Maker Makeover

Add a GPIO to your Linux desktop or laptop system

LINUX PRO MAGAZINE

ISSUE 258 – MAY 2022

Clean IT

The quest for more sustainable software

Backup Integrity
How to avoid silent data corruption

Maltrail
Detect network attacks

LibreOffice Tricks
Build a music database

NocoBD
Create programs without writing code

OpenToonz
Conjure up your own animations

10 FANTASTIC FOSS FINDS!

LINUX NEW MEDIA
The Pulse of Open Source

PEARL Edition

TUXEDO InfinityBook S 14

Intel Core i5-1135G7
Intel Iris Xe Graphics

1,1 kg light
16,8 mm thin

Metallic rosé special color
Trendy & exclusive

73 Wh battery
and Low Power display

100%
Linux

5
Year
Warranty

Lifetime
Support

Built in
Germany

German
Privacy

Local
Support

TUXEDO COMPUTERS 18th ANNIVERSARY

tuxedocomputers.com

BETTER ANGELS AND BETTER MINING

Dear Reader,

We're told that modern war will include the elements of cyber war, and now, as war rages in Europe, we look to our networks with renewed vigilance. In this context, the news of a stolen NVIDIA private key seems especially alarming. Although this event is not related to any military conflict, and does not appear to be a state-sponsored act, it is yet another wake-up call about how far we have to go before we can truly say that our systems are secure.

In early March, reports trickled in about bogus binaries that were signed with an NVIDIA code-signing certificate but *not* created by NVIDIA [1]. According to the reports, the attack appears related to a recent intrusion on NVIDIA's internal systems by the Lapsus$ ransomware gang.

As it turns out, the stolen key used to sign the binaries actually expired back in 2014, which would make you think it was too old to cause any damage, but the attack exposed yet another little known quirk about Microsoft Windows: Under the right conditions, the Windows driver-signing policy will accept certificates issued prior to July 29, 2015, even if they are expired. This measure was adopted for backward-compatibility reasons when Microsoft introduced the policy with Windows 10. Luckily, several malware scanners already know about these rogue drivers and are scanning for them. However, the onus is on all who use or administer Windows systems to review security policies and check for instances of drivers signed with the rogue cert.

As a Linux guy, I would say this seems a little like Groundhog Day – yet another example of Microsoft fixing a problem by opening up another problem elsewhere on the system that they bury in the fine print and don't really explain to people. As a journalist, however, I would have to add that there are probably lessons in this for everyone. For instance, we don't know how the attackers got on NVIDIA's network in the first place, but an unpatched or misconfigured Linux system could well have been part of it. Another lesson: Expired keys can still make mischief, so lock them down and don't let them get lost.

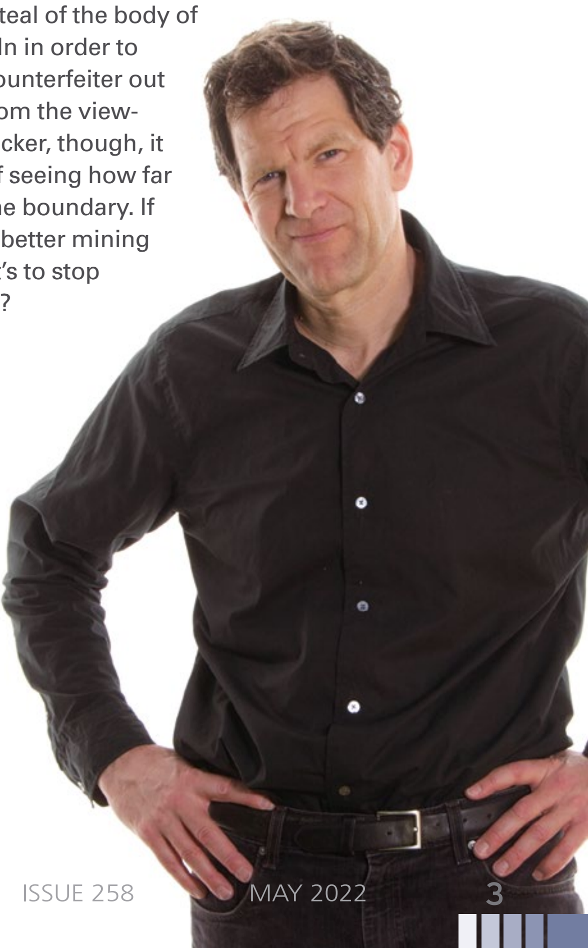
Perhaps the weirdest and most intriguing part of this story is the apparent motive for the attack. According to reports, NVIDIA recently introduced new safeguards in the drivers for its RTX 30-series GPUs designed to limit the use of the GPU as a tool for calculating hash values in crypto mining. NVIDIA is apparently tired of crypto miners buying up all the high-end hardware and thereby shorting the company's traditional customers, such as gamers and other graphics power users.

In this case, the ransom the attackers wanted was not money but was, instead, the key to unlock the Lite Hash Rate feature [2] so that they could use the graphics cards for crypto mining. If this scheme had worked, it would have ushered in a whole new era for ransom attacks, where you don't just try to get rich but actually attempt to control a company into changing or circumventing its own policies. There are some reasons why this kind of attack doesn't sound very feasible. A company can begrudgingly admit to its shareholders that it had to pay money in a ransomware attack, but to actually change your corporate policies because a ransom attacker told you to looks really bad for management, which means that the CEO who agreed to do it would probably get fired for doing so. And to make a deal under the table with the criminals to avoid pubic scrutiny might actually be illegal.

As a means for exerting power, this whole scenario seems a bit hairbrained to me, kind of like when the two counterfeiters tried to steal of the body of Abraham Lincoln in order to spring a third counterfeiter out of prison [3]. From the viewpoint of the attacker, though, it is all a matter of seeing how far you can push the boundary. If it works, lots of better mining up ahead. What's to stop you from trying?

Joe

Joe Casad,
Editor in Chief



Info

[1] "Leaked Stolen NVIDIA Key Can Sign Windows Malware" by Gareth Corfield, *The Register*, March 5, 2022, *https://www.theregister.com/2022/03/05/nvidia_stolen_certificate/*

[2] A Further Step Towards Getting GeForce Cards into the Hands of Gamers: *https://blogs.nvidia.com/blog/2021/05/18/lhr/*

[3] "The Plot to Steal Lincoln's Body" by Peggy Robertson, *American Heritage*, April/May 1982, *https://www.americanheritage.com/plot-steal-lincolns-body*

LINUX MAGAZINE
MAY 2022

ON THE COVER

NEWS

REVIEWS

COVER STORIES

IN-DEPTH

Clean IT

Most people know you can save energy by changing to more efficient light bulbs, but did you know you can save energy with more efficient software? This month we examine the ongoing efforts to bring sustainability to the IT industry.

Manjaro 21.2 Qonos and DragonFly BSD 6.2.1

Two Terrific Distros on a Double-Sided DVD!

Manjaro 21.2 Qonos

64-bit

Sometimes called the Ubuntu of Arch Linux derivatives, Manjaro has made Arch Linux accessible for general users much like Ubuntu did for Debian. While Manjaro has replaced Arch Linux's arcane installation process with an adaptation of the graphical Calamares installer, it does support rolling installations like Arch Linux and bases its repositories on those of Arch Linux. In addition, Manjaro continues the traditions of Arch, initially installing a minimum of applications and emphasizing a user's choice rather than installing a curated array of applications.

