name and the target square. "Rook to c6" is therefore `Rc6`.

Chs logs the moves in Portable Game Notation (PGN) [3], which is shown in Figure 1 to the right of the chessboard. `Bxc6` means that the white bishop has been moved to c6; the `x` indicates that it has taken an opponent's piece.

Chs doesn't have a stop clock, but no worries. The way I play follows a saying by the German writer Otto Galo. In comparing politics to chess, Galo said it's hard to make the best moves under pressure. ■■■

Listing 1: Installing chs

```
$ sudo apt install stockfish python3-pip
$ pip install chs
```

Listing 2: Editing stockfish.py

```
[...]
elif 'Linux' in platform.system():
  engine_path = '/usr/games/stockfish'
[...]
```

Info

[1] Deep Blue: *https://en.wikipedia.org/wiki/Deep_Blue_(chess_computer)*

[2] Stockfish: *https://stockfishchess.org*

[3] PGN: *https://en.wikipedia.org/wiki/Portable_Game_Notation*

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.

Hacking free software for creative writing

Creative Writing

Some tools designed for programming can also be very helpful for writing fiction. A few to look at include personal wikis, random word generators, and version control tools. *By Bruce Byfield*

One of the most important lessons I have learned from using free software is the ability to improvise. Although I am not a developer, I long ago learned to hunt for useful scripts and adapt them for my own purposes. However, it is only recently that I realized that, with a little improvisation, tools designed for programming can be made useful for writing fiction.

Some free software, of course, is already designed for use by writers. Although a sadly high number of users have yet to learn that, as Robin Williams said in the title of her book, *The PC is Not A Typewriter* [1], LibreOffice is designed for writers of long documents. Similarly, Calc, like any spreadsheet, is ideal for outlining scenes and for keeping track of more abstract elements of storytelling, such as the phases of the moon or the course of a character's illness. However, the repositories of free software also contain some less obvious tools that are useful for writers, including those detailed below.

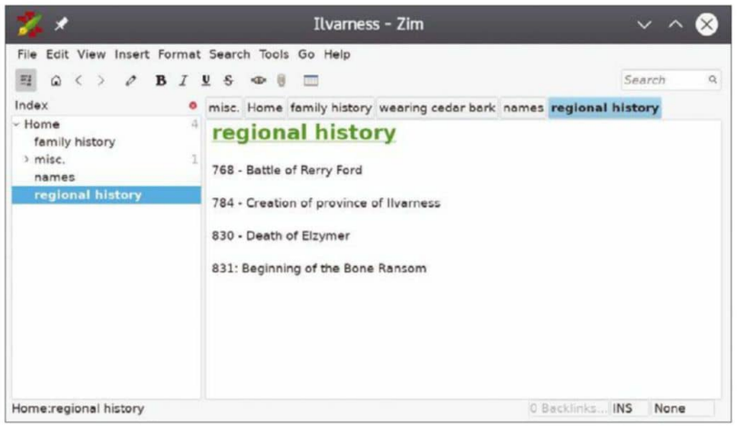


Figure 1: A personal wiki like Zim is a useful place to store background and personal notes.

Personal Wikis

Wikis are popular for free software projects. However, individuals also find them useful for many other purposes, which is why the personal wiki has become common. For writers, personal wiki files are useful because they can be linked and lightly formatted, which is a more reliable way to organize background information like the biography of characters than relying on directory and subdirectory structures. Countless personal wikis are available, but VimWiki [2] is a command-line option that is easy to learn, while Zim [3] offers the same advantage on the desktop (Figure 1).

Coining Fantasy Names

Invented names are an important part of fantasy and science fiction. Gamers often use online databases, but a writer needs something more original. But how to generate names that can be easily pronounced? One useful tool is a random word generator like xkcdpass [4].

Xkcdpass is a password generator named for the famous xkcd comic that suggests that a string of random words can be used for strong passwords. By default, xkcdpass generates five random words that can also be used for name coining. For example, say xkcdpass generates:

Photo by Lauren Mancke on Unsplash

limelight diary deepen recovery collected

From these five words, I could generate names like Diardeep, Deeplim, Colvery, and Rylight, adding more until inspiration fails and I generate a new set of random words.

Using xkcdpass's options, I could vary the number and length of the generated words. If I wanted to use a particular string – for instance, "skul" to suggest something sinister – I could choose to generate only words that contained that string. And if the words sound too English? Then with the option `--wordfile=WORDFILE`, I could use the dictionary for another language. Xkcdpass currently supports Finnish, French, German, Italian, Norwegian, Portuguese, and Spanish, and it would be relatively easy to create a new dictionary by copying and pasting other dictionary systems.

Version Control Tools

For most writers, version control is haphazard. A common joke is that a writer's files will be a collection of names like `final.odt`, `2nd-final.odt`, and `absolutely-final.odt`. Version control systems like Git [5] and Mercurial [6] have never been widely used by writers because of the difficulty of using them with office suite files.

However, with a little searching and the recent release of diffoscope, version control is now possible (more on diffoscope below). Admittedly, merging files remains mostly manual, but I find that is what I prefer anyway. The most useful feature of version control is the existence of multiple branches, so that the relation of one draft to another is easily detectable (Figure 2). Despite the fact that a command like `git` has dozens of options, a writer's needs are relatively simply, which allows the basic workflow to be learnable in less than an hour.

```
bb@nanday:~/wip$ git branch
  1st
  2nd
  Final
  master
  outtakes
* sub-plot
```

Figure 2: Git's branches are a useful way to organize different drafts of a manuscript, including experimental ones.

Binary Diffs

I covered diffoscope [7] in depth in the November 2020 issue of *Linux Magazine* [8]. Here, I will only say that diffoscope is a new command that brings the time-honored `diff` command to the desktop and works with over 60 binary formats, including LibreOffice's ODT format. Although originally written by Debian's Reproducible Builds project, diffoscope is an important building block for writers who choose to work more like developers.

At its simplest, diffoscope requires no more than the command and two file names to compare versions (Figure 3). However, if you choose, you can specify options such as the length of each excerpt from the two files and regular expressions to include or omit. You can also experiment with fuzzy logic.

```
 not interested in the prophet Lauman, let alone his belief in a
 single god, but the book was one that Talson would not read, and
 that their mother would consider a waste of time. "Trust facts,"
 her mother liked to say, rattling her bunch of keys. "They're
-what a ruler needs." Skule wanted her own keys – her own high
+what a ruler needs." Skulae wanted her own keys – her own high
 seat, too – yet she disliked her mother's hints that her father
 was short of facts. Her father was perfect.

 Her father – something about her father was keeping her awake.
 Biting her lips, she tried to concentrate on the book. It was a
 stupid book; it had more words than illuminations, and some of
 the words, she suspected, were made up by the writer. She was
@@ -477,10 +460,10 @@
 Or maybe Elzymer had died in some other way. Ragger the Horse
 Thane was leading the housecarls now, and it was whispered that
 he had become haggard with madness, and killed all the prisoners
 himself. No prisoners were left alive to question.

 Whatever happened, Elzymer's widow did not cry. Bone Ransom, was
 it?Then she owed the clans for her husband's bones, and maybe
-others besides. Meanwhile, the accounts from the Pass told
+others besides. Meanwhile, the acounts from the Pass told
 nothing that she was not already learning for herself.
```

Figure 3: Diffoscope is a new tool for comparing two different binary files.

Merging Files

One of the barriers to using `git` or `diff` commands for writing is that they usually work with text files, while most writers work in binary formats or at least have to submit their work in one. In addition, as I write, diffoscope is too new to have automated merge capacity. Consequently, most merges have to be manual, although there are several ways to go about it.

If you only want to append files, `ooo_cat` [9] will do the job. From the desktop, you can connect files to each other in LibreOffice using *File* | *Send* | *Create Master Documents* (Figure 4). This feature not only appends documents but also allows files to be rearranged and new text entered. The files in a master document remain separate but can be printed or saved as a single document.

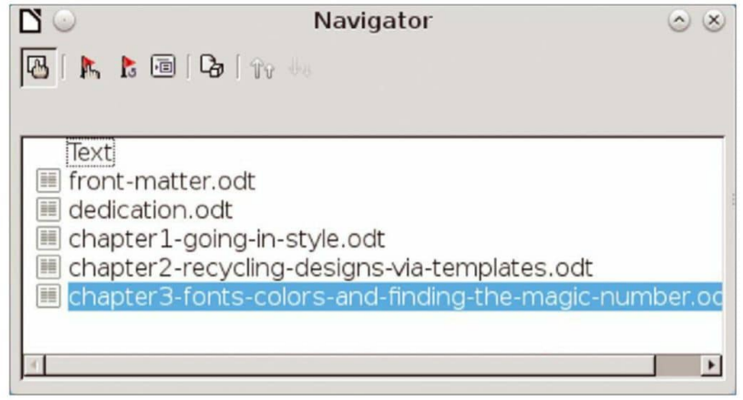


Figure 4: LibreOffice Writer's *Master Documents* feature brings multiple files together.

More complicated file merges can be done with LibreOffice's *Edit* | *Track Changes* | *Compare Documents* (Figure 5). You can open a second file so that you can approve all the changes, one at a time. Alternatively, you can use *Merge Documents* to combine both documents all at once. The *Compare Documents* option in particular has an interface complicated enough to take some getting used to, although it is a powerful solution.

You also have the option of working in plain text so you can use Git's own merge features. In order to do so, you need to convert ODT files to plain text, using a solution like odt2text [10] for LibreOffice files or PDFMiner [11] for PDF files. Unfortunately, while you can convert text back to ODT format by opening a file in LibreOffice and then saving it, much of the formatting to prepare a manuscript for submission still needs to be added – a task eased by styles, but still a tedious one. A more practical solution would be to write in LaTeX or HTML, markup languages that work with plain text and therefore allow you to work directly with Git's `merge` command. When you are finished editing, you can compile a LaTeX file or run a script to convert text to ODT or HTML to ODT.

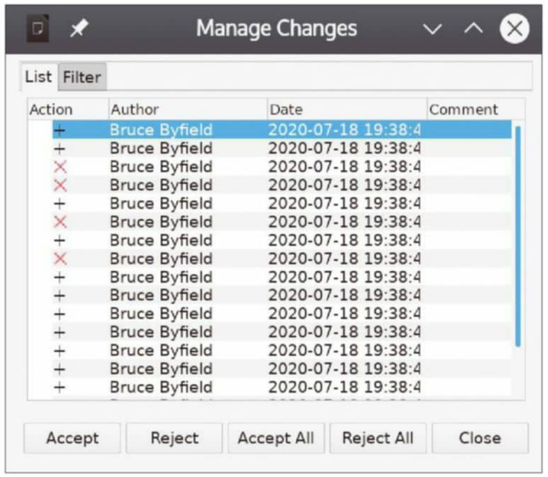


Figure 5: LibreOffice's *Compare Documents* feature is a graphical tool for choosing the changes to accept while comparing two documents.

The New Workflow

None of the tools mentioned here is promoted as being specifically for writers. However, with a little ingenuity, writers can benefit from them almost as much as developers. They require a change in workflow that takes a while to learn, but the effort is worth the struggle. In the last few months, I have found that thanks to these tools I am getting organized in my habits. I no longer have to hunt for background notes, or figure out where my drafts are, or scroll up and down as much as I once did. These development tools have helped me to organize my workflow and increase my efficiency – improvements that, like most writers, I have badly needed.

Info

[1] Williams, Robin. *The PC is Not A Typewriter*. Peachpit Press, 1992: *https://www.amazon.com/Pc-not-typewriter-Robin-Williams/dp/0938151495*

[2] VimWiki: *https://vimwiki.github.io/*

[3] Zim: *https://zim-wiki.org/*

[4] xkcdpass: *https://pypi.org/project/xkcdpass/*

[5] Git: *https://git-scm.com/*

[6] Mercurial: *https://www.mercurial-scm.org/*

[7] Diffoscope: *https://diffoscope.org/*

[8] "A modern diff utility" by Bruce Byfield, *Linux Magazine*, issue 240, November 2020, pp. 30-32

[9] ooo_cat: *http://ooopy.sourceforge.net/*

[10] odt2text: *https://medium.com/@mbrehin/git-advanced-diff-odt-pdf-doc-xls-ppt-25afbf4f1105*

[11] PDFMiner: *https://github.com/euske/pdfminer*

Correction

The September 2020 issue incorrectly stated that *Full Circle* magazine is no longer published. We regret this error. The excellent *Full Circle* is alive and well – and certainly worthy of a visit for anyone interested in the Ubuntu family of Linux systems:

http://fullcirclemagazine.org

Author Bruce Byfield writes, *"Being mistaken is always embarrassing, but I am happy to say that I was wrong, and that* Full Circle *is still active and providing useful information. My apologies to* Linux Magazine, Full Circle*'s editor Ronnie Tucker, and to readers of both magazines."*

Personal micro server

Useful Companion

Turn a Raspberry Pi into a useful personal micro server for streaming Internet radio, reading RSS feeds, jotting down notes, sharing files, and more. *By Dmitri Popov*

Using a Raspberry Pi for streaming Internet radio is probably one of the easiest projects you can do with the little machine. Install a player application that supports streaming, connect an external speaker or headphones to the Raspberry Pi, and you are pretty much done. But why stop there?

Why not improve on radio streaming and add other features that transform your Raspberry Pi into a simple yet useful personal server? Enter Nyttig [1], simple PHP software (Figure 1) I developed that can be used as it is or as a starting point for building your very own Raspberry Pi-based micro server (see the "Why Nyttig?" box).

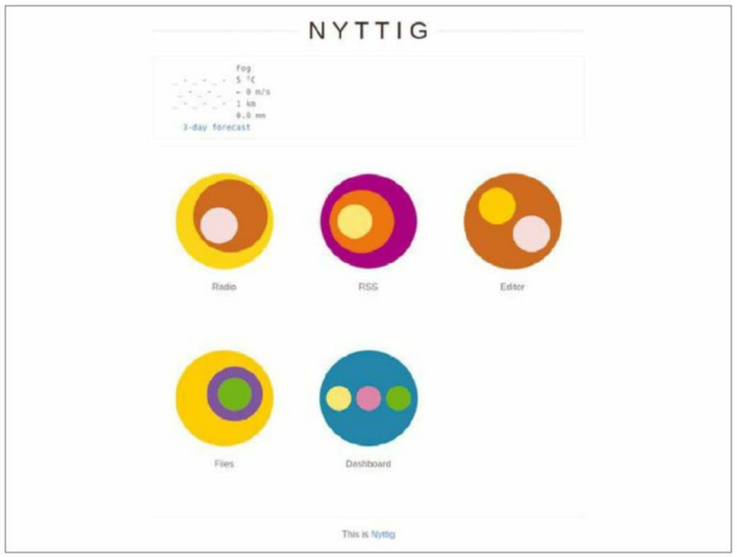


Figure 1: Nyttig offers a handful of simple applications.

Installing Nyttig

Nyttig includes a shell script that automates the entire installation procedure on a Raspberry Pi. Before you install Nyttig, make sure that the Raspberry Pi is running the latest version of Raspberry Pi OS Lite. You also need to confirm that the machine has an Internet connection and is accessible via SSH. Establish an SSH connection to the Raspberry Pi, and run the following command:

```
curl -sSL ➥
  https://gitlab.com/dmpop/nyttig/-/raw/➥
  master/install-nyttig.sh | bash
```

This command fetches and runs the installation script that does all the work for you. During the installation, the script prompts you to install the `puvmeter` utility that turns a Blinkt! [2] LED stick into a colorful VU meter.

Once Nyttig has been installed, open the `nyttig/config.php` file for editing and modify the example settings. By default, access to Nyttig is protected by a password, and it's a good idea to change the default password by modifying the value of the `$PASSWORD` variable. You can also disable password protection by setting the value of the `$PROTECT` variable to `false`. Set the `$CITY` variable to your actual city to allow Nyttig to fetch weather conditions. You can leave the rest of the variables at their default values for now.

While you can deploy Nyttig with a proper web server like Apache or NGINX, it runs perfectly well using the server that comes with PHP. Better still, the installer script automatically adds a cron job that starts the PHP server on boot. To enable Nyttig, restart the Raspberry Pi and point your browser to the Raspberry Pi's IP address (see the "Handy Little Helper" box for locating the IP address.)

Why Nyttig?

With so many excellent open source applications out there, why would you choose Nyttig instead of just installing existing tools or deploying something like Nextcloud? There are several reasons why Nyttig may be a better solution. First, Nyttig is lightweight. It consumes very little resources, and it runs happily even on something like Raspberry Pi Zero. Simplicity is another important Nyttig trait. All PHP scripts in Nyttig are intentionally made as simple as possible, which has two benefits: You can figure out how to use Nyttig's features in a matter of minutes, and you can easily tweak, improve, and extend Nyttig even with modest PHP skills.

Lead Image © Andrea Danti, 123RF.com

Using Nyttig

While it's possible to control the player running on the Raspberry Pi via an SSH connection, it's hardly practical in everyday use. And Nyttig provides a no-frills web interface that allows you to switch between stations, control volume, and manage radio stations (Figure 2). The list of radio stations contains a couple of examples. Pick the one you like, plug headphones or an external speaker into the Raspberry Pi's audio jack, and hit the *Play* button. Of course, you can add your own stations to the list. First, you need to obtain a playlist link for the desired radio station. One of the best places to do that is the radio-browser website [4], which maintains a large database of radio stations around the world. Find the station you like, download its `.pls` file, and add it to the list of radio stations in Nyttig using the upload form.

During playback, you can adjust the volume by specifying the desired value

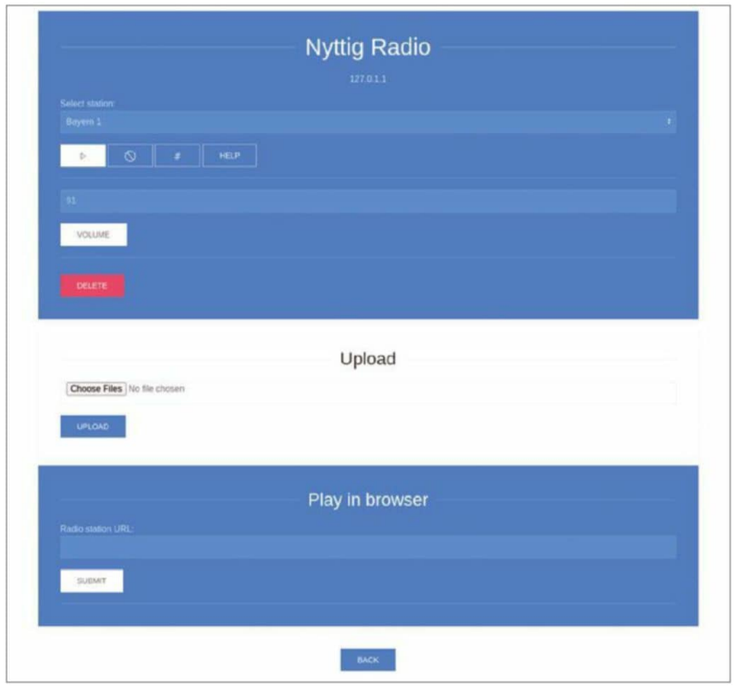


Figure 2: The Nyttig Radio interface lets you manage radio stations and control playback.

Handy Little Helper

If the Raspberry Pi running Nyttig sits permanently on your local network, then finding its IP address is not that difficult. But what if you carry your little micro server with you and want to use it on different wireless networks? How can you connect it to the desired WiFi network without access to the Raspberry Pi? Comitup [3] helps to solve this catch-22. Install the utility, and it automatically connects the Raspberry Pi to the default WiFi network. If the network is not found, Comitup enables the Access Point mode, so you can connect directly to the Raspberry Pi and select the desired WiFi network.

in the appropriate field and pressing *Volume*. By default, the radio interface controls the volume of the headphones device (on Raspberry Pi, it's the 3.5mm audio jack). But if you are using a different sound device, you need to specify it in the `config.php` file. To find what sound devices are available on the Raspberry Pi, run the `aplay -L` command.

Speaking of sound devices, since Nyttig can easily run on Raspberry Pi Zero, you can use the latter to build a diminutive personal server that can fit in a shirt pocket and can be powered by a battery. Raspberry Pi lacks an audio jack, but this problem can be solved by adding a cheap generic external USB sound card. If you opt for this solution, you have to set the value of the `$SOUND` variable in the `config.php` file to `Master`. To enable the USB sound card on Raspberry Pi, use the command

```
sudo nano /usr/share/alsa/alsa.conf
```

to open the `alsa.conf` file for editing and locate the following lines:

```
defaults.ctl.card 0
defaults.pcm.card 0
```

Modify the lines, so that they look as follows:

```
defaults.ctl.card 1
defaults.pcm.card 1
```

Then, reboot the Raspberry Pi Zero to enable the sound card.

In addition to the basic radio controls, the radio interface provides a couple of

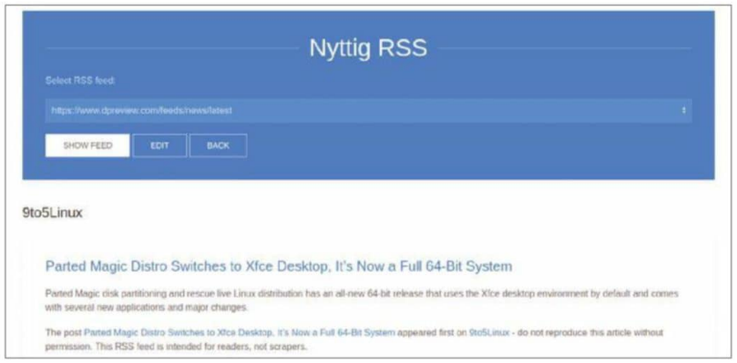


Figure 3: Nyttig's no-frills RSS reader is perfect for quickly checking your favorite feeds.

other features. The # button plays white noise instead of radio. This can come in useful when you work in a noisy environment, but you find listening to a radio too distracting.

Instead of streaming a radio station to the external speaker or headphones connected directly to the Raspberry Pi, you can listen to it in your browser. Enter the streaming link in the *Radio station URL* field, press *Submit*, and use additional controls to listen to the radio.

With so many excellent RSS aggregators at your disposal, Nyttig's RSS reader may seem superfluous (Figure 3). However, it's not designed to replace a proper RSS aggregator, but rather offers you a quick and easy way to check your favorite RSS feeds when you have a spare minute. Since the RSS reader is not exactly overloaded with the creature comforts of a proper RSS aggregator, it's supremely easy to use. Select a feed from the list of feeds and press *Show Feed* to view articles in the feed. To add and remove the feeds, press *Edit*. That's all there is to it.

Figure 4: You can use the Nyttig Editor for jotting down notes and drafting texts.

Figure 5: Need to quickly share a file or a photo? Use Files to do that.

If you often find yourself using random text files for jotting down notes or drafting articles, then you might appreciate the editor that comes with Nyttig (Figure 4). While it's decidedly barebones, it does have one clever trick up its sleeve. Whenever you press the *Save* button, the editor saves the previous version as a separate file with a time stamp. So the editor keeps track of all previous versions, and none of your edits are lost.

Need to share a file or an image with someone on the same local network? Use Nyttig's Files tool (Figure 5). Upload a file, and give the recipient the generated link.

Finally, the dashboard interface provides a basic overview of the server's vitals, including processor temperature, CPU load, and RAM usage (Figure 6). You can use the appropriate buttons to restart and shut down the Raspberry Pi.

Tweaking and Extending Nyttig

Since Nyttig consists of just a handful of simple PHP scripts, you can tweak and extend it to your heart's content. What functionality you can add depends entirely on your imagination and PHP skills. Here are a couple of examples to get you started.

The simplest improvement that requires no coding is to enable an existing module in Nyttig. In the `nyttig` directory, you'll find a folder named `did`. It contains a small application that can be used to record what you've accomplished during the day. DID supports Markdown, so you can format your entries as lists. The tool creates a separate file for each day when you add the first entry and then appends subsequent entries to the file. DID is not included in Nyttig by default, but there is an empty tile for it on Nyttig's main page. To enable DID, open the `index.php` file for editing and locate the empty tile with the `Empty tile` comment in it. Replace the comment with the following HTML code:

```
<a href="did/">
  <img data-src="did/favicon.png"
  width="" height="" alt="" uk-img></a>
<p>DID</p>
```

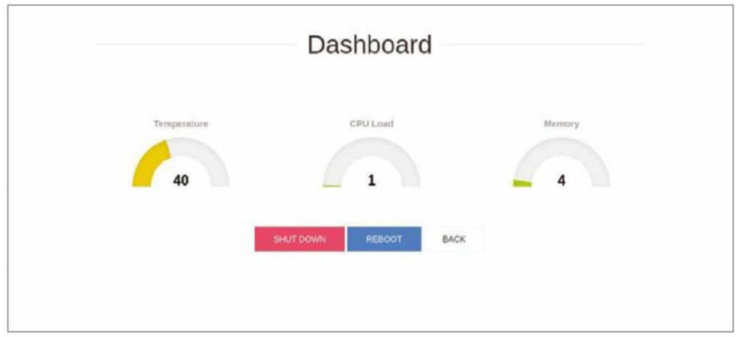



Figure 6: Dashboard gives a quick overview of resource usage.

Save the changes and you are done. Refresh Nyttig's main page, and you should see the DID tile.

The next tweak involves a bit of PHP coding. Using just a few lines of PHP code, you can display a random line from a text file on Nyttig's main page. This can be useful if you want to show random quotes, new words and expressions, or something else. First prepare a list and save it in Nyttig's directory under the `list.txt` name. Open the `index.php` file for editing and locate the following line:

```
<h1 class="uk-heading-line
  uk-text-center">
  <span>N Y T T I G</span></h1>
```

Add the following code snippet right after the line:

```
<p class="uk-text-center">
```

```
<?php
$lines = file('list.txt');
echo $lines[array_rand($lines)];
?>
</p>
```

Save the changes, refresh Nyttig's main page in the browser, and you should see a random line from the text file displayed right under the header.

To try something more advanced, you can add a module that pulls a photo from NASA's Astronomy Picture of the Day service and displays it in a lightbox. For this tweak to work, you need to install a PHP JSON library on the Raspberry Pi using the command:

```
sudo apt install php-json
```

Assuming you want to add the new module to the empty tile, replace the comment in it with the code block shown in Listing 1.

Listing 1: Adding a Module to an Empty Tile

```
<?php
  $request = "https://api.nasa.gov/planetary/apod?api_key=DEMO_KEY";
  $response = file_get_contents($request);
  $data = json_decode($response, true);
  $apod = $data['hdurl'];
?>
<div uk-lightbox>
<a href="<?php echo $apod; ?>"><img data-src="dummy/favicon.png" width=""
   height="" alt="" uk-img></a> <p>Astronomy Picture of the Day</p>
</div>
```

The code in Listing 1 performs several tasks. First, it sends an HTTP request to NASA's API. It then reads the received JSON-formatted response and extracts the value of the `hdurl` key (this value is the URL to the actual image). The obtained URL is then used as an image source in the HTML code that displays it in a lightbox.

By the way, Nyttig is based on the UIkit framework [5], and it's worth perusing UIKit's documentation to understand how you can use it to tweak and improve Nyttig.

Wrap Up

Despite being decidedly simplistic, Nyttig can quickly transform a Raspberry Pi (or any Linux-based machine for that matter) into a no-frills personal micro server. However, its second most important goal is to serve as a foundation for building your own solution with a minimum of coding skills. So whether you just want to turn a Raspberry Pi into a radio streaming device, or you plan to develop an all-in-one platform that can handle a variety of tasks, Nyttig might be just what you need.

Info

[1] Nyttig: *https://gitlab.com/dmpop/nyttig*
[2] Blinkt!: *https://shop.pimoroni.com/products/blinkt*
[3] Comitup: *https://davesteele.github.io/comitup*
[4] radio-browser: *https://www.radio-browser.info/gui*
[5] Ulkit: *https://getuikit.com*

Author

Dmitri Popov has been writing exclusively about Linux and open source software for many years, and his articles have appeared in Danish, British, US, German, and Russian magazines and websites. You can find more about his work on his Tokyo Made (*https://www.tokyoma.de*) website.

Finding and retrieving Google Drive files with Go

Book Collector

Mike Schilli does not put books on the shelf; instead, he scans them and saves the PDFs in Google Drive. A command-line Go program then rummages through the digitized books and downloads them as required. *By Mike Schilli*

I have lots of PDFs of scanned books on my Google Drive and often download what I need from my digital bookshelf to my hard drive [1]. The browser interface on *drive.google.com* is very useful for this (Figure 1). However, Google Drive could be easier and faster to use when searching for books by listing the results and downloading matches immediately. The Go program presented in this issue does this at the command line, which goes down well with programmers who feel at home in the terminal window and are reluctant to ever leave it.

Building the source code from Listings 1 to 4 [2] and calling the generated binary that accepts a search string like `algorithms-in-cpp` is shown in Figure 2. On the user's Google Drive, the program finds the PDF for the book *Algorithms in C++*. It offers up the file for selection and downloads it after confirmation. While the PDF, which is around 150MB in size, is crossing the wire, the Go program displays a slow or fast progress bar, depending on the Internet connection, to give an impression on the number of bytes received in relation to the expected total number.

Owners Only

Of course, I don't let every Tom, Dick, or Harry have access to my treasured and paid-for books. Therefore, a newly written client like the `gd` program shown in Listing 1 first has to identify itself to Google Drive as personally authorized by me. This is not done by means of a username and password, but rather by following an OAuth 2 flow and then by means of access tokens, which the client can renew from locally cached refresh tokens after the former have expired.

The main program in Listing 1 uses the *flag* package to read the search term from the command line; the pattern provided by the user needs to match one or more files in Google Drive. In the version shown in this issue, the program searches for matches with file names. A full-text search would also be possible with a minor change to the source code.

Before sending the user through the OAuth 2 flow, line 20 parses the `credentials.json` file, which defines the client secrets (i.e., the data with which Google identifies the application – the Go program, not the user). The user's consent is later obtained by a browser dialog, started by the user by cutting and pasting a URL produced by the Go program, when started for the first time. Equipped with valid credentials, `pickNGet()` in line 35 then sets out to offer up the user documents that match the search term and downloads them upon request.

What the client is ultimately allowed to do after the user has given their permission is determined by the so-called OAuth scope, which developers set in their applications in advance and tell Google when they register the application. When the user later agrees to let

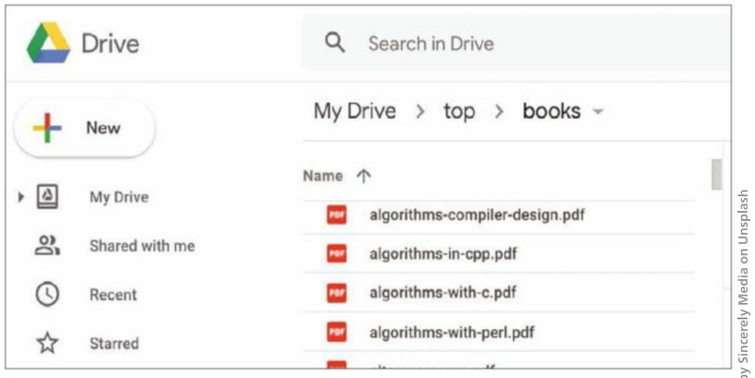


Photo by Sincerely Media on Unsplash

Figure 1: The browser shows scanned programming books as PDF files in Google Drive.

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.


```
$ go mod init gd
go: creating new go.mod: module gd
$ go build gd.go oauth2.go pick.go download.go
go: finding module for package github.com/schollz/progressbar/v3
go: finding module for package golang.org/x/net/context
go: finding module for package google.golang.org/api/drive/v3
go: finding module for package golang.org/x/oauth2/google
go: finding module for package golang.org/x/oauth2
go: found github.com/schollz/progressbar/v3 in github.com/schollz/progressbar/v3 v3.5.0
go: found golang.org/x/net/context in golang.org/x/net v0.0.0-20200822124328-c89045814202
go: found golang.org/x/oauth2 in golang.org/x/oauth2 v0.0.0-20200902213428-5d25da1a8d43
go: found google.golang.org/api/drive/v3 in google.golang.org/api v0.31.0
$ ./gd algorithms-in-cpp.pdf
Download algorithms-in-cpp.pdf (y/[n])? y
Downloading algorithms-in-cpp.pdf (299145819) ...
downloading  22% |[redacted]                                | (65/285 MB, 1.232 MB/s) [52s:2m58s]
```

Figure 2: The Go program shown in Listings 1 to 4 searches Google Drive for files and downloads them if requested.

the client have access, Google displays the scope to make sure the user knows what's at stake. Possible scope values range from access to meta-information of the files stored on the drive to actual reading and download rights, up to unlimited write access. Listing 1 defines `DriveReadonlyScope` for the Go program; this allows the client to query the file names and download the content in question.

Second-Class Citizen Go

While the developer pages for the Google Drive API show examples in Java, Python, and Node.js (Figure 3), Google doesn't seem to consider Go to be a first-class citizen.

I was rubbing my eyes in disbelief, while scrolling through auto-generated spaghetti code, having to reverse engineer how to call the API functions with their correct signatures. On Stack Overflow, you will find five-year-old requests from perplexed programmers, and only a few brave developers have ever found an answer to handling even the most trivial of tasks.

Of course, you could also access the web API directly in Go, but if Google already provides an SDK, then they should document it and maintain it to keep it up-to-date.

Figure 3: Except for Java, Python, and Node.js, Google offers very little help in the use of the Google Drive API.

OAuth, Take Two

So where does the `credentials.json` file come from? It is the result of the developer registering the client program with Google as an API application. On the Developers Console for the Google API [3], you first need to enable the Google Drive API with a valid Google account (Figure 4). On the same page, you then have to register the Go client as a desktop application and define how it will be repre-

Listing 1: gd.go

```
01 package main
02
03 import (
04   "flag"
05   "fmt"
06   "golang.org/x/oauth2/google"
07   "google.golang.org/api/drive/v3"
08   "io/ioutil"
09   "log"
10   "os"
11 )
12
13 func main() {
14   flag.Parse()
15   if flag.NArg() != 1 {
16     log.Fatalf(fmt.Sprintf("usage: %s partial-name",
       os.Args[0]))
17   }
18   query := flag.Arg(0)
19
20   b, err := ioutil.ReadFile("credentials.json")
21   if err != nil {
22     log.Fatalf("Error reading client secret file: %v", err)
23   }
24   config, err := google.ConfigFromJSON(b, drive.
     DriveReadonlyScope)
25   if err != nil {
26     log.Fatalf("Error parsing config: %v", err)
27   }
28
29   client := oauth2Client(config)
30   srv, err := drive.New(client)
31   if err != nil {
32     log.Fatalf("Error retrieving gdrive client: %v", err)
33   }
34
35   err = pickNGet(srv, query)
36   if err != nil {
37     log.Fatalf("Error retrieving document: %v", err)
38   }
39 }
```

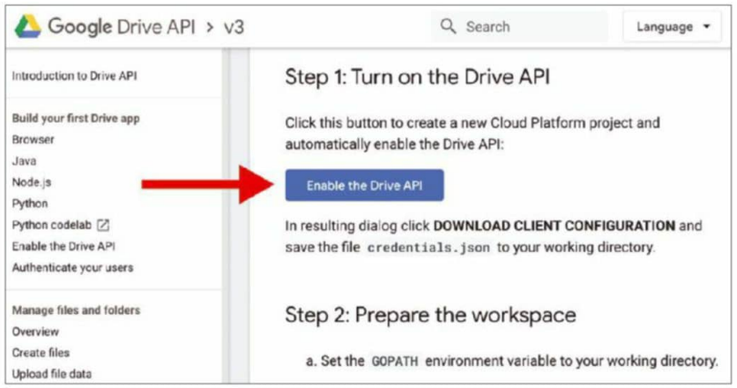


Figure 4: Developers can click to enable the API on Google Drive.

Figure 5: The JSON download after creating a desktop client on the API console, in the *Credentials* tab.

```
$ ./gd algorithms-in-cpp
Go to the following link in your browser then type the authorization code:
https://accounts.google.com/o/oauth2/auth?access_type=offline&client_id=732904659045-
rqeucf0fjs3skomjhpu9d5uvamsmm6ml.apps.googleusercontent.com&redirect_uri=urn%3Aietf%3
Awg%3Aoauth%3A2.0%3Aoob&response_type=code&scope=https%3A%2F%2Fwww.googleapis.com%2Fa
uth%2Fdrive.readonly&state=state-token
4/3gFpqjD_APrkZLNuGdSM4Z1p7TpBYIKnBFoJDfKghfz_xUfxKlb6iWY
Saving credential file to: token.json
...
```

Figure 6: On first launch, the program from Listing 1 expects the `credentials.json` file; it outputs a URL, which guides the user through the Google OAuth 2 flow in the browser.

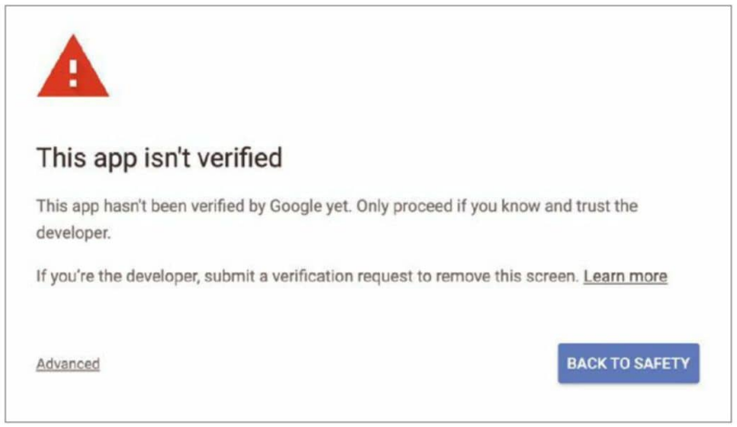


Figure 7: Google warns you against using the unverified app, but *Advanced* lets you proceed at your own risk.

sented later when Google asks the user whether they want to grant the app access to their Google Drive data.

Google lets you download the registration data in JSON format under the file name `client-secret*` (Figure 5), but you should rename it, because the client in Listing 1 expects the data as `credentials.json`. Keep in mind though that the app is not given access to user data on the basis of these credentials yet; instead, it's only for registration purposes of the app with Google.

When the Go program is later launched for the first time, it finds the JSON credentials file in the same directory and prints a link to standard output (Figure 6), which the end user is instructed to copy into the URL field of a web browser before being directed to the OAuth 2 flow.

Seal of Approval Still Missing

Google then asks the user logged in with their Google account whether they want to grant the app read access to their Google Drive data. Since the Go program is a homemade app and does not yet bear the Google seal of approval, the dialog in Figure 7 warns the user against granting access. Brave readers of this column will want to press *Advanced* and grant the Go program read access to their Google Drive anyway on the following page.

After confirming again, the Google OAuth 2 flow finally outputs a hex code (Figure 8), which you need to copy to the standard input of the Go program that has been waiting at the command line. When you type `Enter`, the program devours the code and continues to run. It contacts the Google server with the code and receives an access and refresh token from the server. Listing 1 then bundles these credentials into a local JSON file, which it stores as `token.json` in the same directory.

Now that it has persistent credentials, Listing 1 no longer sends the user through the OAuth 2 flowwhen next called, but can use the access token in the JSON file to read the user's data on Google Drive. It is important to protect this JSON file against unauthorized access: Anyone in possession of the token can gain access to your Google Drive data. However, write access is not possible, because the scope was previously defined as read-only during the OAuth 2 flow.

Hop, Skip, and Google

Listing 2 implements the `oauth2Client` function to let the program collect and manage the token when run for the first time. When done, it returns an HTTP client to the main program, which handles user authentication under the hood when communicating with the Google Drive web server.

When the program is first called, there is no token in the `token.json` file yet, so calling `readCachedToken()` in line 15 returns an error. This is then remedied by calling the `fetchAccessToken()` function in line 17. Starting in line 23, the code prints out a Google URL, which you need to paste into your web browser; from there, you then go through the OAuth 2 flow.

At the end, the flow shows a hex code that the user then copies into the main program waiting at the command line. Line 29 grabs the code from the standard input; line 33 exchanges it on the Google server for an access token, which line 18 stores in JSON format on the local filesystem. On subsequent invocations of the program, the code instructs `oauth2Client()` to retrieve the token from the local file without the previously necessary riga-

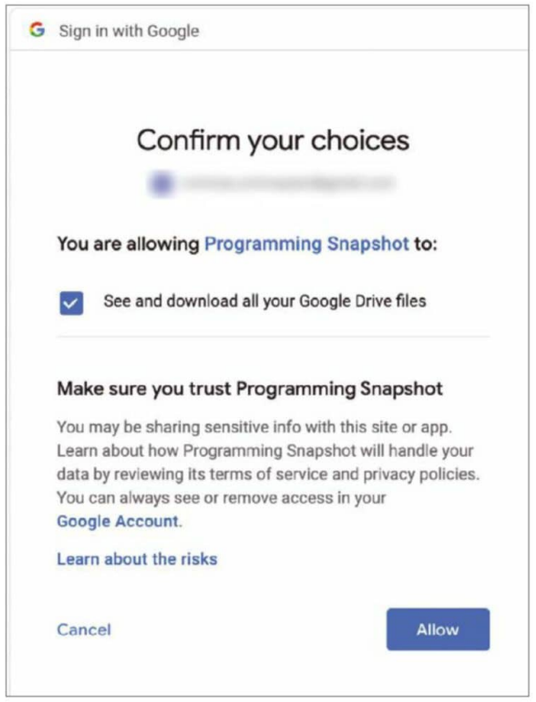


Figure 8: After further confirmation, Google creates an access token, which the app can use to access the drive.

Listing 2: oauth2.go

```
01 package main
02
03 import (
04   "encoding/json"
05   "fmt"
06   "golang.org/x/net/context"
07   "golang.org/x/oauth2"
08   "log"
09   "net/http"
10   "os"
11 )
12
13 func oauth2Client(config *oauth2.Config) *http.Client {
14   tokFile := "token.json"
15   tok, err := readCachedToken(tokFile)
16   if err != nil {
17     tok = fetchAccessToken(config)
18     cacheToken(tokFile, tok)
19   }
20   return config.Client(context.Background(), tok)
21 }
22
23 func fetchAccessToken(config *oauth2.Config) *oauth2.Token {
24   url := config.AuthCodeURL("state-token",
     oauth2.AccessTypeOffline)
25   fmt.Printf("Point browser to %v, follow the flow and then
     paste "+
26     "the code here.\n", url)
27
28   var authCode string
29   if _, err := fmt.Scan(&authCode); err != nil {
30     log.Fatalf("Error reading auth code %v", err)
31   }
32
33   tok, err := config.Exchange(context.TODO(), authCode)
34   if err != nil {
35     log.Fatalf("Error getting access token: %v", err)
36   }
37   return tok
38 }
39
40 func readCachedToken(file string) (*oauth2.Token, error) {
41   f, err := os.Open(file)
42   if err != nil {
43     return nil, err
44   }
45   defer f.Close()
46   tok := &oauth2.Token{}
47   err = json.NewDecoder(f).Decode(tok)
48   return tok, err
49 }
50
51 func cacheToken(path string, token *oauth2.Token) {
52   fmt.Printf("Saving credential file to: %s\n", path)
53   f, err := os.OpenFile(path,
     os.O_RDWR|os.O_CREATE|os.O_TRUNC, 0600)
54   if err != nil {
55     log.Fatalf("Can't write token to %s: %v", path, err)
56   }
57   defer f.Close()
58   json.NewEncoder(f).Encode(token)
59 }
```

marole. No more extra Google hops needed at this point.

The *json* package from the Go core collection makes reading and writing token data a breeze. Juggling the tokens to gain access to a protected resource isn't as convenient; someone might want to sit down and provide a standardized token handling package on GitHub for everyone else to use.

The `pickNGet()` function in Listing 3 sends the search query to Google Drive. The service stores files in a freely definable hierarchy of folders. However, users often simply search for file content or names, and the search engine giant returns a unique ID for matching files.

My file names are usually unique, which is why line 14 in Listing 3 uses the search query `name contains x`. If you prefer to let Google search for text chunks in the content, just replace the string in line 14 with `fullText contains x`. Further search queries are explained in the API document [4].

Machine-Generated SDK

It turns out that the Go SDK for the Google API provided by Google is simply a machine-generated wrapper around the web API.

The API endpoint for listing files is addressed by `srv.Files.List()` (line 15). The concatenated calls to `Q(q).PageSize(100)` append the search query and set the maximum number of matches delivered for each request to 100. If so desired, the user can pick up the next batch of matches on the next call, but Listing 3 does not do this, because a command-line client would be unsuitable for processing more than 100 matches anyway.

The concatenated `Fields()` function limits the fields returned per match to the unique document ID, the file name, and the size of the file in bytes. `pickNGet()` can therefore offer the user a compact list for selection. Pressing *Y* starts the download.

The `for` loop starting in line 28 iterates over all the matches in the `r.Files` array and prompts the user on `os.Stdin` at each pass to press *Y* if they want to download the current match to their local drive. To allow this to happen, the `ReadString()` function in line 30 waits for keyboard input from the user, which is sent after the Enter key gets pressed. If the user did not type `y` (e.g. by accepting the default `"n"`), line 32 uses `continue` to go to the next round, asking the user what they want to do with the next match.

If the user wants to download a matching file, line 34 sets the progress bar to zero percent and the maximum length to the size of the file. The Google Drive call `srv.Files.Get()` in line 37 selects the desired file by referencing its ID and calls `Download()` to initiate the download. In this case, the Google Drive server sends back a download URL, which the client docks onto, and the download process starts via HTTPS.

Line 43 defines a buffered reader from the standard *bufio* package for the incoming data stream. In line 44, the program passes the reader to the `download()` function from Listing 4, together with a reference to the progress bar `bar` and the file name to store it under later locally.

Reader Without a Type

The fact that a function accepts a reader interface is typical of Go. The function receives an object that can use the `Read()` method with which it can consume the data. It does not matter where the data come from – a local file, the web, or a database. In this way, Go, which is so type-strict otherwise, opens itself up to polymorphism, without needing tedious declarations. The consuming function simply calls `Read()` and is not in the least interested how the data transpired.

Chunk by Chunk

Wherever data are read and written chunk by chunk, several things can go wrong at any given time, and the programmer has to keep an eagle eye on the process to ensure that the data written to the target file arrive intact (i.e., that the local bytes are 100 percent identical to those on Google Drive).

When loaded off the web, data usually trickle in in small portions. Even a file of several hundred megabytes usually arrives at the downloading application in chunks of no more than 32KB. Since the recipient already knows in advance how large the entire file is going to be from the meta-information provided initially, it only needs to string together the chunks to restore the file on the client side. At the same time, it knows at all times what percentage of the data has already arrived and how much is still outstanding.