Codenamed Qonos, Manjaro 21.2 is a minor release, and its visible changes consist mostly of updates to stock applications. Worth noting, however, are some improvements to the Calamares installer and an increase in speed that has been noted by several reviewers. Also included is the first new Xorg Server for some time, as well as additional driver fixes for the NVIDIA drivers for Kepler video cards. Other new additions include PipeWire, the sound server expected to eventually replace PulseAudio, and System76's new COSMIC Desktop.

If you have ever wondered about Arch Linux but had difficulties with the installation, Manjaro is the ideal solution. With its emphasis on usability, Manjaro is well on its way to becoming a major distribution for beginners and mid-level users with innovations not found in other distributions.

DragonFly BSD 6.2.1

64-bit

Like Linux, FreeBSD is a free Unix-like operating system. DragonFly BSD in particular was forked from FreeBSD in 2003 over differences about the direction of development. Its subsequent development was influenced by the old AmigaOS, which founder Matthew Dillon had previously worked on.

From the start, DragonFly BSD has focused on redefining the basic core of BSD. The latest release is no exception. It features virtual kernels, a rewritten network subsystem, support for up to 55TB of swap space, and optimization for the swap cache of solid state drives (SSDs). Of particular interest, DragonFly BSD's filesystem, HAMMER2, shares many of ZFS's features, including easy access to as many as 60 one-day snapshots for backup and instant recovery after a crash. In addition, the latest release includes new enhancements for a mail agent, with support for local mail delivery and basic remote mail transfers. These technical innovations are joined by thousands of ported third-party applications.

DragonFly BSD will appeal especially to those with a knowledge of the internal workings of Unix-like systems. However, DragonFly BSD is also user-friendly enough to be a perfect place to start learning how the BSD family of operating systems compares to Linux.

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

Take Your First Steps In Web Development!
Linux Professional Institute
WEB DEVELOPMENT ESSENTIALS
JavaScript
Node.js
HTML
CSS
SQL
NEW

Web Development Essentials

Modern software applications are commonly developed for the Web. Linux Professional Institute‘s new Web Development Essentials program supports you in your first steps in software development. It includes learning materials suitable for both training and self-study. Upon passing the Web Development Essentials exam, you receive a certificate to prove your skills.

LPI Learning Materials are available at learning.lpi.org.
Find out more about the program
and the exam at lpi.org/wde

Linux Professional Institute

NEWS

Updates on technologies, trends, and tools

THIS MONTH'S NEWS

Linux Mint Dropping Blueberry Bluetooth Configuration Tool

For the longest time, Linux Mint depended on Blueberry for its Bluetooth background service. With the release of Linux Mint 21, that all changes, as the developers have opted to migrate to Blueman.

One of the primary reasons for this change is that the latest version of gnome-bluetooth (the Bluetooth back end for Blueberry) introduced a few changes that broke compatibility with Blueberry. Unfortunately, the Blueberry developer has no desire to see his work used outside of Gnome. Because of this, Blueberry will have trouble with non-Gnome desktops going forward, which is one of the reasons why the Linux Mint team decided to go a different route.

Another reason for this change is the Blueman tool works better with Bluetooth audio headsets and can connect to a much wider range of devices.

The Linux Mint developers are currently in the process of integrating Blueman with the desktop OS and hope to have it ready for the full release of version 21. Currently, there is no official release date set for Linux Mint 21, but (if history is any guide) it should be sometime this summer (2022).

To read more about this upcoming change, and other interesting bits about the new release, check out this issue of the Linux Mint Monthly Newsletter (*https://blog.linuxmint.com/?p=4285*).

Fedora 36 Beta Now Has a Release Date

It's official, Fedora 36 now has two different release dates. If things go as planned, the beta of the distribution will become available on March 15, 2022. If there's a delay, Fedora 36 will be released on March 22, 2022. Once the public beta testing is complete, the official release will be April 19, 2022, or, if there's a delay, April 26, 2022.

As for new features, the most notable will be the addition of Gnome 42, which improves both user interface and functionality. The changes to Gnome 42 include a system-wide dark theme preference, wallpapers for both dark and light themes, updates to the folder icon theme, even more support for the *libadwaita* library, an improved System Settings application (thanks to GTK4), a new default text editor (shifting from gedit to Gnome Text Editor), and an improved screenshot tool and native screen recording.

Other additions to Fedora 36 include the 5.17 kernel, Java 17, Golang 1.18, Noto typefaces as the default, GCC 12.0, glibc 2.35, and PHP 8.1. In addition, the default

Wayland session now uses the proprietary NVIDIA driver, and the RPM database has been moved from `/var/` to `/usr`.

To see all of the changes being made to Fedora 36, check out the ChangeSet on the Fedora Wiki (*https://fedoraproject.org/wiki/Releases/36/ChangeSet*).

AV Linux MX-21 Released for All Your Audio/Video Production Needs

Code-named "Consciousness," the latest release of AV Linux has been completely rebuilt, from the ground up, which makes it the first iteration that wasn't a respin of a previous release. Based on Debian 11, AV Linux MX-21 was built with the same tools used to build MX Linux and antiX.

Because this new release is a complete rebuild, there is no way to upgrade from previous releases to MX-21. In other words, you'll have to do a fresh install to gain the benefits of AV Linux MX-21.

What are those benefits? First off, AV Linux MX-21 ships with kernel 5.15 and a brand new Mesa graphics stack. The one caveat to those two changes: AV Linux no longer supports 32-bit architecture. The kernel version in MX-21 is Liquorix, which is a high-performance kernel geared for streaming and ultra-low latency.

You'll also find a new Yabridge GUI, YADbridge, that makes managing Windows Virtual Studio Technology (VST) plugins much easier. VST is an audio plugin interface that integrates software synthesizers and effects into a digital audio workstation (DAW). There are also new versions of all the regular audio/visual tools that are installed by default, as well as a new AV Linux Assistant tool.

AV Linux MX-21 uses the Xfce 4.16 desktop environment that has been customized with a new theme, first-run splash screen, 4K wallpapers, gradient wallpaper generator, and the GDebi package installer has been replaced by a custom Thunar Action.

Download the ISO of AV Linux MX-21 now (*https://www.bandshed.net/avlinux/*).

Slax Proves You Can't Keep a Good Linux Distribution Down

Anyone who's ever looked into bringing computer hardware back to life has considered one or more lightweight Linux distributions. In the search for the right operating system (OS), you might have come across Slax, which was a Debian-based OS that was quite popular for a while. However, the pandemic wreaked havoc on the development life cycle, so we hadn't seen anything new from the maintainer since 2018.

That all changes now, with the release of Slax 11.2. Based on Debian Bullseye, Almost ready for release, Slax 11.2 will include features like kernel 5.10, support for 32 and 64-bit systems, the PCManFM file manager, the ConnMan network manager, the SciTE text editor, the xterm terminal emulator, and more. The developer has opted to leave out the Chromium browser by default (due to its size). However, by clicking on the Chromium icon, the browser will automatically install.

Although Debian shifted to the OverlayFS union mount filesystem, Slax had to continue with aufs to provide the necessary support for the `slax activate` command, as OverlayFS doesn't allow for modification of the existing overlay filesystem on the fly.

At the moment, there's no definitive release date, but anyone interested in testing Slax 11.2 can download an ISO image for either 64- or 32-bit architecture (*https://sharegorilla.com/q/Rd6dZ87*).

Follow the Slax developer blog (*https://www.slax.org/blog.php*) to stay in the know about this lightweight Linux distribution.

MORE ONLINE

Linux Magazine
www.linux-magazine.com

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

Working with the Lustre Filesystem
• Petros Koutoupis
The Lustre open source distributed, parallel filesystem scales to high-performance computing environments.