Careful When Copying

Listing 4 optimistically creates a buffer of 1MB in `data` in line 19, into which the `Read()` function drops the next incoming chunk of data. However, incoming packets are typically only 32KB in size; the rest of the buffer remains unoccupied when `Read()` returns. The number of bytes actually present is available in the return value `count`, and line 28 uses `data[:count]` to reduce the length of the byte slice to the actual length of the data by truncating the trailing garbage data.

If the data stream from Google Drive dries up because the end of the file is reached, `Read()` returns `io.EOF` in line 22 as an error code. But be careful: This does not mean that there's no data left in `data`. Instead, `count` also again shows you in this case the number of bytes that arrived before the EOF. A client that disregards this last morsel because it thinks that there is nothing left on an EOF will create corrupted download files.

The back end of the copy loop has the writer code, which receives pieces of data from the reader and stores them in a file previously opened for writing. Not only does it have to write the very last bit from the reader into the target file (the bit arrives at the same time as the EOF), but it also has to empty its internal buffers at the very end, by flushing their contents into the target file. If this were left out in line 39, the last few bytes would be missing in the generated PDF document, which causes astounding erratic behavior in some PDF readers. Don't ask me how I know that.

Conclusions

Presto! The fully functional Google Drive client is now ready to use as a command-line tool – always ready for action, "information at your fingertips," as Bill Gates once said. And since you no longer have to leave the command line to go to the bookstore, you can save time and outpace the competition! ■■■

Info

[1] "Scanning and storing books on Google Drive," by Mike Schilli, issue 146, January 2013, p. 64

[2] Listings for this article: *ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/242/*

[3] Google Developers Console: *https://console.developers.google.com*

[4] Google Drive API searches: *https://developers.google.com/drive/api/v3/search-files*

Listing 3: pick.go

```
01 package main
02
03 import (
04   "bufio"
05   "fmt"
06   pb "github.com/schollz/progressbar/v3"
07   "google.golang.org/api/drive/v3"
08   "log"
09   "os"
10   "strings"
11 )
12
13 func pickNGet(srv *drive.Service, query string) error {
14   q := fmt.Sprintf("name contains '%s'", query)
15   r, err := srv.Files.List().Q(q).PageSize(100).
16     Fields("nextPageToken, files(id, name, size)").Do()
17   if err != nil {
18     log.Fatalf("Error retrieving files: %v", err)
19   }
20
21   if len(r.Files) == 0 {
22     fmt.Println("No files found.")
23     return nil
24   }
25
26   reader := bufio.NewReader(os.Stdin)
27
28   for _, file := range r.Files {
29     fmt.Printf("Download %s (y/[n])? ", file.Name)
30     text, _ := reader.ReadString('\n')
31     if !strings.Contains(text, "y") {
32       continue
33     }
34     bar := pb.DefaultBytes(file.Size, "downloading")
35
36     fmt.Printf("Downloading %s (%d) ...\n",
       file.Name, file.Size)
37     dwn, err := srv.Files.Get(file.Id).Download()
38     if err != nil {
39       log.Fatal("Unable to get download link %v", err)
40     }
41     defer dwn.Body.Close()
42
43     reader := bufio.NewReader(dwn.Body)
44     err = download(reader, file.Name, bar)
45     if err != nil {
46       log.Fatal("Download of %s failed: %v", file.Name, err)
47     }
48   }
49
50   return nil
51 }
```

Listing 4: download.go

```
01 package main
02
03 import (
04   "bufio"
05   pb "github.com/schollz/progressbar/v3"
06   "io"
07   "os"
08 )
09
10 func download(r io.Reader, lpath string, bar
   *pb.ProgressBar) error {
11   outf, err := os.OpenFile(lpath,
     os.O_WRONLY|os.O_CREATE, 0644)
12   if err != nil {
13     return err
14   }
15   writer := bufio.NewWriter(outf)
16   defer outf.Close()
17
18   total := 0
19   data := make([]byte, 1024*1024)
20
21   for {
22     count, rerr := r.Read(data)
23     if rerr != io.EOF && rerr != nil {
24       return err
25     }
26     total += count
27     bar.Add(count)
28     realdata = data[:count]
29
30     _, werr := writer.Write(realdata)
31     if werr != nil {
32       return werr
33     }
34
35     if rerr == io.EOF {
36       break
37     }
38   }
39   writer.Flush()
40   return nil
41 }
```

Network monitoring from the cloud

Lookout

Netdata helps you monitor your network with ease through a cloud dashboard. *By Mayank Sharma*

Netdata [1] is a distributed, real-time, performance and health monitoring tool that can be used to monitor machines running Linux, FreeBSD, and macOS, as well as Kubernetes and Docker. Available for free under a GPLv3 license, you can run Netdata on physical machines, virtual machines, containers, and even on Internet of Things (IoT) devices, thanks to its minimal resource footprint.

Netdata's dashboard balances both form and function and does a nice job of visualizing a computer's processes and services. You can use Netdata to monitor the CPU, RAM usage, disk I/O, and network traffic, along with several other aspects of the systems on which it runs. In addition to hardware, it can also keep an eye on web servers, databases, and applications. Netdata's interactive dashboard can also store long-term historical metrics for days, weeks, or months, all at one second granularity.

Designed by system administrators, DevOps engineers, and developers, Netdata not only visualizes the collected metrics, but it also identifies and troubleshoots complex performance problems without wasting time. In fact, Netdata complements its monitoring features with an alarm notification system that will detect performance and availability issues.

One of the best things about Netdata is that the tool ships with sensible defaults, letting you put it into active service immediately after installation. That said, once you are more familiar with Netdata, you can rely on its extensive customization options and tune it to better align with your requirements.

Client Rollout

To use Netdata, you'll have to install the Netdata Agent on all your computers. The agent also installs its own custom database engine to store all the collected metrics. These will then be visualized on the cloud-based dashboard.

Netdata offers several installation options [2], although the recommended way is to use its one-line installation script. The script will fetch all the components necessary to compile the Netdata Agent on your computer. Fire up a terminal on the machine you want to monitor and enter the following command as a regular Linux user:

```
$ bash <(curl ➥
  -Ss https://my-netdata.io/kickstart.sh)
```

That's all there's to it. The script will refresh the package management repositories and install all the dependencies before compiling the Netdata Agent. It'll also ensure that the agent keeps itself updated with nightly releases.

Another popular option is to install the Netdata Agent inside a Docker container [3]. This is useful for a one-off analysis of a host, since it makes no permanent changes to the host computer and can be easily removed.

After installing the agent, fire up a web browser on the computer and head to *http://localhost:19999*. This will bring up Netdata's dashboard (Figure 1) and show you the live metrics from the system. You can also view the metrics from any other computer on the network. Just replace *localhost* with the IP address of the computer on which you've installed the agent.

Monitor Multiple Machines

To monitor multiple machines, use the one-line installation script to install the agent on all the computers you want to monitor. The agents are distributed by design, which means they operate independently of each other and only collect and chart the metrics for the system on which they are installed.

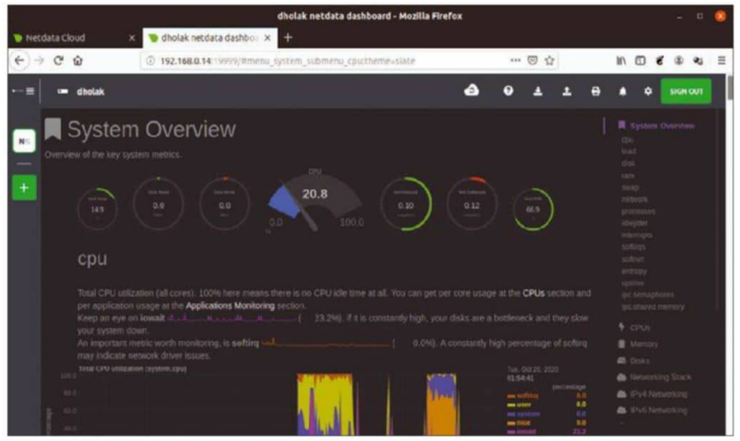


Figure 1: Netdata claims it collects thousands of metrics per server every second, using just about one percent of a single core's CPU.

To string together the various agents into a single interface, you'll have to sign up for a free account with the Netdata Cloud service, which will collate and display metrics from all the agents deployed on your network (see the "Netdata Cloud" box).

Once you have the login credentials, bring up the dashboard on any of the monitored computers. Click the *Sign In* button at the top of the dashboard and then enter your Netdata Cloud account credentials when prompted. You'll be redirected back to the Netdata dashboard, which now sports a new menu on the left. The computer will be listed under the *Visited Nodes* section.

To add more computers, navigate to their respective dashboards and use the *Sign In* button to connect them to your Netdata Cloud account. As you connect more computers, they'll start populating the *Visited Nodes* section. Once you've added more than one node, you can switch between them from the menu.

Netdata Cloud

Netdata Cloud shows you all the computers on your network in single screen view. To get started, head to *https://app.netdata.cloud* and use any of the options to create an account with the service.

Once your account has been created, the Netdata Cloud will take you through a brief on-boarding process. During this process, you'll be asked to create at least one Space and one War Room. The Spaces and War Rooms help you organize the computers in your network. Think of them as virtual spaces that help you customize how you and your team use Netdata Cloud to monitor your infrastructure.

You can read about the Netdata Cloud's organizational benefits to familiarize yourself with the concept of Spaces and War Rooms [4]. For now, just create one of each as requested. You'll use them later to monitor your computers.

Claim Systems

When you sign into the Netdata Cloud service from the dashboard, your computers will be listed under the *Visited Nodes* section. You should also take a moment to add computers to the Space you created when signing up for the Netdata Cloud account.

The process of adding a node to a Space is called claiming. Claiming makes sure you have administrative access to the computer and the Netdata Agent running on it. You'll be given a chance to claim a computer during the signup process for your Netdata Cloud account. Of course, you can also do this later from within the Space as well.

To add a computer to a Space, log into the Netdata Cloud account and select a Space from the list on the left. Now click the green + icon within the Space, which will reveal a rather long command (Figure 2). Copy the command, and paste it in a terminal on the computer you want to claim.

After verifying your credentials, the computer will be added to the Space (Figure 3). Now you can repeat the claiming process on every computer you want to add to Netdata Cloud. Remember that you can only claim a computer inside a single Space. Once claimed, however, you can add that computer to multiple War Rooms within the Space.

Dashboard Tour

Now that you've installed the agents on multiple computers and can access them from the Cloud dashboard, it's time to familiarize yourself with the dashboard.

Netdata collects monitoring data from dozens of hardware and software components, such as CPU, memory, disks, networking, filesystems, and more. What makes it even more useful though is that Netdata can also collect metrics

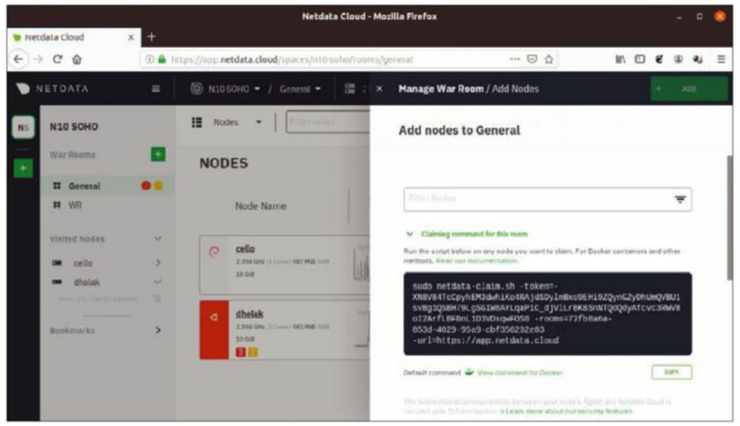


Figure 2: Netdata uses the Agent-Cloud link (ACLK) to securely transmit the metrics from the agent to the Cloud.

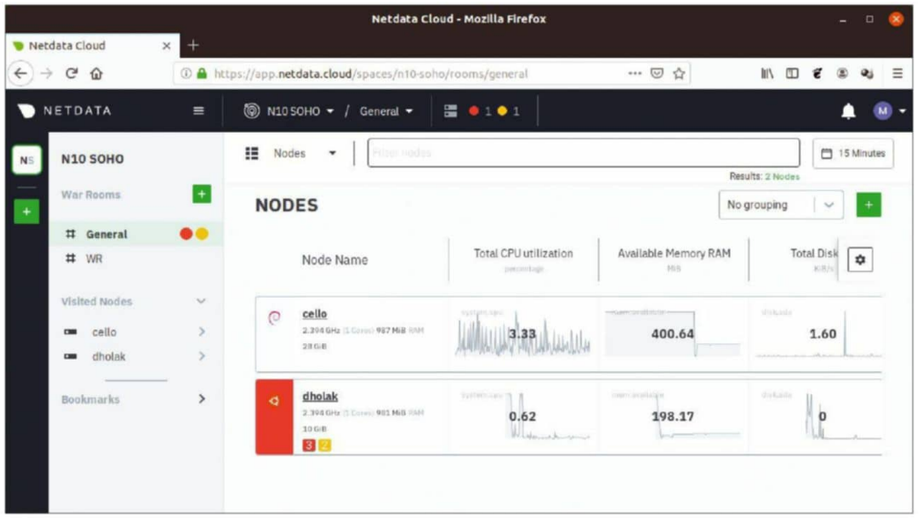


Figure 3: In addition to the metrics, Netdata also transmits information regarding all configured alarms and their current status to the Cloud.

from hundreds of popular services and applications.

Netdata deploys collectors to gather the metrics. It includes collectors for collating performance data from some of the popular services and apps, such as Apache, NGINX, Tomcat, MySQL, Postgres, MongoDB, Ceph, OpenLDAP, Tor, Docker, and more [5].

All the collected metrics are exposed via the Netdata dashboard as interactive charts. Netdata shows all its charts on a single scrollable page. You can also navigate between the various elements using the menu placed on the dashboard's right-hand side. Note, however, that if you run Netdata on multiple computers that run different operating systems or different versions, the menus might look a little different for each one.

Using the mouse, you can drag the charts to the left or right to move forward and backward through the different time intervals. Similarly, you can change the time markers by holding down the Shift key as you scroll within a chart. To reset a chart to its default view, simply double click inside it.

The good thing about Netdata's visualization is that when you change the view on one chart, it automatically replicates the same view on the other charts as well. Thanks to this feature, you'll always get a synchronized view of the metrics.

The charts themselves are self-explanatory. At the top, you get an overview of the computer's resources. This is followed by a summary of the computer's CPUs, including their utilization and information about the interrupts handled by each, in addition to other aspects. Similarly, you get real-time information about the system's memory utilization, and so on (Figure 4).

Most of the charts have a brief description to explain the feature they display along with its importance. Unless you're well-versed with monitoring Linux/BSD systems, you should spend some time exploring the individual metrics and how they can be used to monitor your systems' health.

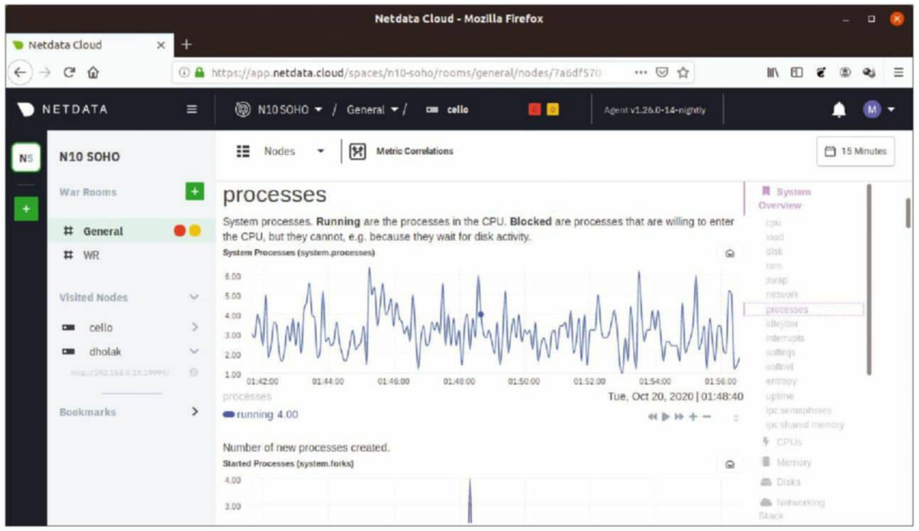


Figure 4: In addition to the built-in collectors, you can pull in additional ones via plugins.

Get Alerts

In addition to the active performance monitoring, the Netdata Agent can also help you ensure your systems and applications are healthy by alerting you about possible issues. The Netdata Agent includes dozens of preconfigured alarms that trigger alerts when a

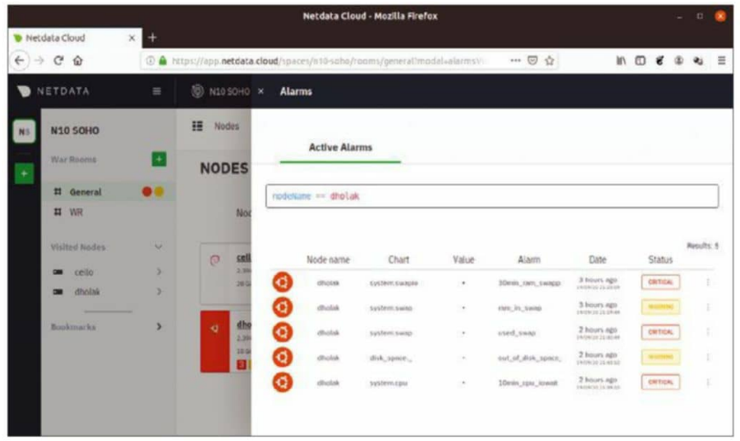


Figure 5: You can view details about the raised alarms, as well as go to the chart where the alarms were raised for further analysis.

monitoring component requires your attention.

As mentioned earlier, these alarms are preconfigured with sensible defaults. Just like Netdata itself, these alarms have been designed by the tool's system administrator community, which means the alarms will be activated automatically upon the agent's installation. That said, while you don't need to edit them, the alarms can be customized to meet your needs.

You can access Netdata's alarm notifications system by clicking the alarms button (the bell icon) at the top of the dashboard. This will bring up a screen that shows the currently raised alarms, along with tabs to view all running alarms, as well as the alarms log (Figure 5).

To tune a default alarm, switch to the *All* tab. This page will list the various alarms along with their preconfigured settings. The *source* row in the tab points to the configuration file that controls the settings for a particular alarm (Figure 6). You'll need to edit the file and adjust the settings as per your requirements.

For instance, the `/usr/lib/netdata/conf.d/health.d/ram.conf` file controls the alarms related to a computer's physical RAM. By default, Netdata will warn you when the amount of used RAM crosses the 80 percent threshhold. You can change this behavior by editing the value in the *warn* line.

After you've saved the file, you can reload the health monitoring settings with:

```
$ sudo netdatacli reload-health
```

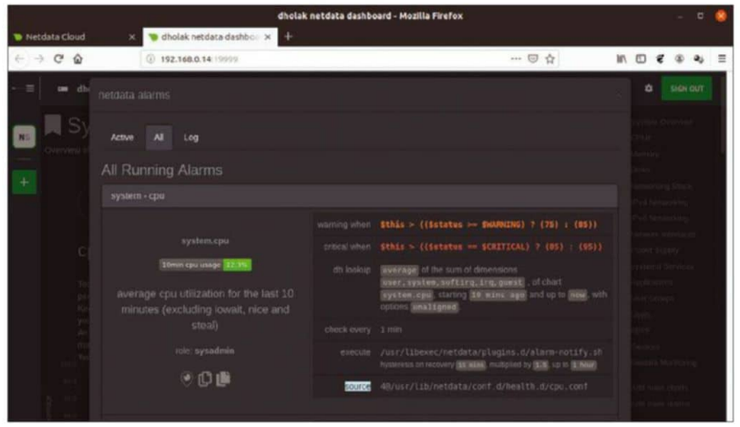


Figure 6: Below every alarm's name is a badge that updates automatically to show the chart's current value.

See the project's documentation on the health monitoring system [6] to understand the other lines in an alarm's configuration file.

Going Further

Once you've become accustomed to Netdata, it's time to explore the various settings and configure it to meet your requirements. The project has excellent documentation (as referenced throughout the article).

While I've covered most of Netadata's basic features, there's a lot more you can do with it. You can, for instance, export and import snapshots [7] of your dashboard's contents, which helps diagnose major errors and anomalies. You can also create custom dashboards that do a better job of visualizing the metrics in which you are interested.

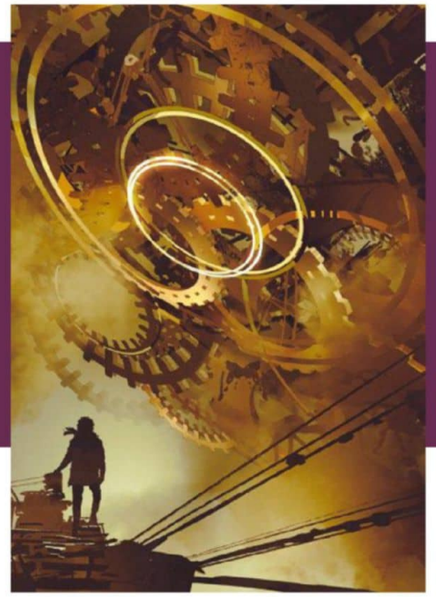
Despite server monitoring being an already crowded space, Netdata has managed to create a wide berth for itself thanks to its ease of use and customizability. No wonder then that it is one of the most starred projects in the Cloud Native Computing Foundation landscape. ■■■

Info

[1] Netdata: *https://netdata.cloud*
[2] Installation options: *https://learn.netdata.cloud/docs/agent/packaging/installer#have-a-different-operating-system-or-want-to-try-another-method*
[3] Netdata client inside Docker: *https://github.com/netdata/netdata/blob/master/packaging/docker/README.md*
[4] Spaces and War Room: *https://learn.netdata.cloud/docs/cloud/organize*
[5] Netdata collectors: *https://learn.netdata.cloud/docs/agent/collectors/collectors*
[6] Health monitoring system: *https://learn.netdata.cloud/docs/agent/health*
[7] Exporting/Importing snapshots: *https://learn.netdata.cloud/guides/step-by-step/step-07#export-and-import-a-snapshot*

Author

Mayank Sharma has been writing and reporting on open source software from all over the globe for almost two decades.



MakerSpace

Home-built shooting game with Nerf targets and a Raspberry Pi

Ready, Aim, Fire

A cool Nerf gun game for a neighborhood party provides a lesson in Python coding with multiple processors.

By Scott Sumner

Last Halloween, I was asked to put together a Nerf game for a local neighborhood party. Not wanting to do just the same old thing, I got together with a couple of local makers, and we built a set of electronic targets to create a real-life tower defense game! Although COVID put a damper on this year's plans, we still managed to get most of the hardware together to expand the experience for a second round.

Figure 1: The original Nerf targets: The blue disks are worth one point; the red disk in the center tower is worth three points.

The original game had three targets: an Arduino Uno [1] brain, an audio amplifier, and a loudspeaker. Each target was placed inside a wooden structure that we called a "tower" (Figure 1). The object of the game was to shoot at the targets with a gun firing Nerf darts. A confetti canon was also built into the tower to announce when the tower "fell," which means that the target on the tower had sustained a predefined number of hits from the Nerf gun. We built two identical tower sets, one for each end of the field. Each system operated independently. The game monitors were responsible for powering down the system when the other team won.

The larger targets are worth one point each, and the smaller center target is worth three points. Games can be selected to run between 10 and 100 hits. A "traffic light" health gauge on the center tower gave an approximate value of how many hits remained before the game ended (see Figure 1).

When we deployed the system a second time, we arranged the hardware a little differently. Each target now has a single piezo sensor and its own NodeMCU-based (open source firmware) Arduino. Targets are powered by 3.7V LiPo batteries, so they are self-contained. Hits are transmitted to a server running on a Raspberry Pi on the edge of the playing field. Score announcements are still made by audio because outdoor video displays are unwieldy and expensive.

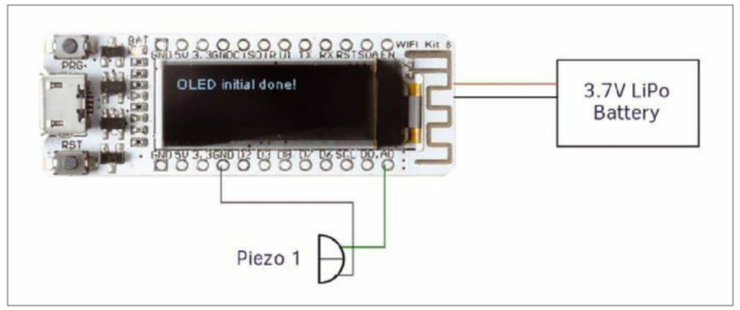


Figure 2: Schematic for the second version of the Nerf tower.

More Inputs

The ESP8266 only has a single analog input. Although it possible to wire additional piezos in parallel, it is also possible to add a little bit of external hardware and expand the analog capability as well. Consider the LM339 quad comparator chip. It has four comparators that operate independently. Each comparator has a positive and a negative input. When the voltage on the positive input is higher than the voltage on the negative input, the output is active. It should be noted that the output of each comparator switches between ground and high impedance rather than V+. So if you want to use this with your Arduino, you'll need to turn on the pull-up resistor on whatever pin you connect it to.

This circuit connects the piezo to the positive input. The negative input will have a potentiometer with the lower end grounded, the top end connected to V+, and the wiper connected to the positive input of the comparator. The potentiometer then is the trigger level, or the voltage that needs to be exceeded by the piezo (how hard a hit) to enable the output.

With this one chip, I can add up to four piezo sensors to my target. The downside to this approach is that each piezo requires an associated variable resistor to tune its sensitivity. The variable resistor can't be adjusted remotely like the analog input can.

Arduino Code

The schematic for the second version of the Nerf tower is shown in Figure 2. Each node has only a single piezo, and all of the sound production now comes from the Raspberry Pi, which greatly simplifies the nodes compared to the original design. The nodes themselves are massively simplified. The main processor in this version is an ESP8266 on a WiFi Kit 8 [2]. (See the box entitled "More Inputs.")

Each target is a self contained wireless node. A NodeMCU-based processor connects to WiFi and transmits hits to a central game server. The Arduino code is simpler, because all it does is report hits. All of the game logic lives in the game server itself. Each node also accepts a few commands to control game play or calibrate the node.

The code for the Arduino is shown in Listing 1. There are several different things needed to set up a WiFi connection so that's what most of these includes are for (lines 1-7). `Arduino.h` provides info about the hardware this sketch is currently running on (line 8), and `U8g2lib.h` is a library to draw on the attached OLED screen on the back of the board.

Listing 1: Arduino Code

```
001 #include <ESP8266WiFi.h>
002 #include <ESP8266WiFiGeneric.h>
003 #include <ESP8266WiFiMulti.h>
004 #include <ESP8266WiFiSTA.h>
005 #include <ESP8266WiFiType.h>
006 #include <WiFiClient.h>
007 #include <WiFiUdp.h>
008 #include <Arduino.h>
009 #include <U8g2lib.h>
010
011 #ifdef U8X8_HAVE_HW_SPI
012 #include <SPI.h>
013 #endif
014 #ifdef U8X8_HAVE_HW_I2C
015 #include <Wire.h>
016 #endif
017
018 const char* ssid = "YOURNETWORKNAME";
019 const char* password = "YOURNETWORKPASSWORD";
020 const char* server = "IP_OF_SERVER";
021
022 int iTrigger = 333;
023 int iHits = 0;
024 unsigned long ulNextHit = 0;
025
026 WiFiClient client;
027
028 U8G2_SSD1306_128X32_UNIVISION_F_HW_I2C u8g2(U8G2_R0,
      /* reset=*/ 16, /* clock=*/ 5, /* data=*/ 4);
029
030 void setup(void) {
031   int iConnectCount = 0;
032
033   u8g2.begin();
034
035   WiFi.mode ( WIFI_STA );
036   WiFi.begin ( ssid , password );
037
038   u8g2.clearBuffer();    // clear the internal memory
039   u8g2.setFont(u8g2_font_ncenB08_tr);
        // choose a suitable font
040   u8g2.drawStr(0,10,"Connecting to AP");
        // write something to the internal memory
041   u8g2.sendBuffer();
        // transfer internal memory to the display
```

Listing 1: Arduino Code (Continued)

```
042
043  while ( WiFi.status() != WL_CONNECTED )
044  {
045    delay ( 500 );
046    u8g2.clearBuffer();
047    u8g2.drawStr(0,10,"Connecting to AP");
048    u8g2.setCursor ( 0 , 20 );
049    u8g2.print ( iConnectCount );
050    iConnectCount ++;
051    u8g2.sendBuffer();
052  }
053
054  if (!client.connect( server , 9000 )) {
055    u8g2.clearBuffer();  // clear the internal memory
056    u8g2.setFont(u8g2_font_ncenB08_tr);
         // choose a suitable font
057    u8g2.drawStr(0,10,"Connection Failed");
         // write something to the internal memory
058    u8g2.sendBuffer();
059    delay(5000);
060    return;
061  }
062
063  u8g2.clearBuffer();
064  u8g2.setFont(u8g2_font_ncenB08_tr);
065  u8g2.drawStr ( 0 , 10 , "Connected to AP" );
066  u8g2.setCursor ( 0 , 20 );
067  u8g2.print(WiFi.localIP());
068  u8g2.sendBuffer();
069 }
070
071 void loop(void) {
072   int iSensor = 0;
073
074   iSensor = analogRead ( A0 );
075
076   u8g2.clearBuffer();
077   u8g2.setCursor ( 0 , 10 );
078   u8g2.print ( iSensor );
079   u8g2.setCursor ( 30 , 10 );
080   u8g2.print ( iTrigger );
081   u8g2.setCursor ( 70 , 10 );
082   u8g2.print ( iHits );
083   u8g2.setCursor ( 10 , 20 );
084   if ( ulNextHit > millis() ) u8g2.print ( ulNextHit -
        millis() );
085   u8g2.sendBuffer();
086
087   if ( iSensor > iTrigger && millis() > ulNextHit )
088   {
089     iHits += 1;
090     ulNextHit = millis() + 1000;
091
092     if ( client.connected() == false )
093     {
094       client.connect ( server , 9000 );
095     }
096     client.print ( "H:" );
097     client.println ( iHits );
098   }
099
100   if ( client.available() > 0 )
101   {
102     String received = client.readStringUntil ( ';' );
103     switch ( received [ 0 ] )
104     {
105       case 'R': iHits = 0;break;
106       case 'T': iTrigger = received.substring
            ( 1 ).toInt();break;
107       case 'D': ulNextHit = millis() + received.substring
            ( 1 ).toInt();break
108     }
109   }
110 }
```

Lines 11-16 are a unique kind of `if` statement. `ifdef` is an `if` statement that is interpreted by the compiler itself rather than the C code in other parts of the sketch. In this case, if `U8x8_HAVE_HW_SPI` is defined (line 11) then `include <SPI.h>` (line 12). `endif` on line 13 closes the block. Lines 14-16 work the same way, but I'm checking for I2C and including `Wire.h` instead.

Any variables that are defined outside of functions are global – they can be accessed by any function in the program. On lines 18-20, I'm defining constants. These constants can't be changed once they are set up. In this case, it is the network SSID (line 18), network password (line 19), and the IP of the game server (line 20).

On line 22, `iTrigger` is the value that the analog input must go above to trigger a hit. I default to 333, but this can be changed by the server once it connects. Line 23 sets up `iHits`, which is how many hits this particular node has registered. The game server tracks hits for scoring purposes, but this value is used to update the display, so it is easy to confirm that the sensor is working. Line 24 sets up `ulNextHit`, which generally is assigned from the `millis()` function plus a delay. Once `millis()` is greater than this value, then the timer has expired.

Line 26 creates `client` as an instance of `WiFiClient`. This client's purpose is to interact with all of the network commands. Finally, line 28 sets up all of the connections for the OLED display attached to the board.

Just like in any other Arduino program, `setup` runs once. The `setup` function (lines 30-69) defines some variables, connects to the WiFi, sets up the display, and then connects to the game server.

The main loop appears in lines 71-110. Each cycle through the main loop updates the display with some internal readings, checks to see if the sensor has been hit, and, finally, sees if there are any characters waiting to receive and process.

Line 72 initializes `iSensor`; then I do an `analogRead` on `A0` to get the voltage coming off the piezo disk [3]. On lines 76-85, I update the display with three values: the most recent analog reading from `A0`, the current value of `iTrigger`, and `iHits`. This hit counter is only for this node, not game wide. Then, if `ulNextHit` is greater than `millis()`, I display the remaining time out value – how long until this node will respond to a hit again.

If `iSensor` is greater than `iTrigger`, then this target has been hit. I also check that `millis()` is greater than `ulNextHit`. If it's not, then this target is currently disabled. If all that checks out, I increment the hit counter with `iHits += 1` (line 89) and disable the target for one second with (line 90):

```
ulNextHit = millis() + 1000;
```

Line 92 checks to make sure that the client is still connected to the game server, and if not, line 94 re-establishes the connection. Finally, I send the string `H:` and the number of hits this node has recorded.

Line 100 checks `client.available()` to see if there are any characters waiting. If the value is greater than zero, the client has received a message and needs to process it. Line 102 reads incoming data into `received` until it finds a semicolon.

The `switch` statement on lines 103-108 checks the first character of the string `received [ 0 ]` to see if it's a command. If it's an `R`, then I should reset the hit counter with `iHits = 0`. If it's a `T`, I use `received.substring` to get the rest of the string, use `toInt` to make it an integer, and set `iTrigger` with this value. That changes how sensitive the piezo disk is. Finally, if it's a `D` then I set `ulNextHit` to `millis()` plus the integer value of whatever the rest of the received string equals. This sets the delay before any new hits will be registered. A delay of zero will re-enable the target immediately.

The Server

The Raspberry Pi server receives information from the Arduino clients and manages the game. The server software is written in Python and shows a text user interface in a terminal (Figure 3). All of the game logic lives in the server software. Whenever a hit is received from the wireless nodes, the server announces "Red target hit!" or "Blue target hit!" Targets are assigned to colors in the server interface. Similarly, the length of the game (number of hits) is also adjusted there. The server can also adjust each node's input sensitivity and enable a global disable on all the node's targets. Disabling the targets is handy for pausing the game for additional instructions.

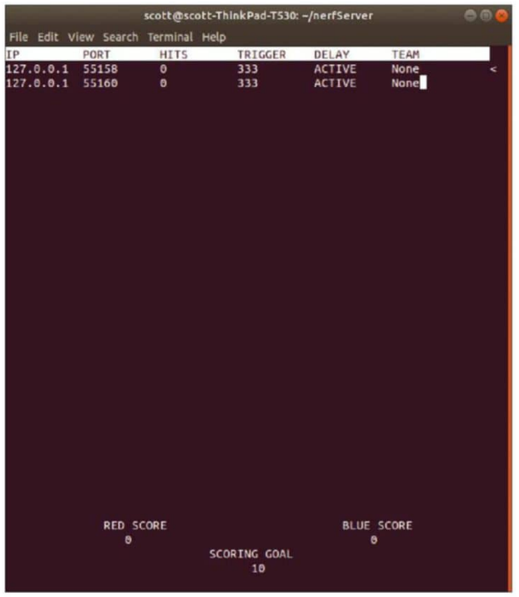


Figure 3: The user interface presented by the server is text only, so it will run in any terminal. The connections are simulated with netcat, as noted by the *127.0.0.1* address.

I've utilized multiple threads for this server; each task is split off to its own process and deposits input into a queue as it is received. Then the main process handles input from the threads as it arrives and takes care of the human interface. Each thread can block or wait as long as it needs to, because it is running independently of the main program.

Listing 2 shows the server code. The external libraries imported for the game appear in lines 1-7. `socket` accesses the computer's networking hardware and allows for network communication. `thread` allows programs to separate into subprocesses. `curses` controls character cell displays (like terminal windows). `time` lets you access the system clock and other timing functions. `os` allows you to talk to the operating system that the program is running on. `Queue` creates first-in/first-out thread-safe channels to communicate between different threads.

`send` (lines 25-31) is the "transmitter" of the socket. It accepts a single parameter, `msg`, which is the data to be sent. Line 26 sets up `totalsent`, which is the number of bytes actually transmitted. Sockets don't always send all the data at once. Each time you call `self.sock.send`, it will return how many bytes it actually sent. It is up to the program to keep calling `self.sock.send` until the entire message has been transmitted. Here I accomplish this with

```
while totalsend < len ( msg )
```

(line 27). If I try to send and zero bytes get transmitted, there's a problem with the socket connection. This is caught on lines 29-30. Finally

```
totalsent = totalsent + sent
```

updates the transmitted byte count.

In lines 33-45, `receive` is the counterpart to `send`. Line 36 asks the socket for a data chunk with `self.sock.recv(64)`. (`64` asks for no more than 64 bytes.) If the data returned is empty, then there's a problem with the socket connection. Lines 37-38 check for that. Then `strip chunk` removes any whitespace characters and adds the result to `self.receivedData` (line 39).

Line 40 checks for a semicolon in `self.receivedData` – the end of an incoming message. There's nothing special about the semicolon as end of message; its just the character I picked to signify it. If a semicolon is found, I create the string `output`, which is all of `self.receivedData` up to the location of the semicolon (line 41):

```
self.receivedData.index ( ";" )
```

`nodeClass` (lines 47-81) is the server representation of the hardware node I described earlier. The `gameServer` class (lines 83-252), which is initialized when the program starts and sets every-

thing else in motion, is the main entry point in the code.

I initialize the curses library [4] to provide character-cell management of the terminal window and then set up variables that I use to manage instances of `nodeClass` and the screen itself. `self.allNodes` stores each instance of the class. `self.nodeIndex` is a count of the number of nodes that have connected. `self.cursorIndex` and `self.oldCursor` track the cursor movement in the user interface. Other variables track each team's score and the number of points to win the game.

The main loop appears in lines 116-178. Line 116 is an infinite `while` to continually process events from both the nodes and user input. Line 117 checks if `self.q.empty()` is `False`. If it is (the queue is *not* empty), then line 118 gets the next entry from the queue. If the first character of the received data is an "H" (a target has been hit), then I set up a loop to walk down `self.allNodes`. Once I find the matching IP address and port, I increase the hit counter, set the delay to one second from now, increment the appropriate team's score, and announce the hit.

The next instance of `thread.start_new_thread.self.speak` executes an external program, so it will block until the program finishes running. In this case, the external program is the speech program that makes an announcement. By launching the speech program in its own thread, the main program will continue to run even while speech is happening. Finally, I call `self.checkScore`, since scores have been updated. This will redraw the score on the interface. If a milestone has been hit, it will also announce a percentage of health remaining.

Lines 133 and 134 loop through all of the nodes again and call `statusLine`. The argument is the reference to `self.screen`, the curses screen buffer.

Line 137 uses `self.screen.getch()` for the most recent key pressed (if any). If no key has been pressed, then it will return `-1`. If a key has been pressed, then I move on to the blocks of `if/elif` to process each key. Lines 139-149 manage responses to a user pressing the arrow keys. The up and down arrows move the cursor up and down in the node list. The left and right arrows change the sensitivity for how hard the dart has to hit the target.

Lines 150-153 let you press *r* or *b* for `self.allNodes [ self.cursorIndex ].team` to refer to either the `RED` or `BLUE` team. This value is used for the overall game score to decide which team should be credited with a hit.

Pressing *x* on the keyboard calls the current node's `clearHits` method, which resets the node's hit counter to zero.

The `checkScore` function (lines 180-217) draws the score to the user interface and also plays speech if certain milestones are reached. Line 191 checks to see if `self.redScore` is greater than or equal to 50 percent of `self.scoreGoal` and that `self.announceRed` is less than 5. This signifies that the 50 percent announcement has not yet been spoken. If both conditions are true, then `self.announceRed` is set to 5 (line 194) and `speak` is called in a new thread. This process repeats for 70 percent, 90 percent, and 100 percent (game over). The entire block is then repeated for the blue team.

The `speak` function (lines 220-234) makes a system call to eSpeak [5]. Each type of message is customized within each `if/elif`. If this is a `HIT` then say "[team color] target hit!"

If this is a `SCORE` update and it's the red team, then pause for 1.5 seconds and say "red base " (calculate percent remaining by subtracting `self.announceRed` from 10 and then multiply by 10) and then say "percent." The message ends up being something like "red base 50 percent". The same thing happens for the blue team.

If the message is `WINNER`, wait 1.5 seconds and then say "[team color] base has fallen, game over!"

Starting Up

The `startListening` function (lines 236-242) invokes all of the socket commands to open a socket and listen for incoming connections from nodes.

`socket.socket` creates a socket; its parameters `socket.AF_INET` and `socket.SOCK_STREAM` specify IPv4 addresses and TCP sockets respectively. `serversocket.bind` tells the socket where to listen for connections. `socket.gethostname` asks the system what its name is on the network. You can also specify `127.0.0.1` to only accept local connections. The second argument is the port number, 9000 in this case. Then I call `serversocket.listen` for up to five connections. If connection number 6 arrives before any previous connections have been processed, the sixth connection attempt will fail. Once earlier connections have been established, further connections can proceed normally.

Line 240 enters an infinite loop to `accept` incoming connections and create a `self.node` thread for each connection. `startListening` itself runs in its own thread so the infinite loop waiting for connections won't hold up the main program.

The `node` function (lines 244-252) accepts three parameters: the `client` socket, the `address` its coming from, and `q`, which is the communications path back to the main thread. `node` runs in its own thread, so after creating its own socket handler and adding it to the list of `allNodes`, it sits in an infinite loop. When `data` is received, it is sent up the queue if it is not equal to `None`.

Line 254 instantiates an instance of the `gameServer` class. Without this step, everything I've just talked about is only definitions. This step sets everything in motion.

Different Games and Future Expansions

Since the game logic now resides on the Raspberry Pi, it will be easy to implement different game modes in the future. Our plan for next year is to add an RGB LED that can be controlled by the server. This addition will allow targets to change teams, randomly deactivate themselves, or become "destroyed" in the process of game play. Other than modifying the node with instructions for an RGB LED, changes will only have to take place in the server.

I hope this project has shown how a microcontroller and a larger server can interact. In this example, it's for a fun block party game, but the same concepts apply to any project with distributed sensors or smaller processing nodes. Even if you're not using wireless, the same principles apply to any project with multiple processors that need to talk with each other.

Info

[1] Arduino Uno: *https://store.arduino.cc/usa/arduino-uno-rev3*

[2] WiFi Kit 8: *https://heltec.org/project/wifi-kit-8/*

[3] Piezo disks: *https://en.wikipedia.org/wiki/Piezoelectric_sensor*

[4] Python curses library: *https://docs.python.org/3/library/curses.html*

[5] eSpeak text-to-speech engine: *http://espeak.sourceforge.net/*

Listing 2: Server Code

```
001 import socket
002 import thread
003 import pprint
004 import curses
005 import time
006 import os
007 import Queue
008
009 class MySocket:
010   """demonstration class only
011     - coded for clarity, not efficiency
012   """
013
014   def __init__(self, sock=None):
015     if sock is None:
016       self.sock = socket.socket(
017           socket.AF_INET, socket.SOCK_STREAM)
018     else:
019       self.sock = sock
020     self.receivedData = ""
021
022   def connect(self, host, port):
023     self.sock.connect((host, port))
024
025   def send(self, msg):
026     totalsent = 0
027     while totalsent < len ( msg ):
028       sent = self.sock.send(msg[totalsent:])
029       if sent == 0:
030         raise RuntimeError("socket connection broken")
031       totalsent = totalsent + sent
032
033   def receive(self):
034     chunks = []
035     bytes_recd = 0
036     chunk = self.sock.recv(64)
037     if chunk == b'':
038       raise RuntimeError("socket connection broken")
039     self.receivedData += chunk.strip()
040     if ";" in self.receivedData:
041       output = self.receivedData
            [ :self.receivedData.index ( ";" ) ]
042       self.receivedData = self.receivedData
            [ self.receivedData.index ( ";" ) + 1 : ]
043       return output
044     else:
045       return None
046
047 class nodeClass:
048   def __init__
        ( self , address , port , socket , displayLine ):
049     self.address = address
050     self.port = port
051     self.socket = socket
052     self.hits = 0
053     self.trigger = 333
054     self.delay = 0
055     self.team = None
056     self.displayLine = displayLine
057
058   def statusLine ( self , screen ):
059     if time.time() < self.delay:
060       delayLeft = int ( ( self.delay - time.time() ) *
                1000 )
061     else:
062       delayLeft = "ACTIVE"
063
064     colSize = int ( ( curses.COLS-1 ) / 6 )
065     screen.addstr ( self.displayLine + 1 , 0 , self.
                address )
066     screen.addstr ( self.displayLine + 1 , colSize , str
                ( self.port ) )
067     screen.addstr ( self.displayLine + 1 , colSize * 2 ,
                str ( self.hits ) )
068     screen.addstr ( self.displayLine + 1 , colSize * 3 ,
                str ( self.trigger ) )
069     screen.addstr ( self.displayLine + 1 , colSize * 4 ,
                str ( delayLeft ) )
070     screen.addstr ( self.displayLine + 1 , colSize * 5 ,
                str ( self.team ) )
071
072   def changeTrigger ( self , delta ):
073     self.trigger += delta
074     self.socket.send
          ( "T:" + str ( self.trigger ) + ";" )
075
076   def clearHits ( self ):
077     self.hits = 0
078
079   def setDelay ( self , delay ):
080     self.delay = time.time() + delay
081     self.socket.send
          ( "D:" + str ( delay * 1000 ) + ";" )
082
083 class gameServer:
084   def __init__ ( self ):
085     self.screen = curses.initscr()
086     curses.noecho()
087     curses.cbreak()
088     self.screen.keypad ( True )
089     self.screen.nodelay ( True )
090
091     self.allNodes = list()
092     self.nodeIndex = 0
093     self.cursorIndex = 0
094     self.oldCursor = -1
095
096     self.redScore = 0
097     self.blueScore = 0
```