What Is an IOPS Really?
• Jeff Layton
IOPS is mentioned quite often when benchmarking and testing storage systems, but what does it really mean? We discuss and explain what an IOPS is, and how to measure it.

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

Attack and Defense Techniques
• Matthias Wübbeling
The MITRE ATT&CK and D3FEND knowledge databases provide useful techniques for securing your IT infrastructure.

macOS File Recovery and Security
• Thomas Joos
macOS on-board tools and third-party applications can help prevent the loss of files and make security and backup your first priority.

eSIMs in the Enterprise
• Marco Föllmer
In this overview, we look at the opportunities eSIM technology offers for the mobile workplace and what IT managers need to watch out for in deployment and management.

Dirty Pipe Might Be the Most Severe Vulnerability to Hit Linux in Years

The name Dirty Pipe is an homage to the Dirty Cow vulnerability, discovered in 2016, and a pipeline, which is a mechanism within Linux that allows processes to share data. Tracked as CVE-2022-0847, Dirty Pipe was discovered when a researcher was troubleshooting corrupted files that continued to appear on a customer's Linux server. It took months of analysis, but eventually, Max Kellermann (the researcher in question, from Ionos) discovered those files were due to a bug in the Linux kernel and figured out a way to weaponize the vulnerability. Once weaponized on a Linux machine, anyone with an account could then add an SSH key to the root user's account such that any untrusted user could remotely access the server with full root privileges.

The same vulnerability also makes it possible for attackers to hijack an SUID binary to create a root shell, which allows untrusted users to overwrite data, even in read-only files. Other actions that can be taken on a vulnerable machine include creating a cron job that serves as a backdoor and modifying a script or binary used by a privileged service.

Find out more about Dirty Pipe in this Red Hat security bulletin (*https://access.redhat.com/security/vulnerabilities/RHSB-2022-002*).

A Decades-Old Linux Backdoor Has Been Discovered

Back in 2013, during a forensic investigation, the Advanced Cyber Security Research team from Pangu Lab discovered a rather elusive piece of malware. Between 2016 and 2017, the hacker collective, The Shadow Brokers, leaked a large amount of data that was allegedly stolen from the Equation Group (with links to the NSA) that contained a number of hacking tools and exploits. Around the same time, the group leaked another data dump that contained a list of servers that had been hacked by the Equation Group.

According to the Advanced Cyber Security Research team, Bvp47 was used to target the telecom, military, higher-education, economic, and science sectors and hit more than 287 organizations in 47 countries. These attacks lasted over a decade as the malicious code was created so the hackers could retain long-term control over an infected device. And because the attack used zero-day vulnerabilities, there was no defense against it.

The Pangu Lab operation was code-named "Operation Telescreen" and the end result of the operation discovered this backdoor requires a check code bound to the host in order to function normally. They also determined Bvp47 to be a top-tier APT backdoor.

As far as whether or not Bvp47 is still in use today, there is no indication that is the case. But given the nature of the exploit, it wouldn't come as a shock to any research lab to discover those leaked tools had been used to cobble together even more dangerous malware.

Read the Pangu Lab report to find out more (*https://www.pangulab.cn/en/post/the_bvp47_a_top-tier_backdoor_of_us_nsa_equation_group/*).

Photo by David Szweduik on Unsplash

Zack's Kernel News

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

By Zack Brown

Git Merge "Simplification" Advice

Bjorn Helgaas submitted some PCI patches in the form of a merge request from another Git tree. This is a standard part of the development process for larger distributed projects like the Linux kernel, and this one included work from dozens of contributors. The idea is that a bunch of people work on a given sub-project in relative isolation so their changes don't break everyone else's work on the main Linux tree. Then, with the merge request, the contributors ask Linus Torvalds to resolve any conflicts that their changes might have produced with other changes going into the kernel at the same time. No biggie, nothing to see here. Tens of thousands of contributors can get their hands dirty at the same time, without throwing dirt onto any of their fellow contributors' hands while they're at it.

In this case, Linus noticed some wonky twirling going on behind the scenes, and it posed a problem for him. Specifically, Bjorn and his fellow PCI travelers had already done some merging from multiple separate trees (used for different sub-sub-projects within their sub-project), followed by a patch reversion, so that all merges going into Linus's official repository would seem to come from the same tree. It's not psychotic; they were just trying to keep things simple.

So first of all, Linus objected to the patch reversion itself. Patch reversions remove a patch that was previously accepted into a tree, but a reversion is itself a patch that also needs to be accepted via the same process as other patches – including having a meaningful commit message, which the PCI patch reversion did not. However, it's a relatively common occurrence for patch reversions to have no meaningful commit message – developers don't tend to see the point of it because all the patch reversion does is take something out that had recently been put in.

In this particular case, Linus pointed out that the purpose of the reversion was not that the patch itself had been bad – it was specifically in order to reduce potential conflicts when Linus would eventually merge the PCI trees into his own.

There were two problems with this. First, the normal conclusion to draw about a patch reversion is that the patch was bad. In such a case, if the same patch can be seen coming from another tree (which was the case here – the PCI folks reverted the patch so it could come back again from somewhere else), the natural conclusion to draw, said Linus, was that he should avoid merging the patch from that other tree as well. If the identical patch was bad once, it would be bad again, right? So Linus had to figure out that the patch had actually been intended to go into the tree – which he did, but it meant more work.

The second problem was that the reversion and behind-the-scenes merges had been done to reduce the conflicts Linus would see against his own tree. Linux declared it "pure and utter garbage, because I end up with the merge conflict *ANYWAY* due to the other changes, and now instead of going 'ok, the PCI tree had that same commit, all good', I have to go 'ok, so the PCI tree had the same commit, but it was reverted in the networking tree, so now I have both sides making different changes and a very confusing merge'."

Linus compared this to a similar situation where developers "rebase" their tree as a different way to avoid merge conflicts. In Git, sub-projects will pull the latest version of the main project tree into their own to act as a "base" and do all their work against that version. But by the time they're ready to submit the code upstream to Linus, his kernel has already advanced. So the sub-project may choose to pull from his tree again before submitting their own patches, thus changing their base version to match the main official tree. Then they can resolve the conflicts themselves and send a conflict-free merge request to Linus.

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**.

The problem with rebasing is that it can utterly destroy the make-good-senseness of each individual patch in that merge, because now the code differences are not against the version the developers thought they were working on but are against other versions that the developers never even looked at. The end result is the same code, but the history of those developers' changes has been reduced to a fine slurry and garbled.

Linus wants clarity in all kernel patches. Among other things, it helps with debugging, when developers may need to identify an earlier patch to revert. If the bogus patch itself makes as much sense as possible, it's easier to see where it messed things up.

Linus summed up his official policy as, "Don't make my life 'easier' by doing stupid things, and DO put a reason for every single commit you do. Reverts aren't 'oh, I'm just turning back the clock, so no reason to say anything else'."

Loading Modules from Containers

Thomas Weißschuh wanted to make life easier. He said, "We are using nested, privileged containers which are loading kernel modules. Currently we have to always pass around the contents of /lib/modules from the root namespace which contains the modules." He posted a Request for Comments (RFC) suggesting a new `request_module()` system call, which would allow his containers to get modules from the root system without having to do so much bookkeeping.

The whole point of containers, though, is that they are supposed to resemble a separate running system as much as possible. This is the principle behind Google and Amazon offering cloud services that provide computer "instances" to users. Those instances are nothing more than Linux containers running on top of other systems.