Listing 2: Server Code (continued)

```
098     self.scoreGoal = 10
099
100     self.announceRed = 0
101     self.announceBlue = 0
102
103     self.q = Queue.Queue()
104     thread.start_new_thread ( self.startListening , () )
105
106     colSize = int ( ( curses.COLS ) / 6 )
107     self.screen.addstr ( 0 , 0 , " " * ( curses.COLS ) ,
                             curses.A_REVERSE )
108     self.screen.addstr ( 0 , 0 , "IP" , curses.A_REVERSE
                             )
109     self.screen.addstr ( 0 , colSize , "PORT" ,
                             curses.A_REVERSE )
110     self.screen.addstr ( 0 , colSize * 2 , "HITS" ,
                             curses.A_REVERSE  )
111     self.screen.addstr ( 0 , colSize * 3 , "TRIGGER" ,
                             curses.A_REVERSE  )
112     self.screen.addstr ( 0 , colSize * 4 , "DELAY" ,
                             curses.A_REVERSE  )
113     self.screen.addstr ( 0 , colSize * 5 , "TEAM" ,
                             curses.A_REVERSE )
114     self.checkScore()
115
116     while 1:
117       if self.q.empty() == False:
118         data = self.q.get ( False )
119         #print ( data [ 1 ] + " from " + data [ 0 ]
                  [ 0 ] + " port " + str ( data [ 0 ]
                  [ 1 ] ) )
120         if data [ 1 ] [ 0 ] == "H":
121           for nd in self.allNodes:
122             if nd.address == data [ 0 ] [ 0 ] and
                  nd.port == data [ 0 ] [ 1 ]:
123               nd.hits += 1
124               nd.delay = time.time() + 1
125               if nd.team == "RED":
126                 self.redScore += 1
127                 thread.start_new_thread
                      ( self.speak , ( "RED" , "HIT" ) )
128               elif nd.team == "BLUE":
129                 self.blueScore += 1
130                 thread.start_new_thread
                      ( self.speak , ( "BLUE" , "HIT" ) )
131               self.checkScore()
132
133       for nd in self.allNodes:
134         nd.statusLine ( self.screen )
135
136
137       key = self.screen.getch()
138       if key != -1:
139         if key == curses.KEY_UP:
140           self.cursorIndex -= 1
141           if self.cursorIndex < 0: self.cursorIndex = 0
142         elif key == curses.KEY_DOWN:
143           self.cursorIndex += 1
144           if self.cursorIndex > len
                ( self.allNodes ) - 1:
145             self.cursorIndex = len ( self.allNodes ) - 1
146         elif key == curses.KEY_LEFT:
147           self.allNodes [ self.cursorIndex
                          ].changeTrigger ( -10 )
148         elif key == curses.KEY_RIGHT:
149           self.allNodes [ self.cursorIndex
                          ].changeTrigger ( 10 )
150         elif key == ord ( 'r' ):
151           self.allNodes [ self.cursorIndex ].team =
                          "RED"
152         elif key == ord ( 'b' ):
153           self.allNodes [ self.cursorIndex ].team =
                          "BLUE"
154         elif key == ord ( 'x' ):
155           self.allNodes [ self.cursorIndex ].clearHits()
156         elif key == ord ( '1' ):
157           for nd in self.allNodes:
158             nd.setDelay ( 10 )
159         elif key == ord ( '3' ):
160           for nd in self.allNodes:
161             nd.setDelay ( 30 )
162         elif key == ord ( '6' ):
163           for nd in self.allNodes:
164             nd.setDelay ( 60 )
165         elif key == ord ( "-" ):
166           self.scoreGoal -= 10
167           if self.scoreGoal < 10: self.scoreGoal = 10
168           self.checkScore()
169         elif key == ord ( "=" ):
170           self.scoreGoal += 10
171           self.checkScore()
172
173       if self.cursorIndex != self.oldCursor:
174         self.screen.addstr
              ( self.oldCursor + 1 , curses.COLS-1 , " " )
175         self.screen.addstr
              ( self.cursorIndex + 1 , curses.COLS-1 , "<" )
176         self.oldCursor = self.cursorIndex
177
178       self.screen.refresh()
179
180   def checkScore ( self ):
181     screenXhalf = int ( curses.COLS / 2 )
182     screenXquarter = int ( screenXhalf / 2 )
183
184     self.screen.addstr
          ( curses.LINES - 5 , screenXquarter - 3 , "RED
          SCORE" )
185     self.screen.addstr
          ( curses.LINES - 4 , screenXquarter , str ( self.
          redScore ) )
186     self.screen.addstr
          ( curses.LINES - 5 , screenXhalf + screenXquarter
          - 4 , "BLUE SCORE" )
187     self.screen.addstr
          ( curses.LINES - 4 , screenXhalf + screenXquarter
          , str ( self.blueScore ) )
188     self.screen.addstr
          ( curses.LINES - 3 , screenXhalf - 6 , "SCORING
          GOAL" )
189     self.screen.addstr
          ( curses.LINES - 2 , screenXhalf , str ( self.
          scoreGoal ) )
190
191     if self.redScore >= int
          ( self.scoreGoal * .5 ) and self.announceRed < 5:
192       self.screen.addstr ( 15 , 5 , "Red Team 50%" )
193
194       self.announceRed = 5
195       thread.start_new_thread
            ( self.speak , ( "RED" , "SCORE" ) )
196     elif self.redScore >= int
          ( self.scoreGoal * .75 ) and self.announceRed < 7:
197       self.announceRed = 7
198       thread.start_new_thread
            ( self.speak , ( "RED" , "SCORE" ) )
199     elif self.redScore >= int
          ( self.scoreGoal * .9 ) and self.announceRed < 9:
200       self.announceRed = 9
201       thread.start_new_thread
            ( self.speak , ( "RED" , "SCORE" ) )
202     elif self.redScore >= self.scoreGoal and self.
          announceRed < 10:
203       self.announceRed = 10
204       thread.start_new_thread
            ( self.speak , ( "RED" , "WINNER" ) )
205
206     if self.blueScore >= int
          ( self.scoreGoal * .5 ) and self.announceBlue < 5:
207       self.announceBlue = 5
208       thread.start_new_thread
            ( self.speak , ( "BLUE" , "SCORE" ) )
209     elif self.blueScore >= int ( self.scoreGoal * .75 )
          and self.announceBlue < 7:
210       self.announceBlue = 7
211       thread.start_new_thread
            ( self.speak , ( "BLUE" , "SCORE" ) )
212     elif self.blueScore >= int ( self.scoreGoal * .9 )
          and self.announceBlue < 9:
213       self.announceBlue = 9
214       thread.start_new_thread
            ( self.speak , ( "BLUE" , "SCORE" ) )
215     elif self.blueScore >= self.scoreGoal and self.
          announceBlue < 10:
216       self.announceBlue = 10
217       thread.start_new_thread
            ( self.speak , ( "BLUE" , "WINNER" ) )
218
219
220   def speak ( self , team , message ):
221     team = team.lower()
222     if message == "HIT":
223       os.system
            ( "espeak \"" + team + " target hit!\"" )
224     if message == "SCORE" and team == "red":
225       self.screen.addstr ( 16 , 5 , "Speaking score" )
226       time.sleep ( 1.5 )
227       os.system
            ( "espeak \"" + team + " base " + str ( ( 10 -
            self.announceRed ) * 10 ) + "percent\"" )
228     if message == "SCORE" and team == "blue":
229       time.sleep ( 1.5 )
230       os.system
            ( "espeak \"" + team + " base " + str ( ( 10 -
            self.announceBlue ) * 10 ) + "percent\"" )
231
232     if message == "WINNER":
233       time.sleep ( 1.5 )
234       os.system ( "espeak \"" + team + " base has
                    fallen, game over!\"" )
235
236   def startListening ( self ):
237     serversocket = socket.socket
            ( socket.AF_INET , socket.SOCK_STREAM )
238     serversocket.bind
            ( ( socket.gethostname() , 9000 ) )
239     serversocket.listen ( 5 )
240     while 1:
241       ( clientsocket , address ) = serversocket.accept()
242       thread.start_new_thread
            ( self.node , ( clientsocket , address ,
            self.q ) )
243
244   def node ( self , client , address , q ):
245     sock = MySocket ( client )
246     self.allNodes.append
            ( nodeClass ( address [ 0 ] , address [ 1 ] , sock
            , self.nodeIndex ) )
247     self.nodeIndex += 1
248
249     while 1:
250       data = sock.receive()
251       if data != None:
252         q.put ( ( address , data ) )
253
254 gs = gameServer()
```

Everyone is happier when things are tidy – even your computer. But most of us are not especially happy about the business of doing the tidying. If we could line up a smart vacuum or a robot feather duster to do the work, a great many of us would be very pleased to focus on other tasks.

The inside of your computer doesn't get cluttered like your garage or your medicine cabinet, but it does collect its own kind of clutter: logs, cached data, temporary files, crash reports. Part of clutter might be system services that still start up when you boot your computer even though you've forgotten why. In this month's Linux Voice we look at Stacer, an easy-to-use tool that will help you tidy up your Linux system. Also inside, we introduce you to a Linux tool that will track your investment portfolio and show you how to create a presentation in Inkscape using the JessyInk add-on.

Image © Olexandr Moroz, 123RF.com

LINUXVOICE

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

MADDOG'S DOGHOUSE

Recent discussions of introducing moral restrictions into free and open source software licensing have maddog remembering all the reasons early developers decided not to go down that road. BY JON "MADDOG" HALL

Problems with Restrictions

Recently there have been some discussions around free software that are focusing on nontechnical issues that might best be described as "moral" issues.

For example, should people be allowed to charge for "free software" and should free software be used for purposes that are "immoral"? Should we build restrictions into our licensing that require "moral use"?

I cannot speak for every piece of software out there or every person who writes it, but we did have these discussions over a quarter century ago, and while I may not be able to find my car keys, I remember these discussions very well.

As some of you are aware, the Linux kernel's copyright is not held by a single person, but by many people who all have licensed their code under the GPL.

As the Linux kernel moved from a "hobby" to a "technical wonder," several of us could see it moving to commercial use.

As we started the discussions, there were some developers who said "I do not want my software to be used by the military, because I do not like war or the military." Some developers said "I do not want my code to be used by banks." Others said "I do not want my code to be used by the church," while still others took the opposite stance of "I do not want my code to be used by atheists."

As we drew the circles of more and more people who told us who they did not want using their code, it became obvious that the Venn diagram of all of these exclusions meant no one would be able to use the code.

Even if you were to narrow down the groups from, as an example, "The Church" to some particular sect of "The Church," the future requirements would eventually cause the kernel to be useless to anyone, and if your code can not be used, then why write it?

Another reason for not going down this path is the fact that there are plenty of pieces of code out there that could be used for immoral purposes – closed source code, other operating system kernels. Plus uses for immoral purposes would still happen even if you tried to prevent it.

A side effect is the legal situation. I know many lawyers and some of them are my very good friends. Lawyers of large companies and governments look at every license with a critical eye. If they feel that the use of a piece of code puts their company in jeopardy for breaking the license, they will recommend the company not use it. Inserting a "morality" clause in the midst of your license is one way of getting corporate lawyers to say "use something else," because one person's idea of morality differs from another.

A corollary to "morality uses" is the concept of "selling code," or using free software in commercial products. Many developers said "I do not want people making money on the software that I write for free."

The genesis of this thinking is that typically free software is written for your own use, or for the use of people like you who are also programmers and contributors to your software. You typically do not mind "sharing" your software with them, but it grates on you that someone is making money off your software.

On the other hand, having these people selling your software and perhaps providing service helps more people become exposed to your software (and you) then might otherwise happen. These people may do the extra work, such as documentation, integration, and training that you do not want to do. Eventually these people may become sponsors of your project, allowing you to work on your software full time.

Another element of "selling free software" is demonstrated in the recent purchase of Red Hat by IBM. Some thought that this was a "sell out," and opposed the sale based on "people making money from free software." On the other hand both IBM and Red Hat create thousands of jobs and develop solutions for people that need them.

You have to come back to the fact that the "free" in free software is not about gratis, but freedom. Even Richard Stallman and the Free Software Foundation took pains to say that people should be able to make money "selling free software."

As the Linux kernel project got closer and closer to version 1.0, the discussions became more and more intense, and we did lose a few very talented programmers because of some of the decisions that were made, including:

- Free software can be used for any purpose.
- You can sell free software as long as you obey the conditions of the license.

You can not create morality through licensing. You create moral people and acts through teaching and requiring morality in life. ■■■

System maintenance with Stacer

Digital House Cleaning

Stacer simplifies the configuration and maintenance of Linux for newcomers by allowing users to handle most of these tasks conveniently in a graphical interface.

BY ERIK BÄRWALDT

Hardly any other operating system offers as many tools for maintenance and support as Linux. However, not everyone is familiar with all the commands, and some tools offer a large number of parameters, which in turn discourages newcomers and those who want to switch operating systems. Stacer [1] makes managing your installation far easier and more efficient, as the program offers a whole range of useful applications for maintaining and servicing the system.

Installation

Stacer can be found in the software repositories of many distributions. The installation is therefore an easy experience with the appropriate system front ends. If you are using a distribution that does not yet include this tool in its package sources, you can find the source code, plus deb and RPM packages on the project's GitHub page.

If none of the packages suits your system and you are worried about compiling the software manually, there is also an AppImage on the GitHub site that will run on any distribution independently of the package management system. After downloading, you just need to assign the appropriate rights to the AppImage before you start it. You can do this in the terminal using the following command:

```
$ chmod +x Stacer-1.1.0-x64.AppImage
```

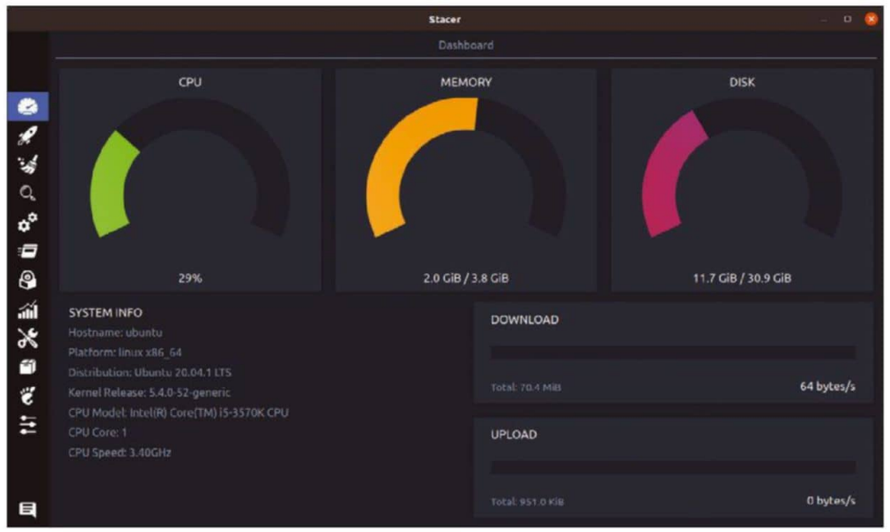
Then call the software using:

```
$ ./Stacer-1.1.0-x64.AppImage
```

Ready, Steady, Go!

After calling Stacer, the program opens a dashboard (Figure 1). You can see at a glance the current CPU, RAM, and storage usage. Stacer also lists some general information about the system. In the lower right third of the window, you can view the network traffic's details.

Figure 1: On launching, Stacer brings up a clearly arranged dashboard.



The software updates the displays virtually in real time. However, the dashboard only provides an overview: For example, it does not break down the CPU load by individual cores, and when graphically displaying the capacity of mass storage devices, it only looks at the current system partition.

In the buttonbar on the left side of the window, clicking on the *Settings* button (second from the bottom) lets you configure the software.

In the Settings dialog, *Alert messages* lets you specify whether you want to trigger an alert when certain thresholds are reached. The software monitors the CPU, RAM, and mass storage if required. You can define a percentage load, which will trigger an alert in Stacer if exceeded.

This helps you discover problems, for example, if a computer's processors are constantly working at full capacity without running computationally intensive applications. If you have several mass storage devices installed on the computer, you can additionally select one of the mass storage devices you want to monitor in the *Disks* field.

In Settings, you can also specify whether to launch Stacer at system start time by sliding the *Autostart Stacer* toggle from red to green.

To get a quick overview of the system load, click on the *Resources* button (eighth from the top). Much like in the KDE system monitor, you will receive continuous information about the status of the components in question.

The software not only provides information on the CPU load, memory requirements, and data transfer on the network, but also data on the average total utilization of the processors, the filesystems that exist on the mass storage devices, and the read and write operations on them (Figure 2).

The software even identifies high-performance workstations with two physical processors and displays the utilization of the individual cores in the graphic as separate curves.

The *Startup Apps* button (second from the top) lets you check which programs are enabled by Autostart when the computer boots. You can (de)activate autostart for a program using sliders to the right of each entry in the table. The *Add Startup App* button, bottom right, allows you to add other applications to the list and enable them for automatic startup in a separate dialog.

You can edit existing entries in the autostart list by clicking on the pencil icon. Data input is handled in a separate dialog. If you want to completely remove programs from the list, press the "X" symbol and delete the entry (Figure 3).

You can manage system services in a similar way by pressing the *Services* button (fifth from the top). The dialog that then appears shows you a table with the available services. There are two sliders to the right of each entry. The first one determines whether that service becomes active when the computer boots. The second one defines the current state of the service. You can switch off running services at any time by sliding the toggle button, as well as switch them back on if needed (Figure 4).

Figure 2: The Resources dialog provides detailed information based on a comprehensive set of parameters.

Stacer also dedicates a separate dialog to the existing processes on the system via the *Processes* button (sixth from the top). This dialog shows the active processes in the form of a table including RAM usage and CPU load.

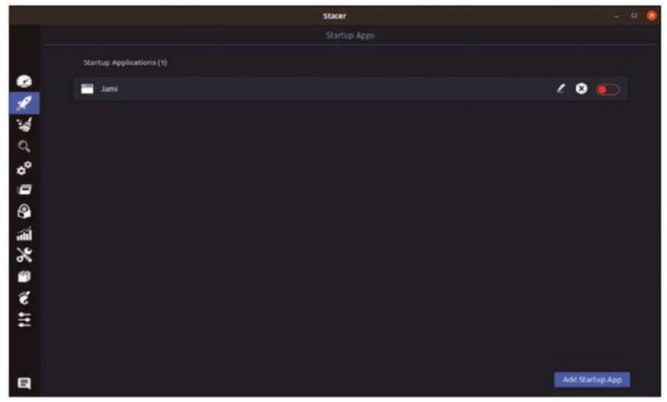
Right-click on a process and then press the blue *End Process* button (bottom right) to deactivate the service. If necessary, you can use the search field to filter the list, which can contain more than 100 processes, depending on the intended use of the system.

Note that killing a process in this list can have a massive impact on the overall system. Only kill services if you are fully aware of the consequences.

Uninstalling

If you want to uninstall packages, you don't have to use the package manager's graphical front end. Clicking on the *Uninstaller* button (seventh from the top) displays a table with installed packages. This table contains Snap packages, but not Flatpaks.

Figure 3: The Startup Apps dialog lets you define which applications are automatically activated at system startup.



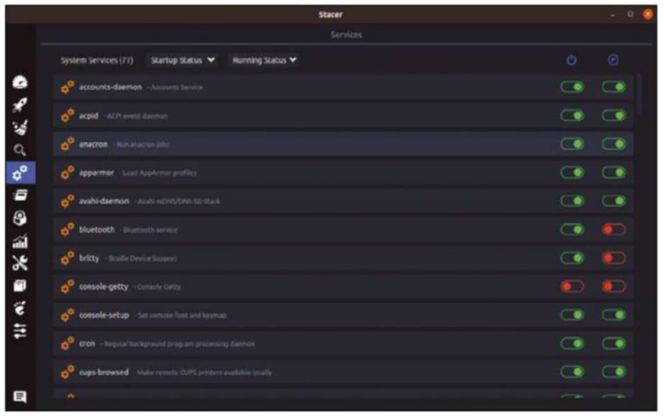


Figure 4: System services can be conveniently switched on or off using the toggle.

Select the package you want to remove from the system using the search field if necessary. Then click the checkbox button to the left of the package. At first, this button appears grayed out, but it changes to green when enabled. Once you have selected all the packages you want to uninstall, press the *Uninstall Selected* button at the bottom of the window.

After authenticating as an authorized user, Stacer deletes the selected applications. This works for distributions with deb or RPM package management, as well as Arch Linux and its derivatives.

Cleanup

Like other operating systems, every Linux installation accumulates an increasing amount of obsolete data over time: logs, cached data, temporary files, crash reports, and more. Tools such as BleachBit [2] erase this data in one fell swoop.

Stacer also offers an intuitive interface for removing the data. Click on the *System Cleaner* button (third from the top) to open a dialog with five categories, each containing files to be deleted. In each category, you will find a grayed out checkbox button, which turns green when clicked.

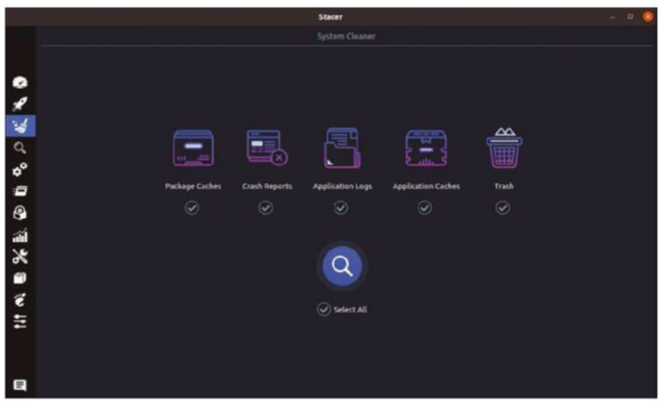


Figure 5: Stacer deletes unneeded data from a Linux installation on request.

This tells the software to delete the data from this category, following a prompt. If you want to delete the files of all categories in a single action, click on the checkbox button to the left of *Select All* (Figure 5) instead. After a further click on the large blue magnifying glass icon, the software switches to a tabular view in which it displays the freed-up storage space sorted by category.

After another click on the *Select All* button in the tabular view (or after selecting individual categories), click on the broom icon at the bottom of the window. After authenticating, the software deletes the data.

Note that Stacer does not make the data unreadable by overwriting the space they occupied, but only removes the corresponding pointers. Tools such as PhotoRec [3] would let you restore it if needed.

In the same window, the program then shows you the volume of data it deleted. Note that Stacer, unlike BleachBit for example, does not let you sort the data to be deleted by application. For example, you cannot delete the caches of one individual application but keep other cache files.

Fast Access

Stacer tucks itself away in the system tray with a speedometer icon. On closing, it asks if you want to keep it enabled in the system tray or if you want to close the application completely.

If Stacer remains in the system tray, you can right-click on the icon to open a context menu containing the individual buttonbar categories. Clicking on one of the categories opens the corresponding dialog. This gives you quick access to the software's functions at any time.

Conclusions

Stacer significantly simplifies the configuration and maintenance of a Linux system. The software reliably displays the most important parameters of the system, which it also updates continuously. The program combines various smaller tools, which are often found in different menus of the Linux desktop, in a uniform graphical interface that offers the same control paradigm in all cases.

Since the software is not limited to specific distributions, it is especially useful for newcomers and generally anybody who is not familiar with Linux. However, always be aware that disabling and deleting system services will impact the stability of the entire system. Caution is advisable. ■■■

Info

[1] Stacer: *https://github.com/oguzhaninan/Stacer*

[2] BleachBit: *https://www.bleachbit.org/*

[3] PhotoRec: *https://www.cgsecurity.org/wiki/PhotoRe*

Create appealing presentations with Inkscape and JessyInk

Your Feature Presentation

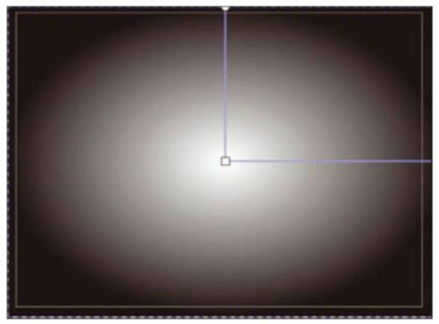
Using Inkscape with the JessyInk add-on helps you to create graphically appealing presentations that can be run in a web browser and are indexable by search engines.