Greg Kroah-Hartman replied to Thomas, saying, "So you want any container to have the ability to 'bust through' the containers and load a module from the 'root' of the system? That feels dangerous." He went on, "why are modules somehow 'special' here, they are just a resource that has to be allowed (or not) to be accessed by a container like anything else on a filesystem."

Thomas said he wasn't trying to dissolve the barriers between the host and virtual system entirely – he wanted to use the `CAP_SYS_MODULE` capability to give a container the right to load modules in this way. Kernel capabilities were introduced in the Linux kernel version 2.2, to divvy up abilities that used to be under the one umbrella of the "root" user. Before then, the root user could do anything and revel in the glory and the blood. After version 2.2, the root user was chastened and humbled and had to check the list of allowed capabilities before doing just any old thing.

The thing about modules, Thomas said, was that they needed to match the running kernel on the host system in order to load. And this was something only the root namespace could access. And he said, "the biggest problems would probably arise if the root namespace has non-standard modules available which the container would normally not have access to."

Meanwhile Andy Lutomirski said, regarding Thomas's initial post, "I feel like I'm missing something, and I don't understand the purpose of this syscall. Wouldn't the right solution be for the container to have a stub module loader (maybe doable with a special /sbin/modprobe or maybe a kernel patch would be needed, depending on the exact use case) and have the stub call out to the container manager to request the module? The container manager would check its security policy and load the module or not load it as appropriate."

Christian Brauner agreed with Andy's assessment that the container manager should be the gatekeeper for loading modules into containers.

Andy asked Thomas, "What exactly is the container doing that causes the container's copy of modprobe to be called?" To which Thomas replied, "The container is running an instance of the docker daemon in swarm mode. That needs the 'ip_vs' module (amongst others) and explicitly tries to load it via modprobe." Swarm mode is a Docker virtualization feature that just means you're starting up and managing a bunch of virtual systems. And IP Virtual Server (`ip_vs`) is a load balancer for distributing network requests amid the "swarm."

To which Andy replied, "Do you mean it literally invokes /sbin/modprobe? If so, hooking this at /sbin/modprobe and calling out to the container manager seems like a decent solution." In other words, the container could invoke `modprobe` as if nothing special was going on, but `modprobe` would "hook" or "catch" or "get weird with" that call. Diverting it to the container manager would then do the right thing with the request, thus protecting the border between the container and the host system.

Thomas liked this idea and said he'd see if it would work for his project.

Meanwhile Luis Chamberlain's ears pricked up at this interesting use case and suggested that someone write documentation that could have helped Thomas and subsequently help others in the future to handle similar situations in similar ways.

Andy replied enthusiastically, "If someone wants to make this classy, we should probably have the container counterpart of a standardized paravirt interface. There should be a way for a container to, in a runtime-agnostic way, issue requests to its manager, and requesting a module by (name, Linux kernel version for which that name makes sense) seems like an excellent use of such an interface."

And suddenly there was an implementation discussion going on.

Apparently Thomas's original problem is something a bunch of people have been duct taping in various ways for quite some time, and a real solution would be welcome. Christian rattled off several ways users had loaded modules from the host system into containers. And he asked Andy, "So what was your idea: would it be like a device file that could be exposed to the container where it writes requests to the container manager? What would be the advantage to just standardizing a socket protocol?"

To which Andy said, "My idea is standardizing *something*. I think it would be nice if, for example, distros could ship a /sbin/modprobe that would do the right thing inside any compliant container runtime as well as when running outside a container."

And Christian replied with this suggestion:

"I think we never want to trust the container's modules.

"What probably should be happening is that the manager exposes a list of modules the container can request in some form. We have precedence for doing something like this.

"So now modprobe and similar tools can be made aware that if they are in a container they should request that module from the container manager be it via a socket request or something else."

Andy asked, "Why bother with a list? I think it should be sufficient for the container to ask for a module and either get it or not get it." And Christian clarified, "I just meant that the programs in the container can see the modules available on the host. […] But yeah, it can likely be as simple as allowing it to ask for a module and not bother telling it about what is available."

Thinking further about this, Andy remarked, "If the container gets to see host modules, interesting races when containers are migrated CRIU-style will result." CRIU is a tool for completely stopping a running container and storing it to disk. The container can then be started up again at any time, from exactly the moment it stopped. So Andy was apparently saying that if a container saw a list of available modules and then was frozen with CRIU, migrated to a new running system, and then started up again, it might then request a module on that list that was no longer available on its new host system.

The conversation ended there, but it's clear that the ability for containers to load modules via the host system will be cleaned up and regularized at some point in the not-too-distant future. To me, it's exciting that Thomas ran into this thorny problem, tried to solve it in one way, almost got the smackdown, but then it turned out that other people joined in to find a more general solution that would work for all cases and make many lives easier.

Git Tree Synchronicity

Konstantin Komarov from Paragon Software has been maintaining the NTFS3 code for a little while and is ironing out the developer process for patch submissions, merge and pull requests, and whatnot. Konstantin asked Linus Torvalds, "Right now my github repo [is] still based on 5.14-rc7. Do I need to update it with git merge up to 5.15-rcX? Or will it be ok to send git pull request as is and back merge master only when 5.15 will release?"

Linus replied:

"Oh, keep your previous base, and just send me a pull request with your changes and no merge.

"In fact, the best workflow is generally to avoid merging from me as much as humanly possible, but then if you notice that we're all in sync, and you have nothing pending in your tree, you can basically fast-forward and start any new development at some new point.

"But even then, it's a good idea to make that new point be something well-defined – like a full release, or at least an rc release (usually avoiding rc1 is good, since rc1 can be a bit experimental).

"But I have no problems pulling from a git tree that is based on older kernels. I much prefer tha[t] to having people rebase their work overly aggressively, or having people do lots of merges of my tree.

"At some point, if you end up being multiple releases behind, it ends up being inconvenient for both of us just because some infrastructure has changed, so _occasionally_ syncing up is just a good idea.

"In my experience, people tend to do it too much, rather than too little. Don't worry too much about it."

And that was that.

The New "No New Warnings" Warning

Recently, Linus threw all the kernel developers into a jar and shook the jar really hard by making all compiler warnings into errors by default. This broke everything for light years in all directions and resulted in much wailing and unhappy bioluminescent patterns.

Now things have returned pretty much to normal, but developers are each finding their own ways to deal with the new "no warnings" policy. For example, Paolo Bonzini submitted some KVM patches from a variety of other contributors, one of which was described in the patch comment as, "avoid warning with -Wbitwise-instead-of-logical."

Linus prepared to put Paulo back in the jar, saying:

"Christ. Please no.

"Guys, you can't just mindlessly shut off warnings without even thinking about the code.

"Apparently the compiler gives completely insane warning 'fixes' suggestions, and somebody just completely mindlessly followed that compiler badness.

"The way to do a logical 'or' (instead of a bitwise one on two boolean expressions) is to use '||'.

"Instead, the code was changed to completely insane

```
(int) boolexpr1 | (int) boolexpr2
```

"thing, which is entirely illegible and pointless, and no sane person should ever write code like that.

*"In other words, the *proper* fix to a warning is to look at the code, and *understand* the code and the warning, instead of some mindless conversion to just avoid a warning.*

"NEVER EVER do mindless changes to source code because the compiler tells you to. Apparently the clang people wrote a particularly bad warning 'explanation', and that's on clang.

"I'm not going to pull this. The clang warning fix is wrong, and then another commit literally disables accounting for another non-fatal run-time warning.

"Again – warnings are not an excuse to just 'mindlessly shut up the warning'.

"They need some thought.

"None of this kind of 'I'll do wrong things just to make the warning go away' garbage that this pull request has two very different examples of.

"I'm adding some clang people, because apparently that

```
note: cast one or both operands to int ⮐
      to silence this warning
```

"thing came from clang. Somebody in the clang community really needs to re-think their 'informational' messages.

"Giving people those kinds of insane suggestions is a disservice to everybody. Clang should fix their stupid 'note' before release. Please, guys."

And that was that. ■■■

Finding a path to energy-efficient software

Clean It Up

Sustainability studies for the IT industry often ignore the contributions of software. This article explores what developers and admins can do to create and maintain more energy-efficient systems.

By Christoph Meinel

The digital transformation has taken hold in all the corners of our culture, from business to our personal lives. Virtually nothing works without digital technologies, and for that reason, it is clear that we won't achieve the sustainable development goals (SDGs) set out by the United Nations unless we rethink our approach to our digital life.

The amount of energy required to operate the world's countless devices, networks, applications, and data centers is immense. Numerous studies have shined light on the increasing appetite for energy in the IT industry, and experts point to a need to massively improve the energy efficiency and sustainability of IT systems.

The problem with the increasing CO2 footprint in IT has been known for some time and has led to various initiatives that fall under the green IT umbrella. However, the main focus of this movement, especially in the data center sector, is on reducing the waste of natural resources in digital devices, using renewable energy sources, and a call for *digital fasting* – the practice of spending part of the day or week away from digital technology.

From the beginning, the emphasis has been on the hardware. As early as 1992, the U.S. Environmental Protection Agency (EPA) and the EU Commission (2003) introduced the Energy Star label for energy-efficient IT equipment and computers. Despite all efforts, energy consumption has continued to rise steeply. The testing procedures for the Energy Star label are neither sufficiently rigorous nor thorough, and the rapidly increasing share of carbon dioxide emissions caused by the use of software and digital applications running on these devices is largely ignored. Algorithmic efficiency and sustainable computing are proving to be a major blind spot in most Green IT initiatives, which focus on the production and operation of the devices but ignore the daily emissions that far-too-energy-hungry software generates over the device's lifetime.

A study by The Shift Project shows that, as early as 2017, energy consumption from the use of digital technologies exceeded that in the production of digital devices by more than five percent, with a steadily increasing share. This suggests that, in the future, measures to reduce the CO2 footprint of digital technologies needs to focus more on improving the energy footprint of software systems and their interaction with hardware, rather than just focusing on the energy footprint of hardware [1].

The quest for more sustainable IT will require us to systematically study the the energy consumption of digital systems and make the figures comparable. Then we need to develop methods for designing IT systems that are more energy-efficient. To this end, we need to combine the principles of algorithmic efficiency and sustainable computing in a fundamental paradigm of Sustainability by Design.

Massive Rise

In recent decades, digital technologies have been hailed as the "clean" counterpart to the "dirty" industries of manufacturing, agriculture, and energy production. Digital devices, products, and services were believed to contribute little or nothing to global CO2 emissions because of their intangible nature. This assumption is wrong: Globally, digital devices and applications have a very significant CO2 footprint.

All data traffic requires energy. The total annual Internet traffic volume has increased exponentially in recent years and continues to rise steeply. In 2007, only 54 exabytes of data passed through the Internet. By 2017, this number had increased by a factor of 20, to 1.1 quintabytes, according to the International Energy Agency. By the end of 2022, annual data traffic is projected to reach 4.2 zettabytes [2]. Already, carbon emissions from digital technologies exceed those from global air travel by a factor of two. In 2019, for example, all air travel was responsible for about 1 billion tons of carbon dioxide emissions, about two percent of total emissions. In the same year, digital technologies emitted about 2 billion tons, about four percent of total human-generated carbon dioxide [3].

An example of technology that is considered very new age and progressive from a technology perspective but that is actually quite regressive in its energy usage, is Artificial Intelligence. Researchers at the University of Massachusetts Amherst have studied the energy consumption of modern AI systems and found that the training phases for new neural networks, in

particular, consumes a significant amount of energy and, as a result, emits huge amounts of carbon dioxide. For example, training a common AI model with Big Data produces about 300 tons of carbon dioxide equivalents, which is like the CO2 lifecycle emissions of five cars, including fuel, or 300 round-trip flights from New York City to San Francisco [4]. (Cryptomining is another huge consumer of electrical power – see the article on Cryptomining elsewhere in this issue.)

But before you rule out AI, keep in mind that many believe AI and Big Data are fundamentally important for *reducing* CO2 emissions overall because of their importance for optimizing processes in energy production, manufacturing, agriculture, and other industries. The solution is not to eliminate or "fast" from these technologies but to make them more efficient.

Sustainability by Design

Programmers can help reduce the growing CO2 footprint by making more efficient software systems. The goal is to develop software that uses less energy to deliver practically equivalent results – with as few, and as simple, computing operations as possible. Further potential for reducing energy consumption lies in the implementation of the underlying algorithms.

When developing innovative software architectures, you can consider trade-offs between precision in the computational results and reduced energy usage. This is especially true of systems that are used millions of times – even the smallest savings in the individual computational processes add up to significant savings.

Because the use of digital technologies already accounts for the largest share of the digital CO2 footprint, and will continue to rise sharply, it is particularly important to make algorithms more efficient. Greater consideration must be given to the trade-off of precision and speed on one hand and energy consumption on the other. Weighing this balance must become a core principle in the design of digital systems.

To solve the apparent paradox of more digitalization using less energy, we need to develop new paradigms in algorithm design and programming, and we need widespread implementation of these paradigms in practice. Principles such as energy-efficient algorithms and sustainable computing must also play a major role in the education of computer scientists and IT engineers. The focus must be on raising awareness of the issue of energy efficiency in software systems and making the principle of sustainability by design the basis for IT development practice.

Programming and the Quest for Cleaner IT

Typically, computer scientists and IT engineers develop software that aims to solve different classes of problems using algorithms. By their very nature, different algorithms can be developed for very similar problems. If these algorithms need to be very fast (when computing the results) and accurate (100 percent accuracy), this can require a significant amount of computation and a long runtime. Computational overhead and long run times translate to high energy requirements, and algorithms for best solutions are often extremely complex. For very large-scale problems, such as complex climate models or traffic predictions, the computation time scales endlessly depending on the desired accuracy.

One important field of research in algorithm engineering focuses on solving the problem using tradeoffs between accuracy and runtime. Appropriate algorithms use heuristics and randomized approaches to produce a result that is as close as possible to the optimum solution but that can be computed with far less overhead and runtime. These "second-best" algorithms, which usually come very close to the exact solution, can shorten the runtime of algorithms by factors of between 100 and 10,000, depending on the problem class.

Experiments at the Hasso Plattner Institute for Digital Engineering (HPI, Figure 1) have shown that applying heuristic algorithms to optimize submodular functions suitable for solutions that optimize traffic, allocate raw materials in production, or allocate goods to markets can reduce the runtime by three-digit factors compared to traditional algorithms. Whereas traditional software might take two days to compute the solution to such a problem, a heuristic algorithm might compute the task in just 10 minutes, reducing energy consumption to less than one percent.

Figure 1: Researchers at the Hasso Plattner Institute are working on energy-efficient data profiling. © HPI

The following are three examples of programming techniques that could lead to significant energy savings if widely implemented.

1. Energy-Efficient Data Profiling

Digital applications, like many new smart technologies, require perfectly organized mass data sets. However, the larger the volume of data, the more time and energy you need to process it. One of the main tasks of data engineers is to automatically organize data in a meaningful way so that the data sets can be used for artificial intelligence applications and other forms of analysis.