BY SIRKO KEMTER

Giving presentations is a regular part of life at many jobs. For this task, Linux fans often choose LibreOffice Impress, which is installed by default in most distributions. In many cases, this will meet your needs and be relatively easy to use. But what if you want more than the options available in LibreOffice? The potential alternatives include LaTeX Beamer [1] or Impress.js [2]. However, these do not rely on the WYSIWYG principle, and they require some know-how. For this author, there is only one solution. I design my talks with my favorite graphics program, Inkscape [3].

Since the Inkscape software stores files in SVG format, they can be easily viewed in a web browser. To turn a collection of SVGs into a presentation with the usual functions, a little JavaScript is all you need. This may sound complicated, but it is not. Inkscape comes with an extension called JessyInk that is designed for precisely this purpose. Using this extension has other positive aspects. If you want, you can simply upload the presentation to a web server and present it at any location, provided you have Internet access. Search engines can index the text, and the slides can be reused. It's easy once you have familiarized yourself with the workflow, which is very different from that in office suites, and the results will make your presentation stand out from the PowerPoint crowd.

Figure 1: Create a simple background by using the gradient tool.



Master Slide

In this article, I'll show how to create a presentation using Inkscape, starting with how to create a master slide with a custom background. Any task in Inkscape always starts with the document settings. Since many projectors do not support the widescreen format, it is best to select the 4:3 format by opening *Document Properties*... (Ctrl+Shift+D) in the File menu and entering *1024* for *Width* and *768* in the field for *Height*. For *Units*, select the *px* option. Check the *Border on top of drawing* box. This option ensures that the document border is always displayed above the drawing. The border shadow, on the other hand, irritates many users, so it is best to turn it off.

Now select the *Create Rectangles and Squares* tool (F4 or *R*) and draw a rectangle that runs across the entire document. It can protrude a little on all sides. Since Inkscape always draws with the settings last selected for fill and outline, you might not see a rectangle now – don't panic, it's there.

The next step is to fill the rectangle with a color. The easiest way to do this is to click on the corresponding color on the palette at the bottom of the program window. For an appealing design and good contrast, select *90% gray* right next to black. Since monochrome areas are boring, let's convert the fill to a gradient.

Press Ctrl+F1 to activate the gradient tool and then select a radial gradient in the tool settings, which are now visible. A simple double-click on the rectangle applies the fill using the alpha channel. Use the Shift+R shortcut to reverse the fill process (Figure 1).

Now click on the small circle of the gradient display. When enabled, the color of the element changes from white to blue. Then you can choose a color again – in the example I have used *70% gray*. This makes the gradient fade into the background, but the area looks more lively (Figure 2).

Background

For the final touch, use Ctrl+I to import the Inkscape logo [4]. Beginning with version 1.0, when importing SVG files, the program displays a dialog that shows how Inkscape will process the imported graphic. You don't want to link or embed the SVG, but instead include the graphic as an editable object in the current file. It can then be modified directly.

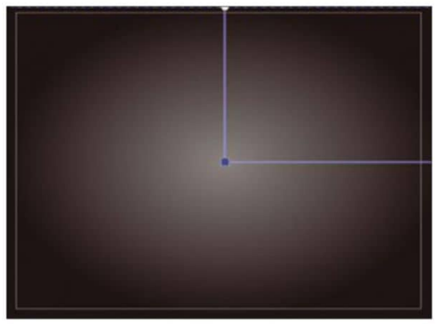
The graphic always appears at the mouse pointer position when imported into the document, and elements of the graphic are grouped. Once the logo has made it into your rectangle, move it to a free space on the drawing area. You can use Ctrl+Shift+G to ungroup the objects, creating three path objects. In the next step you want to select the two white objects. To do this, press F1 to activate the selection tool and click on the two objects one after the other while holding down the Shift key.

Use Ctrl++ (a plus sign) to combine the two path objects into one; they will disappear in the background. Then use Shift to select the remaining path object and press Ctrl+- (a minus sign) to perform the difference operation. The white path object is thus subtracted from the black one; the previously white parts now appear transparent (Figure 3). These Boolean operations are needed time and time again when working with Inkscape, so you need to understand them.

Now drag the logo onto the document and resize it. Scale the logo uniformly to the full size of the workspace. To scale it proportionally, hold down Ctrl. To make the logo a little more subtle, adjust the visibility of the object.

Inkscape offers two ways to do this: via the visibility of the object, which can also be applied to groups, or via the fill color's alpha channel. In our case, both ways work, but it is recommended to work with object visibility only. You should only use the alpha channel when you really need it, such as for gradients with transparency.

Figure 2: With a few tweaks to the gradient, you end up with a background that is unobtrusive but more lively than a solid-colored slide.



To reduce the object transparency, you will find a field with the identifier *O:* at the bottom left of the program window. You can adjust the percentage value to between *20* and *30*.

This concludes the work on the master slide. Save the graphic and check the results in a browser. It renders the graphic starting at the top left of the window without scaling or centering it to the actual size. JavaScript will do this for us later on, as soon as you convert the graphic into a presentation with JessyInk.

JessyInk

To convert the graphic, go to the Inkscape *Extensions | JessyInk | Install/Update*... menu and start the integration of the JavaScript module by clicking on *Apply*. The tool may display warnings, but you can usually ignore them.

Then save the document again and call the SVG file in the web browser again. This will now scale the graphic to the size of the browser window and display it in the center. But that is not all: Press *I* and JessyInk shows all slides in an overview. Press *D* to activate a drawing mode that lets you paint on the graphic.

Back to the master slide. The task is to convert the background we just created into a master. To do this, open the Layers dialog via *Layer | Layers*... and rename the existing *Layer 1* to *master_slide*. After that, you call the JessyInk extension again, but now select the *Master slide*... option from the menu. In the dialog, enter the name of the layer again and accept the change by clicking on *Apply*. You have now created your master slide, and it will automatically appear on all other slides in the background. To avoid accidentally editing the layer, lock it by clicking on the padlock icon.

JessyInk supports two types of presentations: the page-based presentation, which you will be familiar with from LibreOffice Impress or Microsoft Office, and the view-based presentation, which I will discuss in more detail later.

Figure 3: The imported Inkscape logo: The white snowcap becomes transparent thanks to a Boolean operation.

To create a new slide, you only need to add a layer in the Layers dialog. The order determines the place in the presentation. For this example, I will now create another layer from the Layers dialog, which will sit on top of the master slide.

Autotext

JessyInk can also create the page title with autotext. To do this, the program uses the name of the layer. For the example, name the new layer *Inkscape*. In the document, you now need a placeholder for the autotext in the form of a simple text object. Press F8 to select the text tool and click on the center of the drawing area. Afterwards enter some arbitrary text, JessyInk will replace it completely later.

JessyInk only fills the characters later on, the formatting of the text is preserved. So you can set the desired font immediately. This can be done via the toolbar or more conveniently via the text dialog (Ctrl+Shift+T); in the example, I use the Bebas Neue [5] font. For this example, set the text alignment to *centered*. For optimal contrast, select white as the fill color. Now all you have to do is align the text or placeholder. To do this, use the *Align and Distribute* tool Ctrl+Shift+A and center the placeholder horizontally and vertically relative to the document.

Now select the placeholder text again and call the *Auto text...* option from the *Extensions | JessyInk* menu. In the dialog, select *Slide title*. After clicking *Apply*, nothing happens in Inkscape itself. Don't be surprised, JessyInk only replaces the text when the presentation is opened in the browser; in Inkscape itself, everything stays the same. The page number can be inserted in the same way. You have now designed the first slide.

I will now create another layer on top of the current one and name it *Multitalented*. To keep the content of the previous slide from distracting you while you continue working, hide the *Inkscape* layer by clicking on the eye symbol. Again, add a placeholder for the headline, but now make it a little smaller and, instead of centering, place it in the top-left corner, near the document border.

For the slides with content, left justify the title text. I'm using Roboto [6] as the font for the slide text. For further slides, copy these placeholders to the next slide layer using Ctrl+C and Ctrl+Alt+V. Pressing the Alt key forces Inkscape to insert the object at the same coordinates. If you use Ctrl+V instead, Inkscape drops it at the current cursor position.

Contents

Next, import the pocket knife graphic (this is some of my work that you can use freely). You will find this graphic in the Openclipart library [7], a project once started by the Inkscape developers that collects and provides freely usable graphics. Scale the graphic so that it takes up about half of the document. Then open the text dialog and add the text from Figure 4.

Scaling adapts the size of the text to the available space. However, Inkscape lacks the features of a word processor or a full-blown desktop publishing program. It processes text like a graphic element. For example, there is no way to insert bullets. However, you can easily compensate for this shortcoming by drawing your own bullets.

You could now adjust the text object with the usual text tools. However, if you want to display the text line by line later on, you will need each line to be a separate object. To do this, use *Extensions | Text | Split text...* and split the object line by line. Then select the first individual text object and call up the *Extensions JessyInk Effects...* option. The dialog offers three different effects: *Appear*, *Fade in*, and *Pop*. The number indicates the order in which the objects are displayed.

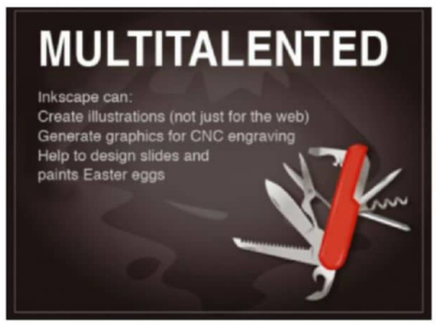
Now only the transitions between the individual slides are missing. Again JessyInk offers different possibilities. You can open the dialog in *Extensions | JessyInk | Transitions...*, and if you want to use one transition for all slides, simply apply the selection to the master slide. The program allows different transitions when fading in and out of a slide.

That's it! The page-based presentation is finished. After saving, open the SVG file in a browser like Firefox. Use the spacebar to progress through the slides step by step (in Inkscape these are the layers). Alternatively, use the arrow keys to scroll through the slides. Pressing / gives you an overview of all the presentation slides that you created.

Views

The only thing missing now is the previously-mentioned, view-based presentation. This is a single large graphic of which you then display selected details – views – in your talk. The size of the graphic itself is irrelevant, and that's what makes this option so interesting. For example, it can be a good option if you need to present a larger network plan or an extensive mind map from the last team meeting. With conventional presentation programs, you would have to use a graphics program and split the graphic into individual images. This requires a great deal of time, and you would also have to update the exported individual graphics whenever changes were made.

Figure 4: Here are the results after inserting the text and importing the pocket knife graphic.



For this example, I will simply create a new layer and import the icon overview offered on Openclipart [7]. Use the *Align* tool to center the graphic on the slide (Figure 5). After ungrouping, select the black background and delete it with the Delete key.

After importing, the icons are still very small and are therefore difficult to see on the slide. However, with the help of the View menu item, any individual icon can easily be enlarged during the presentation. To do this, simply take the *Create Rectangles and Squares* (F4) tool and draw a rectangle that is large enough to fit one of the icons. The square's fill and the contour are not important. To be able to see the background more clearly, it makes sense to reduce the object's visibility or only use an outline (Figure 6).

Once you have scaled the rectangle to the correct size on the page, you only need to duplicate it with Ctrl+D as often as you need to create individual views (details that you want to zoom in on). Then drag the copies onto the individual icons of the overview. Finally, call up *Extensions | JessyInk | View...* and number the view elements of the slide under *Order*. If you start at *1*, the presentation first shows the complete slide and then goes to individual details. But if you start at *0*, JessyInk will shift to the large format with the very first icon.

After saving, reload the presentation in the browser and watch the smooth transition as you switch from one view to the next. Interesting effects can be obtained by rotating the rectangle or changing its orientation (e.g., via *Object | Flip Horizontal*). JessyInk then adopts the alignment into the presentation. It is best to experiment a little with this function.

Conclusions

As our example demonstrates, creating presentations with Inkscape isn't that difficult once you've mastered the drawing features.

However, there are some minor downsides: For example, to display fonts correctly, they have to be available on the presentation computer. If in doubt, you should therefore use standard fonts.

You could also use Google fonts. However, Inkscape does not yet have the ability to add CSS, so this would have to be done manually. Optionally, you could also convert the text objects into paths. However, this would mean sacrificing index capability, and you would no longer be able to edit the text with the text tool. ■■■

Figure 5: The imported icons with the opaque background.

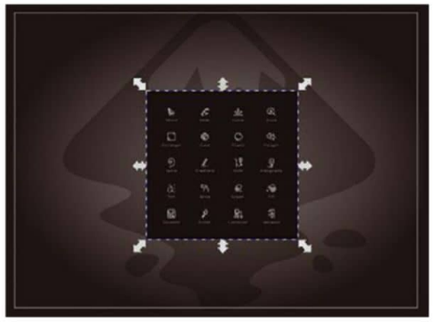


Figure 6: JessyInk uses the content in the rectangle as a view.

Info

[1] LaTeX Beamer: *https://en.wikipedia.org/wiki/Beamer_(LaTeX)*

[2] Impress.js: *https://impress.js.org/#/bored*

[3] Inkscape: *https://inkscape.org*

[4] Inkscape logo: *https://upload.wikimedia.org/wikipedia/commons/4/4a/Inkscape.logo.svg*

[5] Bebas Neue: *https://fonts.google.com/specimen/Bebas+Neue*

[6] Roboto: *https://fonts.google.com/specimen/Roboto*

[7] Pocket knife: *https://openclipart.org/download/129409/1301233178.svg*

[6] Openclipart: *https://openclipart.org*

[7] Inkscape tools icon set: *https://openclipart.org/detail/265449/inkscape-tools-icon-set*

The Author

Sirko Kemter has been using Inkscape for many years and is the author of a book on Inkscape that was published in German. In his spare time, he creates graphics for Openclipart and various open source projects.

FOSSPicks

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

After years of foisting Linux on family members, their response after Graham recently upgraded their machines to run KDE Plasma 5.20 has been overwhelmingly positive. Great work Plasma team! BY GRAHAM MORRISON

Spatial data environment

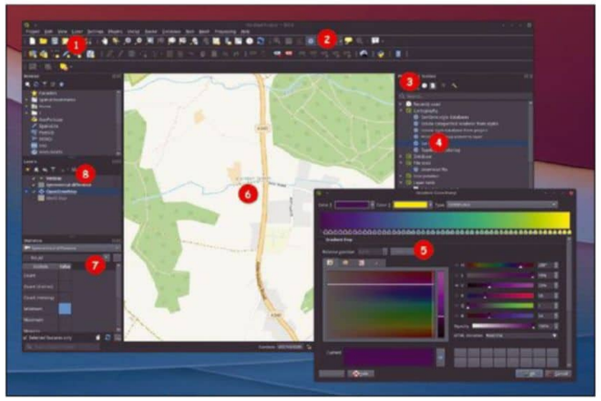
QGIS

GIS is an acronym for geographic information system, and those three letters are often applied to software that can sort, visualize, edit, analyze, and manage positional data, such as the data used to create maps. But it's also a subject that can be as hugely complex and as unresolved as the surface of the Earth, Olympus Mons on Mars, or a tesseract in both three- and four-dimensional space. QGIS tries to help with this. It's a Qt-based application that attempts to tame this complexity by making the software free and open source and keeping it as accessible and easy to use as possible. It accomplishes this while still providing enough functionality to allow for serious research and the analysis of whatever data you throw at it.

When you first launch the application, the main window can still be overwhelming. It's full of buttons, toolbars, panels, and menu items, and it's difficult to know where to start. It gets easier when you remember that the main goal here is mapping geographic data onto your screen. That means lots of these options relate to coordinates, projection modes, and analysis and also that the first step should be loading some data. Sourcing data can be difficult, especially if it's the result of specialist research and you'd like to enjoy the same rights over the data as you do the QGIS source code. However, there's a good starting point built into the application itself, and that's OpenStreetMap. This can be found by clicking *XYZ Tiles* in the browser on the left. Depending on your project mode, when selected it can appear just like an OpenStreetMap browser on the web. Except, in this case, QGIS is interpreting the raw data through its own algorithms to create the view, and you have complete control over these algorithms and how things appear.

As you might expect, QGIS can ingest all kinds of data, including spatially-enabled tables, PostGIS, SpatiaLite, MS SQL Spatial, and Oracle Spatial formats, as well as many spatial-specific vector formats. The project hosts some of its own examples, and you can edit many different file formats or import data from your phone if you have an app that generates GPX files from the integrated GPS device.

There's even an initiative to bring point cloud data into the application, which is undoubtedly the next generation of spatial data now that Apple is shipping LiDAR on its phones and tablets. With the data imported, there's equal variety in the ways the resulting visual maps can be composed and explored. The menus are full of reports and measurements you can run, from the distance between two points or a distance matrix to the parts that are likely to be hillsides. Many of these reports are HTML files you can easily share with collaborators, but you can also publish your maps using WMS, WMTS, WMS-C, or WFS formats, as well as their associated servers. The end result is still an overwhelming application, but one that can literally help you explore the world of geospatial data. Thanks to the Python console and plugin system, you can take this as far as you need to, and the analysis scripts used by the application can easily be expanded or modified to suit your needs.



1. Package browser: Data is key in QGIS, and there are many ways to import pre-existing sets. **2. Measurements:** Check distances and add annotations to your own maps. **3. Processing:** There are many modules for running reports and transformations, plus it's easy to write your own in Python. **4. Terrain shading:** Run various terrain analysis modules to automatically color topography. **5. Gradient control:** Make your neighborhood the color of Mars. **6. Data view:** While QGIS is undoubtedly useful for map makers, the data doesn't have to correspond to geography. **7. Statistics:** See the exact data for the area you're working on. **8. Layers:** Load multiple datasets and view them all at the same time.

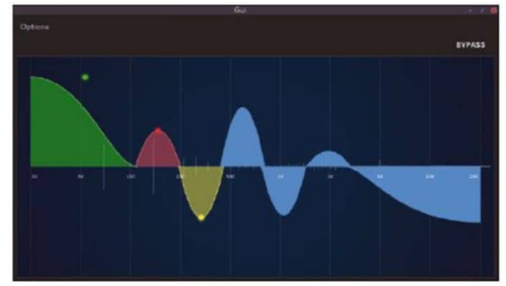
Project Website
https://qgis.org/

Audio equalizer

PrettyEQ

Now that many of us spend too much time in one video conference or another, it's become a cliche, but sound quality really is everything – and when you're in a meeting with 10 other remotely connected people you really start to notice it. Listening to people with poor audio quality requires significantly more concentration. If you have more than a couple of meetings a day, it becomes much more mentally taxing than the equivalent in-person meetings would be. Unfortunately, there's very little you can do to persuade other people to improve their recording setups. The issue often becomes too personal. Because they don't hear their own voices anyway, it may seem superficial (to them, at least). But there are things you can try to help improve the audio quality as it passes through your system on its way to your ears, and one of the better things to try is it, PrettyEQ.

The EQ in PrettyEQ is short for equalization. If you remember those amazing '80s-era tower music systems, you'll remember their huge equalization sections. These were usually in the middle of the unit, with separate vertical sliders for both the left and right channel, and they allowed you to increase or decrease the levels of specific frequency bands in the audio spectrum. Most of us used these to simply ramp up the bass and high-frequency part of the audio, but the humble equalizer is an incredibly useful audio tool for fixing all kinds of problems, including the terrible audio from colleagues in a video conference call. By default, PrettyEQ plugs itself into PulseAudio so that everything your computer produces goes through the effect. You can roll-off frequencies above 20kHz and below 20Hz and modify five variable bands to increase and decrease specific frequencies, which is perfect for removing whistling microphones, the sound of kitchen appliances, and other annoying background noise.



Alongside the adjustable equalization bands, the UI maps input frequencies so you can both hear and see the effects of your tweaking.

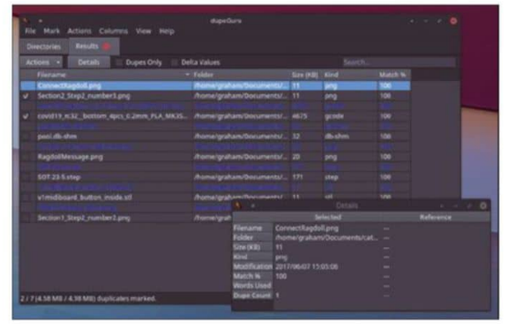
Project Website
https://github.com/keur/prettyeq

Duplicate file finder

dupeGuru

Despite all the power and modularity of the command line, there are a few simple tasks that can prove difficult to perform using this interface. One of these is finding and removing duplicate files. There are various approaches you can take to finding duplicates – from calculating checksums, file sizes, and MD5 checksums, to simply comparing filenames. But none of these gives you the assurance you need when you want to permanently delete files and are anything less than confident about the action. This situation is one of the few times when a graphical desktop application can be a better choice, and dupeGuru is a tool built specifically for the job. It's also a mature application with good support for multiple languages and even multiple operating systems, which means any duplicates you find can be confidently deleted.

The simple Qt-rendered window is easy to understand and navigate. You add folders or files to the main list view and then choose the scanning mode. Alongside a standard file comparison, there are special modes for both audio and images. These both use context-specific comparisons, such as metadata comparisons and even a fuzzy image search for images that are similar but not the same. These tools are exactly what you need when you want to delete lower bitrate audio copies or image thumbnails, both of which are difficult to find any other way. The results list is ordered by filename, with columns for the size of each duplicate and for the percentage confidence in the comparison. You can see the delta of any differences, view image files directly, select only the duplicates, and choose to move them to a different folder. This is all the functionality you need, and it enables you to quickly and confidently remove the many duplicates most of us have littering our home directories.



File scanning does take time, but you can confidently identify and then delete duplicate files.

Project Website
https://github.com/arsenetar/dupeguru/

Disk usage monitor

duf

It seems a little strange to be worrying about disk space in the 21st century. Storage is now cheap, and most of us can afford SSDs or spinning disks with enough storage to save the things we want immediate access to. For everything else, there's Nextcloud or a LAN-based NAS. But like CPU cores and RAM, it seems whatever capacity we have, we use. We still run out of processing power, memory, and storage. And when it's storage, the first command many of us reach for is the humble `df` (disk free). Type `df` on almost any machine, even something running macOS, and you're presented with a table showing every mounted storage device, the number of blocks they contain, which blocks are being used and which are available, and the percentage of used capacity. Most of us add the `-h` flag to turn the large byte-based numbers into something more easily understandable such as megabytes and gigabytes. But other than that, it's a simple command that does what it needs to do with very few flourishes.

Here is where `duf` can be appealing. It's a command that obviously takes its inspiration from the humble `df`, but it succeeds in making the same information much more presentable and eye-catching. `df`'s table output is transformed into a multicolor view with multiple panels for each type of device: local, network, and special, which includes local mount points such as `dev` and `tmp`. Additional arguments can be used to see more, including pseudo, duplicate, and inaccessible filesystems, and the `hide` command can be used to see less. The colors in the output are important, because they immediately communicate when a device is running out of space, regardless of its type or size. Green means there's plenty of capacity, amber means a reasonable amount has been used, and red means you're getting close to the limit.

Check your storage capacity in style with the colorful and completely configurable `duf`.