"Pure" data has no value. To make an impact, you need to categorize data in meaningful ways (explore the nature of the data), normalize the data (homogeneous structure), get rid of redundancies, and so on. Metadata plays an important role in this kind of task because it helps organize data for the value chain. In this sense, organizing data sets is the basis for all data-driven digital products and services.

Unique Column Combinations (UCC) are an important aspect of data profiling. Successfully identifying UCCs can lead to more efficient processing for database systems. Until recently, it was only possible to identify UCCs for small to medium-sized data sets – and with considerable time overhead. For large data sets, UCCs were difficult to discover due to runtime and memory restrictions. The people at HPI have developed a novel algorithm for discovering UCCs (HPIValid) that reduces the UCC discovery runtime by several orders of magnitude while reducing the memory requirements when compared with other algorithms.

On medium-sized data sets, HPIValid was 5 to 100 times faster and consumed only 5 to 20 percent the amount of memory on average, while efficiency decreased for larger data sets. Although the previously used HyUCC algorithm was not able to detect UCCs in extremely large datasets, HPIValid can perform the computation within a reasonable runtime and with a reasonable memory consumption, making it possible to indentify UCCs for massive datasets [5].

2. Energy Efficiency for Artificial Intelligence

The development of machine learning techniques and deep neural networks have meant crucial advances in the field of AI research. However, the ever-improving deep learning algorithms translate to an ever-increasing need for energy, during training in particular, but also during execution.

Modern machine learning systems train neural networks based on 32-bit algorithms like ResNet. However, reducing the complexity of AI models using quantization and pruning techniques can save energy. Rounding data values in deep learning models drastically reduces the energy consumption of AI systems.

In the extreme case, deep learning networks can be executed with binary neural networks (1-bit algorithm). This approach reduces the effort in the individual computing steps and immediately generates energy savings with a factor of 20. Although binary neural networks are currently still about five percent less precise than today's best AI systems, they impress with their 95 percent reduction in energy consumption. When used millions of times a day, enormous amounts of energy can be saved.

Table 1 shows the significant reduction in model size and number of operations of three variants of binary neural networks compared to 100 percent accurate 32-bit ResNet networks [6]. The loss of accuracy is moderate.

3. Energy-Aware Computing at the Data Center

Data centers are considered to be the heart of digitalization. Cloud applications, streaming, complex simulations – everything runs in a data center. The resulting, ever-increasing energy consumption contributes significantly to the global CO2 footprint. Next-generation data centers include an increasingly diverse landscape of accelerators and hardware architectures, each offering advantages for specific classes of algorithms or application areas.

However, today's data center architecture and software largely ignore this level of heterogeneity. Running workloads on the most appropriate hardware can significantly improve energy efficiency. In a preliminary assessment at a research seminar on energy efficiency, participants were able to improve energy efficiency by a factor of more than 10 for weather simulation models by using Field Programmable Gate Array (FPGA) accelerators instead of general-purpose processors.

In this case, a weather simulation ran on different processors with different results (see Table 2). For this particular

Table 1: Energy Saving in Deep Learning

Model Name	Accuracy	Size	Operations
ResNet (32-Bit)	69.3 Percent	46.8MB	$1.61x10^9$
BinaryDenseNet45	63.7 Percent	7.4MB	$3.43x10^8$
BinaryDenseNet37	62.5 Percent	5.1MB	$2.70x10^8$
BinaryDenseNet28	60.7 Percent	4.0MB	$2.58x10^8$

Table 2: More Efficiency with Special Processors

Device	NVIDIA Tesla K20Xm	Intel E5-2630 v4	Xilinx XCKU060
Problem size (MCells)	256	256	1024
Throughput (MCells/s)	1127.88	1435.48	2209.35
Consumption (Watts)	60	85	9.5
Efficiency (MCells/Ws)	18.80	16.89	232.56

task, the Xilinx processor showed higher energy efficiency than other processors. However, other tasks require different specialized hardware. Similarly, Qasaimeh et al. [7] demonstrated a 20-fold improvement in energy efficiency when using FPGAs for certain computer vision tasks. In contrast, computational tasks that rely heavily on floating-point math can often achieve higher performance and better energy efficiency on GPUs [8].

This research opens up the question of how to optimally distribute computing operations in data centers based on heterogeneous computing resources in order to reduce energy consumption.

Conclusion

Digitalization is a significant contributor towards greenhouse gas emissions. At the same time, the use of digital technologies is key to curbing CO2 in other industrial sectors. Until now, IT system development has virtually never been looked at from the point of view of energy efficiency. But doing so offers a huge opportunity to build digital systems that consume significantly less energy. To this end, principles of sustainability must find their way into the curricula of computer science courses and must be part of the discussion from the outset when considering the energy balance of IT systems.

HPI has launched the clean-IT Initiative to educate people in the possibilities of reducing the CO2 footprint in the field of digitalization and, in cooperation with national and international partners from science, industry, and civil society, to demonstrate solutions for how to succeed with sustainable digitalization. The clean-IT Forum [9] on HPI's interactive learning platform presents examples and best practices on energy-efficient algorithms and sustainable computing, describing very specific measures for developing algorithms, data centers, and AI systems in an energy-efficient way (Figure 2).

Figure 2: The Hasso Plattner Institute's (HPI's) clean-IT Forum has offered sustainability experts from around the world a platform for exchanging ideas since April 2021. © HPI

Research on energy consumption of digital systems is still in its infancy. As of now, hardly any relevant scientific conferences and journals exist that explicitly address the issue of energy-efficient software and sustainable computing. There is also a need for initiatives that will start transforming research results into scalable energy-efficient software products.

Policymakers could launch incentives to strengthen research in the area of clean IT. This research could be the basis for reliable criteria that could assess whether a digital application is sufficiently energy efficient. International certifications could accelerate the spread of energy-efficient IT systems. Above all, the public sector, as the largest purchaser of IT systems in many countries, needs to stipulate in its procurement guidelines a requirement for energy-efficient software in order to bolster the market for sustainable software and IT systems. ■■■

Info

[1] The Shift Project 2019: *https://theshiftproject.org/wp-content/uploads/2019/03/Lean-ICT-Report_The-Shift-Project_2019.pdf*

[2] IEA (2017), *Digitalisation and Energy*, IEA, Paris, *https://www.iea.org/reports/digitalisation-and-energy*

[3] Carbon footprint of aviation: *https://www.atag.org/facts-figures.html*

[4] Carbon footprint of digitization: *https://arxiv.org/pdf/1906.02243.pdf*

[5] HPIValid: *http://www.vldb.org/pvldb/vol13/p2270-birnick.pdf*

[6] J. Bethge et al. "BinaryDenseNet: Developing an Architecture for Binary Neural Networks," *2019 IEEE/CVF ICCVW*, 2019, pg. 1951-1960, *http://openaccess.thecvf.com/content_ICCVW_2019/papers/NeurArch/Bethge_BinaryDenseNet_Developing_an_Architecture_for_Binary_Neural_Networks_ICCVW_2019_paper.pdf*

[7] M. Qasaimeh et al. "Comparing Energy Efficiency of CPU, GPU and FPGA Implementations for Vision Kernels," *2019 IEEE ICESS*, 2019, pg. 1-8, *https://ieeexplore.ieee.org/document/8782524*

[8] J. Cong et al. "Understanding Performance Differences of FPGAs and GPUs," *2018 IEEE 26th Annual Symposium on FCCM*, 2018, pg. 93-96, *https://doi.org/10.1109/FCCM.2018.00023*

[9] clean-IT Forum: *https://www.open.hpi.de/clean-it-forum*

Author

Professor Christoph Meinel is the director and CEO of the Hasso Plattner Institute for Digital Engineering gGmbH (HPI) and Dean of the Digital Engineering Faculty at the University of Potsdam, where he heads the Department of Internet Technology and Systems. A member of the German National Academy of Science and Engineering (Acatech), he developed the first European MOOC platform, openHPI, leads the BMBF-commissioned School Cloud project, and is program director of the HPI-Stanford Design Thinking Research Program.

Assessing the environmental impact of software

Efficiency Angel

Germany created Blue Angel, the world's first eco-label for software, back in 2000. The methodology behind Blue Angel could serve as a model for other countries as governments turn their attention to the environmental impact of software. *By Marina Köhn*

It is one of those annoying things that I hope consumers will never get used to, nor should they accept it. A device that costs a large amount of money is suddenly no longer of use simply because the required security update is not available. Or perhaps the new application software needs more powerful components or the interfaces are incompatible with other devices? In all these cases, consumers have no option but to replace devices that are actually still working. In other words, the software forces an unnecessary upgrade to the hardware. This common scenario can lead to significant waste of energy and raw materials.

The greatest environmental impact from information and communications technology products happens during the manufacturing process. A rumor still persists that replacing existing technology with more energy-efficient technology is good for climate protection, but the truth is, considerably higher CO2 emissions are produced during manufacture of that "energy efficient" device than during use, so in many cases, the best thing you can do for the environment is keep using the device you already have rather than drive demand for more production.

Now you could argue that short replacement cycles would not be so bad if accompanied with an effective program for recycling. But, in practice, recycling processes do not exist for some metals, which means that valuable materials are irretrievably lost when you throw away a computer. Large quantities of electronic scrap ends up in landfills.

Until now, the focus of regulation, research, and manufacturer activities has been exclusively on hardware. Environmental labels, voluntary commitments, and minimum legal requirements created the foundation on which laptops, computers, and monitors were able to become significantly more energy efficient in recent years.

Unfortunately, a similar success story cannot be told when it comes to software. Certainly some software is developed by teams that place an emphasis on long service life, but this is by no means true of all software products. Negative examples include programs that force you to disable the computer's power management to avoid data loss, applications that become so bloated with updates that they respond to commands very slowly or crash frequently, and software that consumes more energy than is actually necessary. This list of bad features could easily go on.

Keep in mind that there are software features that reduce the environmental impact during use. The design of the software architecture determines how much hardware and electrical energy you need. For example, depending on how intelligently it is programmed, the software might require less or more processing power and memory. Figure 1 illustrates the various effects that software has directly on hardware and indirectly on the environment.

The path of the search for effective environmental criteria has been rocky and difficult. Why has software flown under the radar until now? Why is there so little research and regulation on this subject? There are at least two crucial reasons for the lack of attention to software's environmental effects. On one hand, the relationship between cause and effect is far less easily discernible for software than for hardware, making it difficult to set verifiable minimum requirements. On the other hand, the large number of different software products and areas of application make it difficult to assess the full scope of the problem.

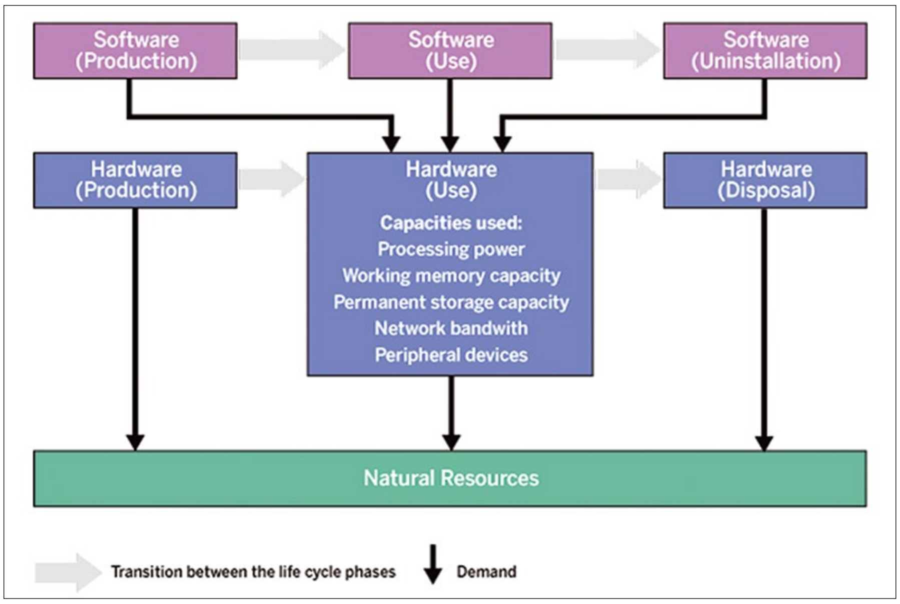


Figure 1: Model for understanding the effects of software on the environment. All phases of the software lifecycle can have an effect on hardware usage. © ETH Zurich

In spite of the obstacles, the German government is currently developing a methodology that will allow it to rate software for its environmental impact. The Blue Angel eco-label developed by German scientists could serve as a model for similar rating systems in other countries.

The Research

In a recent research project [1], Germany's Federal Environment Agency, in collaboration with the Öko-Institut, the Environmental Campus Birkenfeld of the University of Trier and the University of Zurich, initially undertook a classification of various software products (Figure 2). With the help of this classification and a detailed impact model showing the interactions between hardware and software, the researchers developed a catalog of criteria for sustainable software. In total, the catalog lists 76 individual criteria, which it groups under three categories: resource efficiency, influence on the duration of the hardware use, and autonomy of use. Each criterion can be evaluated by reference to one or more indicators.

The project developed an evaluation and measurement methodology that you can use to determine the energy requirements, hardware resource usage, and other environmental characteristics of software. The research report examined software products with the same functionality from four different product groups to assess the methodology. In a standard scenario, in which the typical functions run automatically in a defined sequence, the differences between the software products were very much apparent (Figure 3). The energy consumption of the least efficient software (bottom) is four times higher than that of the better-performing product (top). The difference in energy usage is particularly relevant in light of the fact that excessive use of hardware in the inefficient candidate leads to slower execution. In a worst-case scenario, the user will just throw away the computer and replace it with a newer, faster model, when all they really needed to do was replace the software.