Project Website

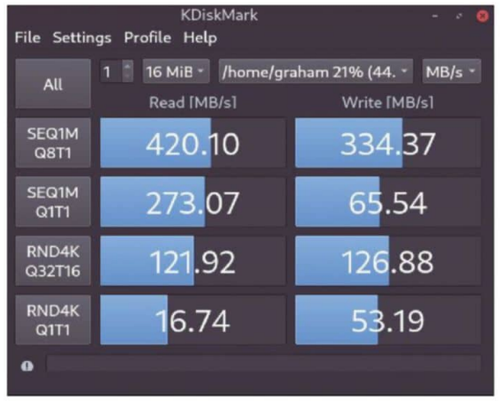
https://github.com/muesli/duf

Disk usage monitor

KDiskMark

As great as `duf` is for checking on available storage space from the command line (see above), it can't tell you how fast your drives are. This is where the beautifully designed KDiskMark can help. This is a simple desktop application that produces benchmarks for whatever storage you have attached to your machine. The main view shows a simple table with whatever devices you have mounted listed on the left. To the right of these are read and write speeds for each device, rendered as both a floating-point number and a histogram across the cell. By default, the units are megabytes per second, but this can easily be changed. The best thing about KDiskMark is that it's very easy to see the relative performance of each device, because the histograms give you a quick overview.

If you need to dive into the details, you can change the block size, queues, and thread count for each test that the application runs on each device. The cells in the table will update accordingly. The back end for the test is Flexible I/O Tester, also known as fio, a separate project with a focus on storage performance. KDiskMark offers performance profiles, which include peak throughput, estimated real-world performance, and a mix between them, all encased within a Qt/KDE-themed window. If you want to return to the world of text formatting, you can even generate a text report. This includes all the same values you can see in the UI, but more readily enables you to track changes over time, perhaps by committing them to your own Git repository and using `diff` to perform the comparison. It's a simple application, executed wonderfully, that can lead to quite complex insights into the performance of your system.



Click on the *All* button to instantiate the benchmark creation for every connected storage device.

Project Website

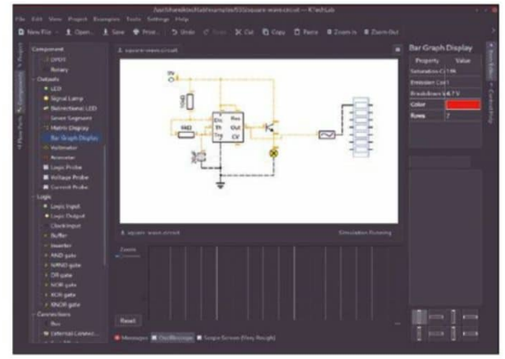
https://github.com/JonMagon/KDiskMark

Circuit designer

KTechLab

While designing circuits and electronic components can be a difficult and specialized subject, there's never been a better time to try. Not only is there a huge amount of learning material available online (and in the archives of this magazine), you can also routinely order your own circuits to be printed and sent to you cheaply and quickly. Best of all, there's lots of open source software that can help. Just recently, for example, we've looked at two brilliant applications for circuit design, QElectroTech and Horizon EDA, and here we're looking at a third, the rebirth of an old KDE stalwart called KTechLab. This is an application that was first developed in 2007 and updated until version 0.3.7, before entering a long period of stasis. However, a new development team and a recent port to Qt 5 and KDE Frameworks 5 means the project is once again alive and has been released as version 0.50.0.

KTechLab is best described as an IDE for circuit design, and it feels a lot like its KDE stablemate, KDevelop. Instead of dealing purely with code, however, KTechLab's primary elements are electrical components. By default, the main display is split into three sections. On the left are the project, component, and parts lists, which can be used to drag components such as resistors, capacitors, and switches into the middle area, which is a canvas for your circuits. The right-hand panel is an inspector to display and edit component characteristics. You can do this as you drag components into your circuit and join them up. A lower panel is the equivalent to KDevelop's debug and console views, replaced with an oscilloscope and message output. These are functional because KTechLab's best feature is that it can simulate the circuit you're building with voltage output over time mapped to the lower panel, which is a brilliant way to learn circuits!



KTechLab lets you design electrical circuits and can even simulate what would happen if you were to feed them electricity.

Project Website

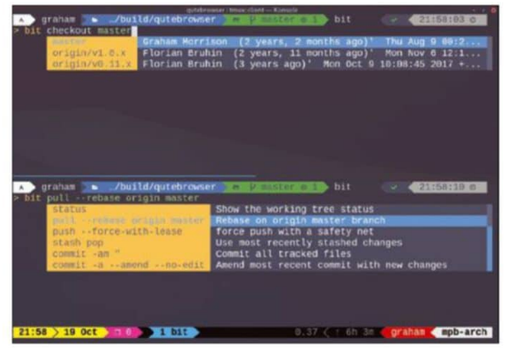
https://github.com/ktechlab/

Git client

bit

Git, the version control system that's used to manage the Linux kernel source code and a million other code projects, is an amazing tool. It's decentralized, powerful, and relatively easy to get started with. For these reasons, more and more people are using it for things it was never designed for, such as managing configuration files and PGP keys or even handling changes in documentation and edits to Markdown documents. But there's a serious problem with Git; it can quickly become overly complicated. When you're not intimately versed in whatever specific incantation it requires, it can leave your local repositories in a mess. What's needed is a Git helper that isn't a GUI and doesn't swamp instant responsiveness of the Git command line. This is exactly what `bit` attempts to do.

There are a few alternatives to Git that attempt to make it a little more manageable, but `bit` is the easiest we've found to use. It uses a pop-up menu system to show you a list of options that are context-sensitive to your current location and repository state. Type `bit` in a repository, and you're immediately presented with a list of Git arguments, commands, and flags, complete with a brief explanation of what each does. The really clever thing is that this menu isn't statically generated but is instead a list of commands populated by the context of your directory and repository state. You'll see your own branches, aliases, and commits, for instance, and pressing return fills out the remainder of the command automatically. There are also a few new commands, such as `bit sync` to synchronize all your changes to the origin branch and `bit save` to save changes to the current branch. It's a brilliant way to get started with Git, and also a great client if you don't use it all that often.



If you love Git but have found it difficult to navigate, `bit` is brilliant.

Project Website

https://github.com/chriswalz/bit

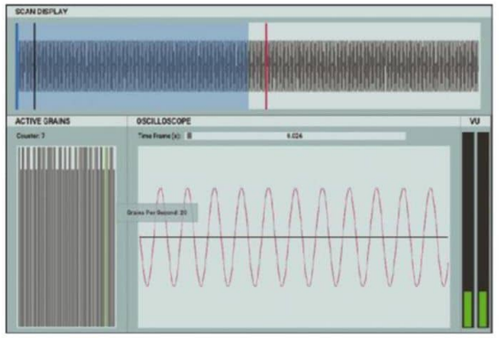
Granular synthesizer

EmissionControl2

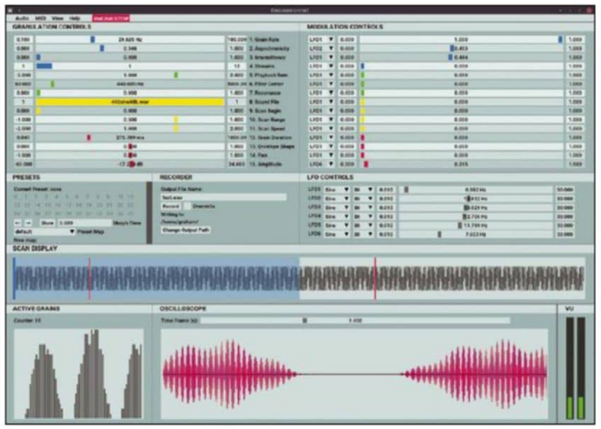
Sound design is an unusual preoccupation. It's seldom used professionally, other than in niche gaming and movie-making scenarios, and even popular music most commonly uses pre-baked sound presets, sound libraries, and default loops from applications. With the exception of people like Brian Eno, the artists and musicians themselves will typically have done very little sound design. And yet, there are many of us with an innate fascination in the qualities of sound, who want to create unique sounding audio, regardless of whether it's destined for the avant garde or the garderobe. EmissionControl2, developed by a group calling themselves the Center for Research in Electronic Art Technology (CREATE), has been made for us.

EmissionControl has been around for a long time. The original version was PowerPC-only, running on a Mac, but version 2 marked a proper open source release and, subsequently, Linux builds alongside those for macOS. This is a great thing because EmissionControl2 is a slightly crazy and utterly unique sound generator for people who want to experiment with sound. It generates audio in real time from a series of audio files using something called granular synthesis. Granular synthesis is the audio equivalent of copying a group of closely positioned pixels in a video, mixing up their relative locations, changing their color, and blasting them onto a blank canvas. The output typically retains some flavor of the input audio files whilst generating something completely different. It can sound low-fi and gritty, with a seemingly low bitrate or sample rate, but it can also sound soft, organic, and evolving. You can play the sounds and control the UI from a MIDI controller, transposing grains without producing the same sampling artifacts you get with traditional samples. It's a sound that can't be created any other way.

The grains are taken in groups from one or more sound files at a specific "grain rate." The grain rate, like any of the other 14 different parameters, can be modulated by one of the six LFOs accessible on the right side of the UI. These six LFOs run independently of each other, between 0.01Hz and 30Hz (times a second), and they can be either sine, saw, or triangle waves, with noise for random variation. Parameters can control how the grains are panned and amplified, how they're picked up from the audio file, and the size of the window used to source them. Many of these parameters can be seen in an animated waveform preview of the audio file from which the grains are being taken. There's even a filter to change the sound, and all of this can be modulated. Beneath the waveform view, you get to visualize the resultant audio output in both an oscilloscope and an active grain counter. When you're happy with your configuration, you can save everything as a preset and then even morph between the parameters of two separate presets. Of course, none of this would matter if the resultant sound was awful, but it's not. It's both musical and unique at the same time, and while those of us fascinated by sound design will simply enjoy geeking out over the details, pop producers might find something they like even more than their synth presets.



The oscilloscope is powerful too, letting you zoom in as needed to see the individual samples and waveforms.



The best thing about granular synthesis is that the output audio can sound as much or as little like the input audio as you choose.

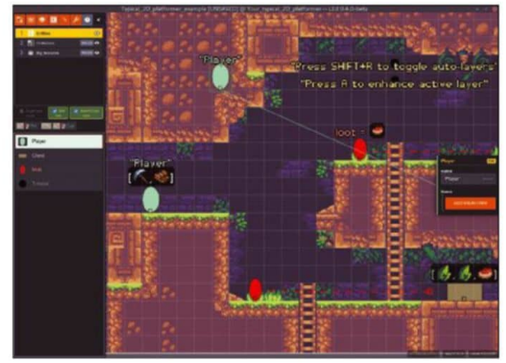
Project Website
https://github.com/EmissionControl2

Level editor

LDtk

One of the best (proprietary) games of recent years is called Dead Cells. It's a beautifully, retro-themed sideways combat oriented platformer. What makes Dead Cells distinctive, besides the Rogue-like pseudo-random level generation, is that your player's character needs to die. A lot. This is because your reincarnated avatar inherits whatever buffs and power-ups you've previously been able to unlock, allowing you to break into new areas and, ultimately, the end game. This is all relevant because one of Dead Cells developers has distilled their experience into a new open source level editor that's been designed to help nascent game developers create the best possible levels for their games. The result is LDtk, a 2D map editor (formerly known as LEd) that hopes to make games designers out of us all.

First and foremost, LDtk attempts to be easy to use. You can drag your mouse across an empty canvas to paint the tiles for your own levels, and after creating a collision map, elements like grass and other textures are added automatically. There's some excellent documentation included within the application, as well as some well-designed example levels to help you through the main concepts. These concepts are mostly to do with the integer grid layers, tile layers, and entity layers, alongside your own definitions of enumerated types for the elements in your game, allowing you to link your graphical level design directly to your code. Levels are saved to a JSON-formatted file, which is very easy to understand and parse within your own games engine. While the editor has been designed to be platform agnostic, it's been written with the Haxe API and integrates wonderfully with the platform.



If you loved the game Dead Cells, one of its main programmers has written an open source level designer to help you write the next indie blockbuster.

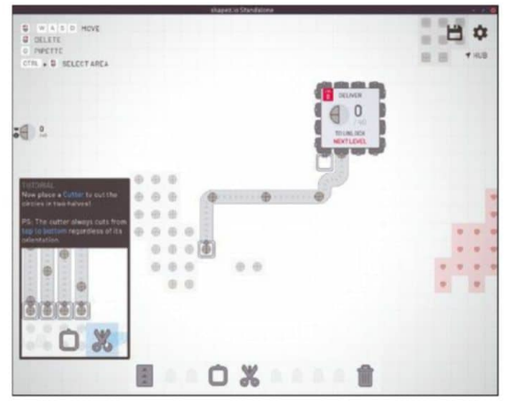
Project Website
https://github.com/deepnight/ldtk

Puzzle game

shapez.io

This is a game that could be called a factory simulator. If you've never seen one before, you might be surprised to learn that they don't typically simulate the factory environment. Instead, they feature the kind of problem solving you need to build an efficient production line. It's a little like some of the more complex levels in the classic game Lemmings. In that game, each lemming has its own speciality, and you solve a problem by placing these lemmings in the right place, often setting off a chain reaction of events with the goal of saving the greatest number of lemmings. In shapez.io, the skilled lemmings are replaced by unlockable components that you drag onto a top-down overview of your factory floor.

You start by joining conveyor belts between the producing and consuming parts of the plant, and there's a helpful tutorial mode to onboard you into the game mechanics as quickly as possible. Tutorial mode is accompanied by onscreen overlays to show you the keyboard controls for each mode you're in, such as navigating the factory floor with *wasd*. The pace of the game is slow and relaxed, and you're often left to your own devices when it comes to efficiency: After solving a particular problem, you might choose to wait and make a cup of tea while the requirement is satisfied, or you could decide to make your production line more efficient, perhaps by parallelizing it or removing inefficient bottlenecks. These will speed up the end of level, but they're often not required. Within no time at all, you will effectively be programming your production lines to process all kinds of demands. I think you will find the process both educational and a lot of fun.



Shapez.io can be played online, through Steam, or with a standalone build for the Linux desktop.

Project Website
https://shapez.io/

Manage your assets with Portfolio Performance
Stock Watcher

Portfolio Performance helps you manage and analyze your investment portfolio.

BY ERIK BÄRWALDT

In an age of ultra-low interest rates that make it impossible for legacy savers to build up assets, more investors are turning to securities and other riskier investments. But stocks and mutual funds will not earn money automatically. Many users prefer active oversight and management of their investment portfolio. Computer-aided portfolio analysis tools let you monitor the performance of your portfolio and make adjustments as necessary to maximize the return.

This kind of investing does not require expensive software subscriptions: All you need is a Linux PC and the Portfolio Performance [1] program. Portfolio Performance tracks your investments and even provides tools for analyzing your portfolio.

The Portfolio Performance application was created in Germany, but it supports a number of different currencies and investment types, so you can use it in other countries as well.

Installation

Pick up Portfolio Performance from the project's website. The tarball weighs in at around 45MB. After downloading, unpack the archive into a directory of your choice and then change to that directory. Then start the program by typing the `./PortfolioPerformance` command at the prompt or by clicking on the file in the file manager.

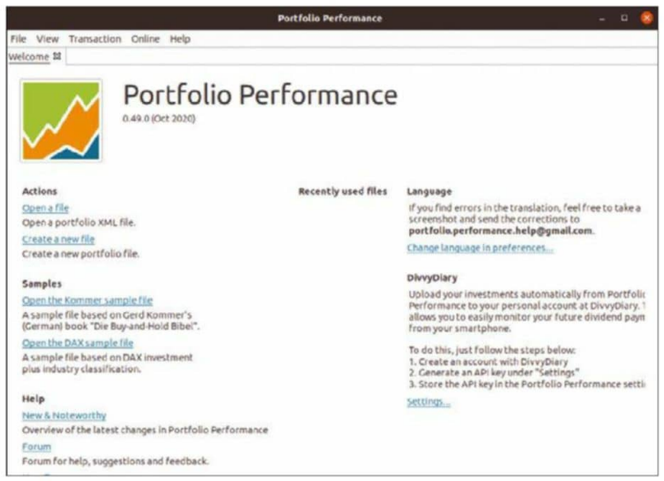


Figure 1: The program window looks a bit spartan when first opened.

To enable integration into the popular desktop environments' menu structures, the developer also provides an XPM file containing a program icon for displaying in a menu or on the desktop. Listing 1 shows an example of the desktop file required for the `/usr/share/applications/portfolio.desktop` menu entry. For the system to find the program and the icon, you will need to adjust the paths in lines 7 and 11.

After launching the software, a fairly spartan program window appears (Figure 1). Except for a small menubar, it has no other controls. In the free area of the welcome window, there are only a couple of links in the *Actions*, *Samples*, and *Help* groups. They are intended to make it easier for users to get started with the application.

First of all, create your own portfolio file by clicking on *File* | *New* top left in the window. The application now opens a wizard in a small superimposed window, in which you select the base currency for the securities account as the first step. In the next step, you create an individual name for the securities account and assign a name for the corresponding clearing account, which is referred to as the *Reference account* in the dialog.

Then press the *Add* button to the right of the input fields. The names of the securities account and the reference account now appear in a table below. If required, you can create additional accounts or switch to the next window by pressing *Next* at the bottom of the dialog. You can then add another account to the program. However, this account is not linked to a securities account, but could be, say, a sight deposit account.

Listing 1: Desktop File

```
01 [Desktop Entry]
02 version=1.0
03 Type=Application
04 Name=Portfolio Performance
05 GenericName=Personal finance
06 Comment=Track your portfolio performance
   (finance)
07 Exec=/<Path>/<to>/PortfolioPerformance %f
08 Terminal=false
09 StartupNotify=false
10 Categories=Office;Finance;
11 Icon=/<usr>/<share>/<icons>/portfolio.xpm
```


The newly added account will also appear in the list of existing accounts further down after you press the *Add* button. This is where you will also find the reference accounts for your securities accounts. After pressing *Next*, the next dialog will let you add existing securities to the securities account. To help you do this, the program shows you a selection of different shares, each of which it lists depending on the current share index. Top left in the wizard, a small selection field shows a popular share index and below that, in the tabular view, the securities listed in this index (Figure 2).

In the last step of the wizard, you classify the shares using a list, which you can use later on in the analysis phase to make the charts more meaningful.

You are then taken to the statement of assets. It contains the *Securities*, *Accounts*, *Reports*, *Taxonomies*, and *General Data* categories on the left side of the window. On the right, you can see context-specific information.

In the *Accounts* section, first update your accounts in the subgroups. This means, above all, entering the current account balance, which the software – if it is a reference account for a securities account – will automatically modify to reflect any new securities investments.

To do this, right-click on the desired account and select the *Add* option from the context menu that appears. The program then opens a dialog where you can enter the required data. Then the account balance appears at the top of the account view in the main window, while the program lists the transactions at the bottom.

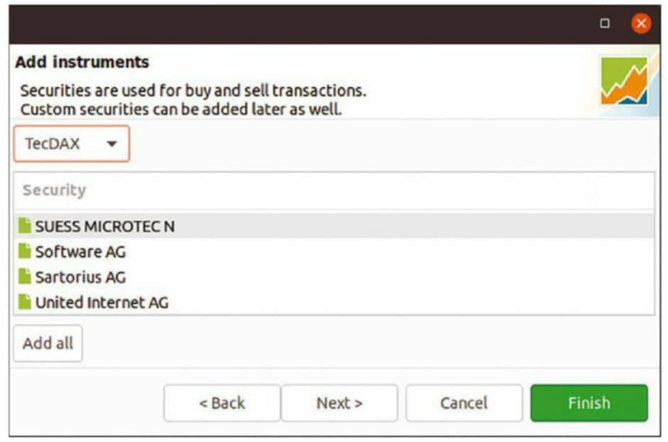


Figure 2: You first need to enter the master data for your securities account in a wizard.

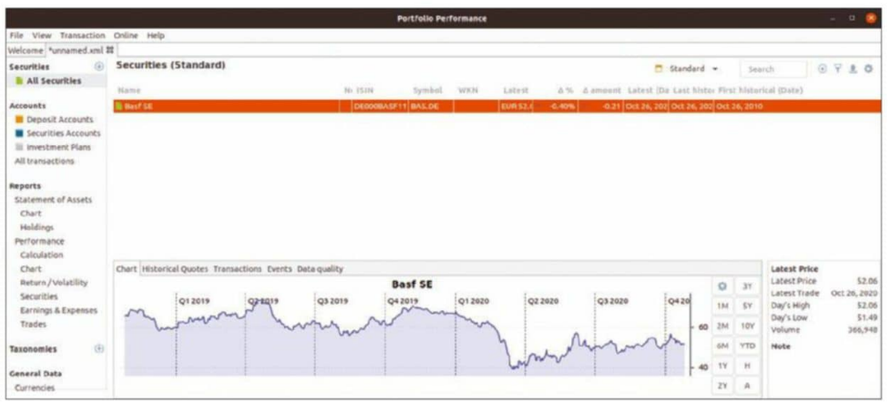
Buy!

If you now want to buy securities or update your stock, open the Securities dialog. If you have already selected some securities in the setup wizard, they will appear on the right side of the program window as soon as you click on the *All securities* subgroup in the *Securities* category.

If you click on one of the values, a chart appears in the lower part of the program window, tracking the share price (Figure 3). By selecting the desired period of time to the right, you can limit the display to a specific time window. On the far right in the window, the program also displays some statistical information. The program retrieves the share price history from the web.

To add new values to the list, press the *Add instrument* button in the top right corner of the program window. If you only know the name of a share, but not the symbol it uses internally, click on the *Search for instruments...* entry in the context menu that appears. Another window opens, and you can enter the name of the security. Then click on the *Search* button to the right of the input field.

Figure 3: Portfolio Performance attractively presents the development of individual shares in a chart.



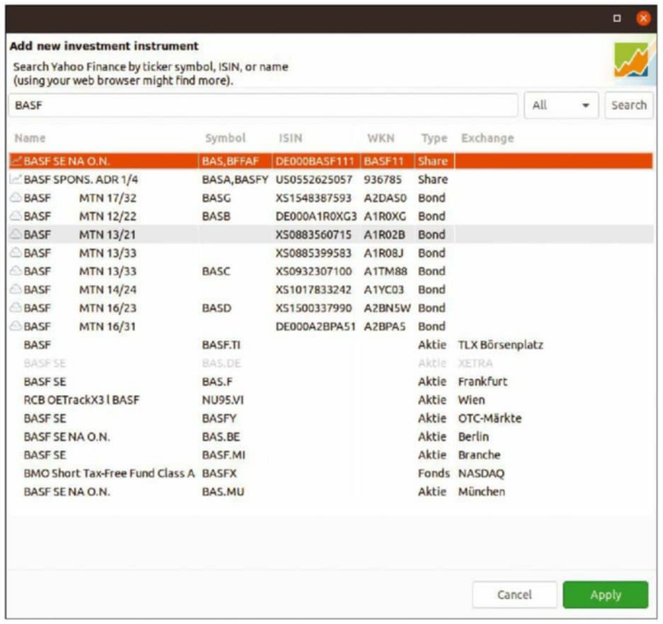


Figure 4: You add new values via a selection dialog. The search supports multiple stock exchanges.

The software now looks for the required information on the web; a longer list can appear containing, say, a separate entry for each trading location (Figure 4). Select the desired entry and then click on *Apply* to transfer the information to the input dialog for the new instrument.

Once you have completed the list, you still have to add the stocks. To do so, select the corresponding share and then right-click to open a context menu in which you select the first option *Buy...*. In a new dialog, you now enter all the required information including the fees and taxes. Make sure the purchase is offset against the correct reference account, which you can modify in a selection field if necessary. After pressing the *Save* button, the inventory is saved and added to the first column of the *Reports | Statement of Assets* list view.

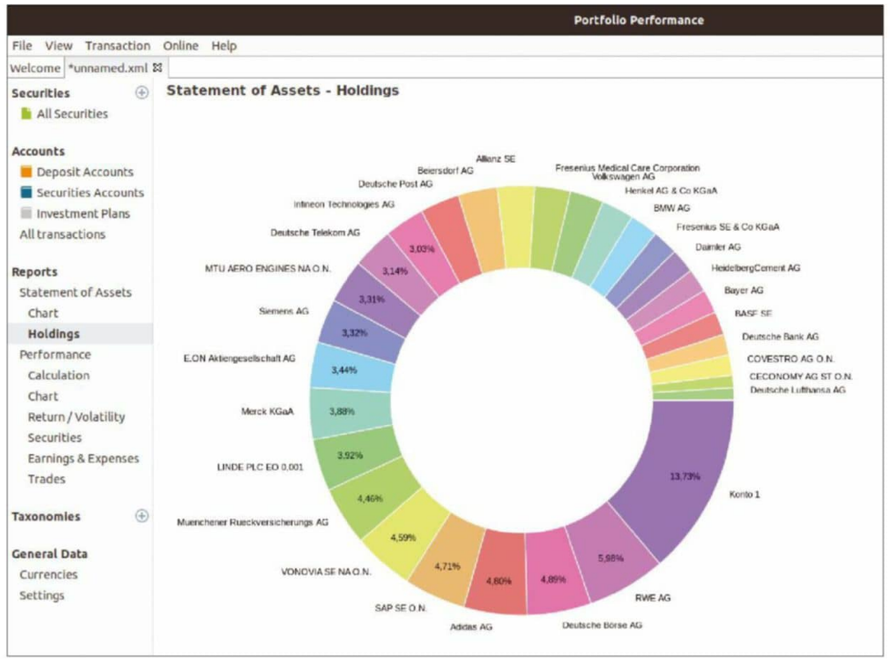
Using these steps, you can manage your entire stock portfolio. The context menu, which can be opened by right-clicking, also lets you enter sales for each value, delete an instrument completely from the listing, and add dividend payments or tax refunds in individual dialogs.

Overview

For an overview of your current asset status, click on the *Reports* group in the navigation area on the left side of the program window. You can call up a tabular view of your portfolio in the *Statement of Assets* category. Securities that are included in the *All Securities* list but do not show any stock information are not included in the assets.

Since a tabular view can quickly become confusing, especially with extensive portfolios, the *Holdings* option gives you the possibility to visualize the portfolio. For this purpose, the application displays all the available shares in a wheel chart by their percentage share in the holding (Figure 5).

Figure 5: The shares in the holding can be displayed graphically in a wheel chart.



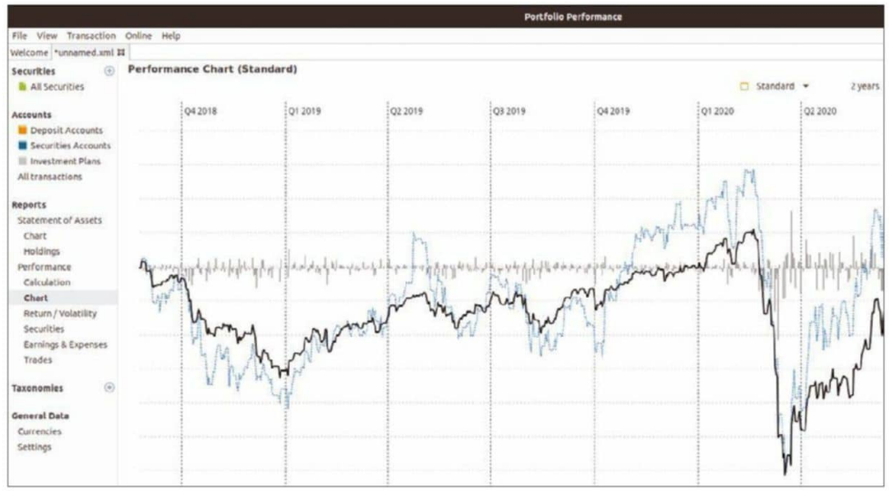


Figure 6: Portfolio Performance visualizes your portfolio's performance as a line graph.

History

In order to determine the performance of a portfolio and also of individual securities, it is extremely important to obtain reliable historical price data. Portfolio Performance integrates a large number of different Internet sources that make such data available. These web services not only let you search for a share during the inventory process, but also form the basis for all historical views. You can view the price history for each security as a table in its individual view in the *All Securities* group in the lower window panel of the *Historical Quotes* tab.

The way the price development is displayed – you will find this on the left side of the *Chart* tab – is also based on historical data. In order to limit the charts in terms of time, you will find several buttons to the right of the graphic display that represent the corresponding time periods to be selected for the display. *6M* covers the last six months, while *10Y*, for example, shows the price development of the current share over the past 10 years.

Based on the same data, the software also displays a line chart in *Reports | Chart*, which – when sorted by different time periods – gives you a clear view of the overall development of your portfolio.

Reporter

In the *Reports* category, you will find detailed information on the performance of your portfolio in several separate views. This is not about drawing pretty charts, but about visualizing popular performance indicators. On the dashboard, which you call up via the *Performance* entry, the program calculates all the key performance indicators for various periods of time that can be set as desired.

If required, you can create additional dashboards with your own criteria. The *Calculation* group lists all your securities and their performance in a table; the desired view period can also be selected in the top right-hand corner of the window. In contrast to this, the *Chart* option shows you a clear-cut line graph for performance evaluation (Figure 6).

In the *Securities* section, you can also view the performance of each individual security. The program lists this in a table by default. But in the lower part of the window the application also displays a line graph for each individual security selected in the upper window segment. Transactions and trades can also be called up in separate tabs specifically for the selected security.

Overview

A complete overview of all your financial investments including paid fees and taxes is provided by the *Earnings & Expenses* view. The dialogs linked there show you all income and expenses sorted by the categories *Dividends*, *Interest*, *Earnings*, *Taxes*, and *Fees*. In addition to tabular lists, you will also find graphic display options that show the reference accounts. In a horizontal tab-like bar above the display area, you can also select different time periods as the basis for the display (Figure 7).

Portfolio Performance also lets you classify your securities. You can do this manually in *Taxonomies | Asset Classes*. You can assign your assets to individual, freely definable classes. To do this, right-click in the free area of the table view and select the option *Add new class* from the context menu. Then assign your assets to the individual classes.

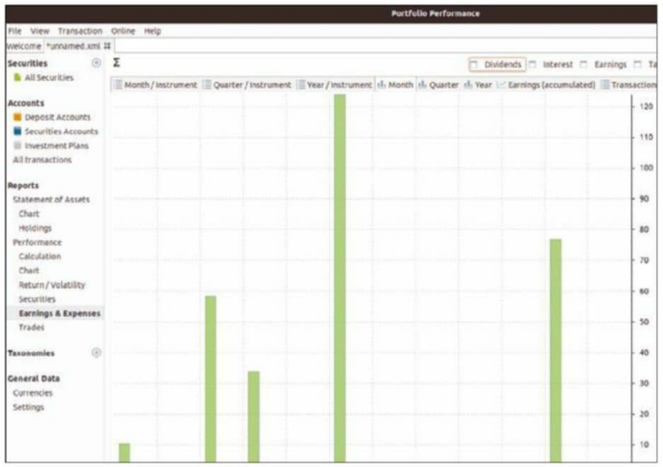


Figure 7: Earnings and expenses can be displayed in great detail.

For a useful overview of the individual classes even if you have a large portfolio, use the small horizontal toolbar on the right above the table view. By clicking on one of the symbols, you can visualize the individual asset classes and their percentage values. Pie, tree, and area diagrams are available (Figure 8).

To share the collected data with other applications and services, Portfolio Performance offers an import and export function that can be accessed via the File menu. When importing, the program not only reads bank documents in PDF format, but also comma-separated (CSV) files and templates from Comdirect, Consors, and Deutsche Bank. The program also understands Ebase account statements. In a separate dialog, Portfolio Performance lets you export to CSV tables, which are usually supported by spreadsheet programs.

Conclusions

Thanks to Portfolio Performance, you can keep track of your investment holdings at all times. The program is not only suitable for managing securities accounts, but also includes sight and fixed-term deposit accounts.

However, the focus is clearly on evaluating securities. With sophisticated input dialogs and the ability to automatically look up historical prices at your fingertips, you can keep tabs on the performance of your securities and trade accordingly. Numerous graphic display options offer a quick overview – even for large portfolios, where a table-only view quickly becomes very dry, and not very meaningful.

For investors who are looking for professional and practical software to manage their investments, Portfolio Performance is an excellent first choice. ■■■

Figure 8: A chart highlights the portfolio performance of individual asset classes.

Info

[1] Portfolio Performance: *https://www.portfolio-performance.info/en*

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.

FOSDEM '21

Date: February 6-7, 2021

Location: Virtual Event

Website: *https://fosdem.org/2021/*

FOSDEM is a free event for software developers to meet, share ideas, and collaborate. Every year, thousands of developers of free and open source software from all over the world gather at the event in Brussels. In 2021, they will gather online.

CloudFest 2021

Date: March 23-25, 2021

Location: Virtual Event

Website: *https://www.cloudfest.com/*

CloudFest returns on an all-digital platform amid a global pandemic as we take stock of the strengths, weaknesses, risks, and opportunities that are revealed by a globe-spanning threat like COVID-19. Join us at CloudFest 2021, and let's build something great together.

Events

DrupalCon Europe	December 8-11	Virtual Experience	https://events.drupal.org/
Deep Learning 2.0 Summit	January 28-29	Virtual Event	https://bit.ly/Deep-Learning-Summit
FOSDEM 2021	February 6-7	Virtual Event	https://fosdem.org/2021/
CloudFest 2021	March 23-25	Virtual Event	https://www.cloudfest.com/
Cloud Expo Europe	March 24-25	London, United Kingdom	https://www.cloudexpoeurope.com/
Kubernetes Community Days	April 8-9	Amsterdam, Netherlands	https://sessionize.com/kcdams2021/
DevOpsCon London	April 19-21	London, United Kingdom	https://devopscon.io/london/
Linux Storage Filesystem & MM Summit	May 12-14	Palm Springs, California	https://events.linuxfoundation.org/lsfmm/
LISA21	June 1-3	Anaheim, California	https://www.usenix.org/conference/lisa21
ISC High Performace	June 27-July 21	Frankfurt, Germany	https://www.isc-hpc.com/
KVM Forum	August 2-4	Vancouver, British Columbia	https://events.linuxfoundation.org/
Embedded Linux Conference North America	August 4-6	Vancouver, British Columbia	https://events.linuxfoundation.org/
Open Source Summit North America	August 4-6	Vancouver, British Columbia	https://events.linuxfoundation.org/
Embedded Linux Conference Europe	September 29-Oct 1	Dublin, Ireland	https://events.linuxfoundation.org/
Open Source Summit Europe	September 29-Oct 1	Dublin, Ireland	https://events.linuxfoundation.org/

CALL FOR PAPERS

We are always looking for good articles on Linux and the tools of the Linux environment. Although we will consider any topic, the following themes are of special interest:

- System administration
- Useful tips and tools
- Security, both news and techniques
- Product reviews, especially from real-world experience
- Community news and projects

If you have an idea, send a proposal with an outline, an estimate of the length, a description of your background, and contact information to *edit@linux-magazine.com*.

The technical level of the article should be consistent with what you normally read in *Linux Magazine*. Remember that *Linux Magazine* is read in many countries, and your article may be translated into one of our sister publications. Therefore, it is best to avoid using slang and idioms that might not be understood by all readers.

Be careful when referring to dates or events in the future. Many weeks could pass between your manuscript submission and the final copy reaching the reader's hands. When submitting proposals or manuscripts, please use a subject line in your email message that helps us identify your message as an article proposal. Screenshots and other supporting materials are always welcome.

Additional information is available at: *http://www.linux-magazine.com/contact/write_for_us.*

NOW PRINTED ON recycled paper from 100% post-consumer waste; no chlorine bleach is used in the production process.

Authors

Erik Bärwaldt	30, 76, 90
Zack Brown	12
Bruce Byfield	6, 25, 36, 46
Joe Casad	3
Mark Crutch	73
Jon "maddog" Hall	74
Thomas Kaffka	20
Sirko Kemter	80
Charly Kühnast	45
Vincent Mealing	73
Graham Morrison	84
Oliver Nickel	16
Dmitri Popov	50
Mike Schilli	54
Mayank Sharma	60
Scott Sumner	64
Juan P. Tobar	40
Jack Wallen	8

Contact Info

Editor in Chief
Joe Casad, jcasad@linux-magazine.com

Copy Editors
Amy Pettle, Megan Phelps

News Editor
Jack Wallen

Editor Emerita Nomadica
Rita L Sooby

Managing Editor
Lori White

Localization & Translation
Ian Travis

Layout
Dena Friesen, Lori White

Cover Design
Dena Friesen

Cover Image
© Aleksei Sysoev, 123RF.com

Advertising
Brian Osborn, bosborn@linuxnewmedia.com
phone +49 89 3090 5128

Marketing Communications
Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
2721 W 6th St, Ste D
Lawrence, KS 66049 USA

Publisher
Brian Osborn

Customer Service / Subscription
For USA and Canada:
Email: cs@linuxpromagazine.com
Phone: 1-866-247-2802
(Toll Free from the US and Canada)

For all other countries:
Email: subs@linux-magazine.com

www.linuxpromagazine.com – North America
www.linux-magazine.com – Worldwide

Printed in Nuremberg, Germany by hofmann infocom GmbH on recycled paper from 100% post-consumer waste; no chlorine bleach is used in the production process.

Distributed by Seymour Distribution Ltd, United Kingdom

LINUX PRO MAGAZINE (ISSN 1752-9050) is published monthly by Linux New Media USA, LLC, 2721 W 6th St, Ste D, Lawrence, KS, 66049, USA. Periodicals Postage paid at Lawrence, KS and additional mailing offices. Ride-Along Enclosed. POSTMASTER: Please send address changes to Linux Pro Magazine, 2721 W 6th St, Ste D, Lawrence, KS 66049, USA.

Published monthly in Europe as Linux Magazine (ISSN 1471-5678) by: Sparkhaus Media GmbH, Zieblandstr. 1, 80799 Munich, Germany.

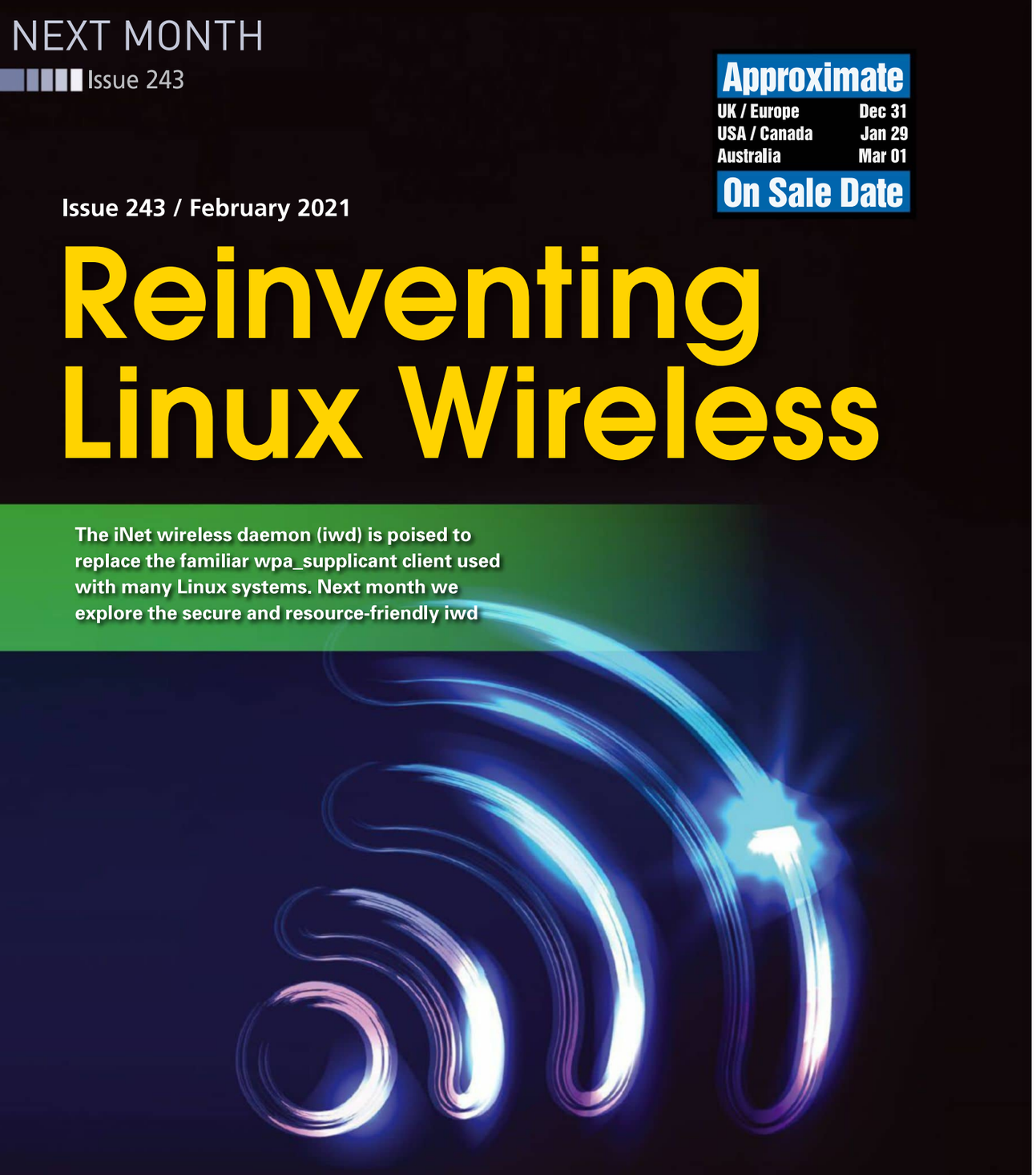
Approximate On Sale Date

UK / Europe	Dec 31
USA / Canada	Jan 29
Australia	Mar 01

Issue 243 / February 2021

Reinventing Linux Wireless

The iNet wireless daemon (iwd) is poised to replace the familiar wpa_supplicant client used with many Linux systems. Next month we explore the secure and resource-friendly iwd



Preview Newsletter

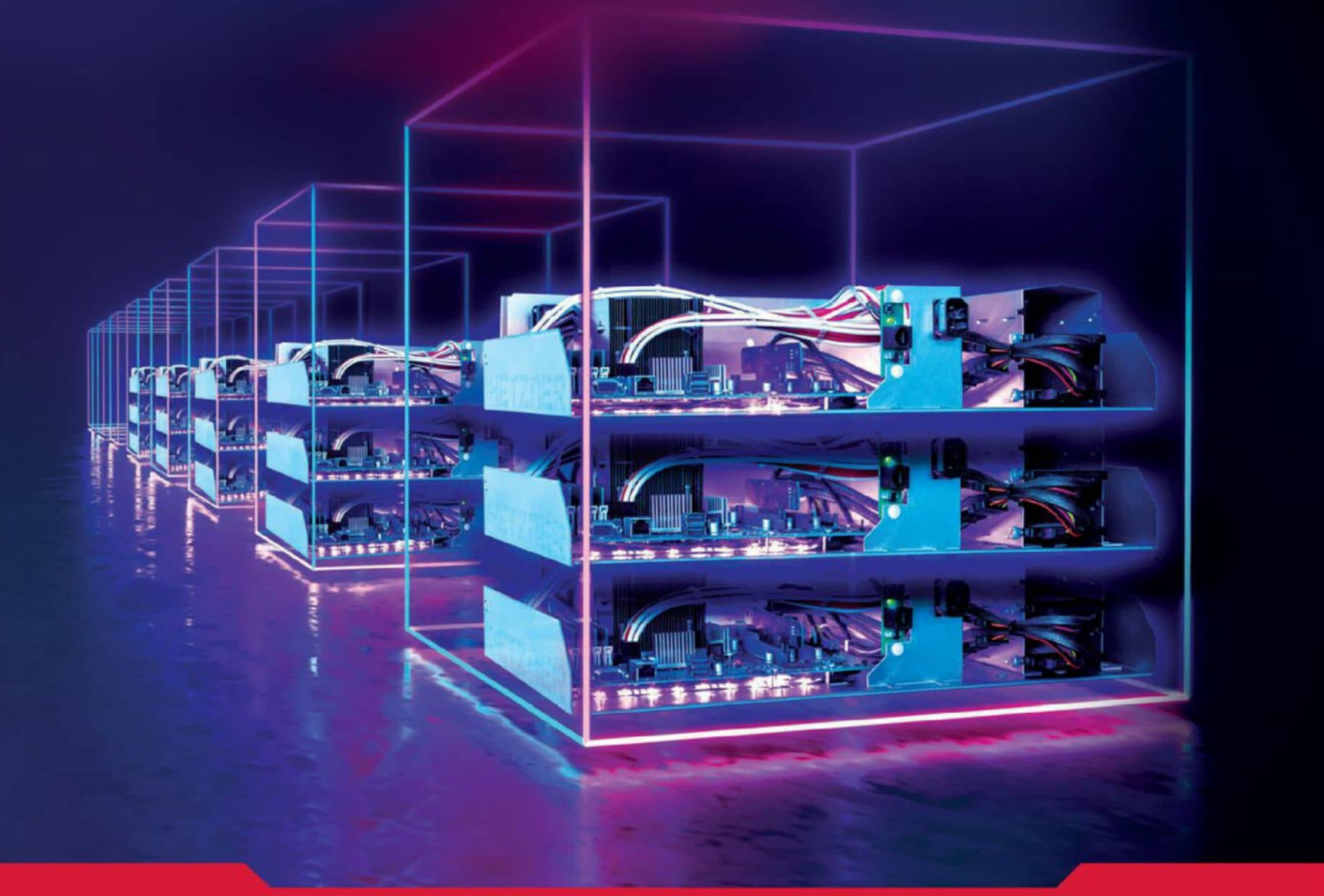
The Linux Magazine Preview is a monthly email newsletter that gives you a sneak peek at the next issue, including links to articles posted online.

Sign up at: ***https://bit.ly/Linux-Update***

Lead Image © file404, 123RF.com

HETZNER

DEDICATED ROOT SERVER
DESIGNED FOR PROFESSIONALS



Dedicated Root Server AX41-NVMe

- ✔ AMD Ryzen 5 3600 Simultaneous Multithreading
- ✔ 64 GB DDR4 RAM
- ✔ 2 x 512 GB NVMe SSD
- ✔ 100 GB Backup Space
- ✔ Traffic unlimited
- ✔ Location Finland and Germany
- ✔ No minimum contract
- ✔ Setup Fee £35.50

monthly from £31

Dedicated Root Server AX51-NVMe

- ✔ AMD Ryzen 7 3700X Simultaneous Multithreading
- ✔ 64 GB DDR4 ECC RAM
- ✔ 2 x 1 TB NVMe SSD
- ✔ 100 GB Backup Space
- ✔ Traffic unlimited
- ✔ Location Finland and Germany
- ✔ No minimum contract
- ✔ Setup Fee £54.00

monthly from £49

www.hetzner.com