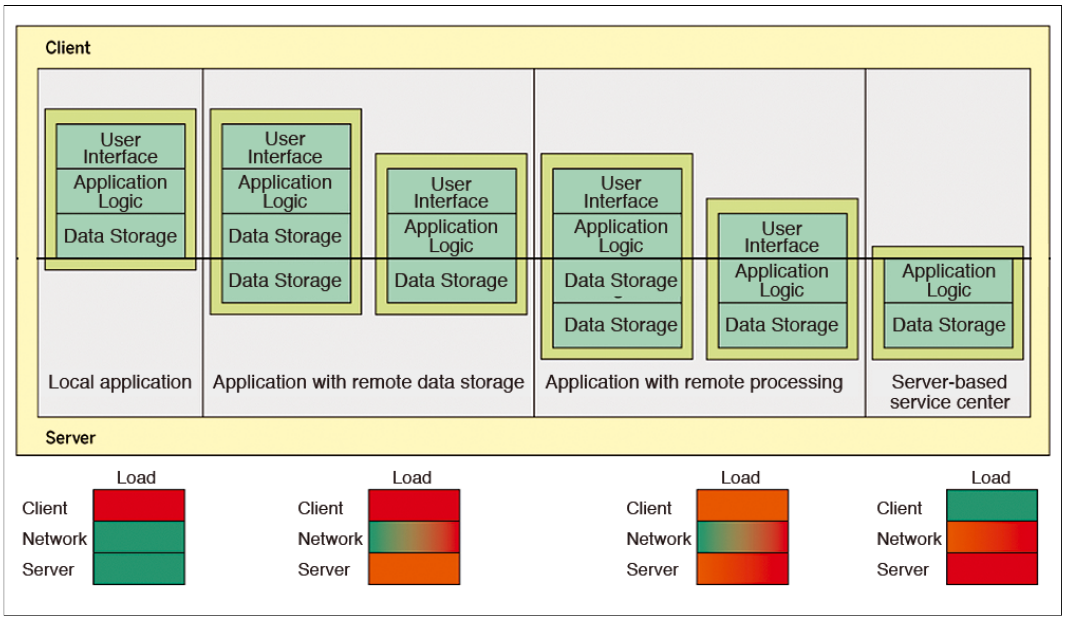

Client
User Interface
Application Logic
Data Storage
Local application
Application with remote data storage
Application with remote processing
Server-based service center
Server
Load
Client
Network
Server

Figure 2: Classification of application software.

The First Eco-Label

Germany's Federal Environment Agency has used the results of this study to develop the Blue Angel eco-label for software. To be eligible for the Blue Angel, software must still run on a computer that is at least five years old and ensure security updates for at least five years. It must not interfere with the computer's energy management, and it must disclose the data formats, so that future use of the data is possible. In addition, the applicant must disclose the software's energy consumption and hardware resource utilization test results. To this end, the German government has developed a test method, including software that can be used to determine energy consumption data and hardware utilization.

The first version of the eco-label imposes requirements for local application software (software intended for computing and storage on the local computer). Ongoing research projects will expand the scope of Blue Angel certification to include distributed software and applications for smartphones or tablets.

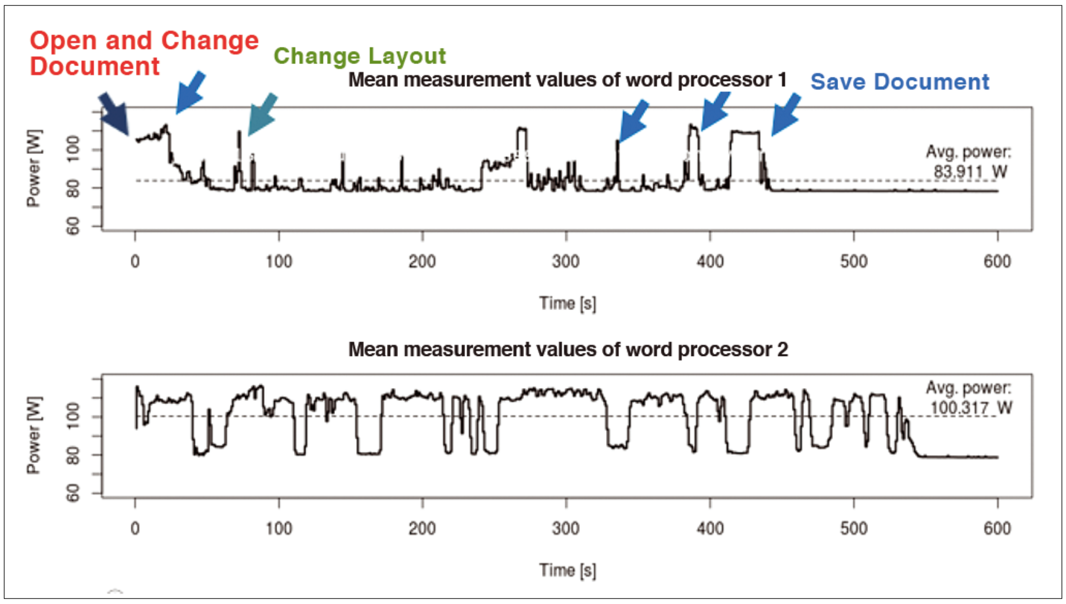


Figure 3: Sample test result for two word processing programs.

Outlook

With the results of its research, the Federal Environment Agency has opened up a view window onto the environmental impact caused by software. But the real challenge is not to rate the software after it is created but to provide the information necessary for developers to make the right decisions early in the software planning and development phase, supporting optimized tools and methods that result in environmentally friendly software.

Another project of Germany's Federal Environment Agency, the SoftAWARE Research project, launched in 2021, is intended to contribute to the goal of encouraging more sustainable software design. The project, which is a collaboration with the Sustainable Digital Infrastructure Alliance and the Öko-Institut, will examine the energy efficiency and resource efficiency of programming languages, components, and container classes. This work should enable developers to create more energy-efficient and hardware-friendly software. This task will require information about the energy consumption and hardware usage of software components, as well as development tools, optimization information, and recommendations for an improved energy efficiency index. The project will also sponsor a hackathon next year to find solutions to documented problems and to award prizes to the most energy-efficient software solutions. ■■■

Author

Marina Köhn is a computer scientist who has worked as a researcher with Germany's Federal Environment Agency since 1992. Her work focuses on environment-related system comparisons, particularly in the field of information and communication technology (ICT). Köhn has been working on issues relating to green IT for over 20 years. As part of this activity, she participated in developing the Blue Angel system and other methods for measuring the energy and resource efficiency of data centers, cloud services, and software.

Info

[1] Development and application of evaluation principles for resource-efficient software: *https://www.umweltbundesamt.de/en/press/pressinformation/environmental-impact-of-software-is-now-measurable*

Assessing the environmental impact of crypto mining

Gold Rush

With climate change wreaking havoc across the planet, we were just beginning to think about how we could use our technology to conserve fuel and reduce our carbon footprint. Then along came crypto mining.

By Jack Wallen and Joe Casad

A study of power consumption in IT would not be complete without a look at one of the biggest offenders – and at a problem that has grown exponentially in recent years: cryptocurrencies. To the user, cryptocurrencies are an efficient, decentralized way to send money around the world, but behind the scenes, the infrastructure that supports crypto mining is consuming power at an alarming rate.

Bitcoin mining alone consumes approximately 91 terawatt-hours of electricity per year [1]. A single US home uses approximately 11,000 kilowatt-hours per year, which means that bitcoin mining consumes as much energy as 8 million homes – more than seven times the total amount of used by Google – and that number is growing every year as Bitcoin gains popularity. According to a recent study [2], if you take the total energy cost of bitcoin mining divided by the total number of bitcoin transactions, every bitcoin purchase has an energy cost of over $100 – even if you're just buying coffee or flagging an Uber.

And keep in mind that Bitcoin is only one of several competing crypto technologies. Overall, the electricity used for crypto mining is about half a percent of all electricity used across the globe (or the same amount of energy used to power the state of Washington for a year). That number has increased by a factor of 10 over the past five years. Obviously, this level of power usage is not sustainable – especially if cryptocurrency becomes the dominant form of currency exchange, as some experts predict.

What's the Problem?

Blockchain technologies such as Bitcoin work by adding *blocks* (groups of transactions) to the *chain* (the complete trail of all blocks, recording all transactions since the beginning of the currency). The blocks are signed and interconnected through cryptographic hash values. The details are quite complex, but for the purposes of this discussion, computer installations called bitcoin miners perform the task of guessing a hash value that will validate the block and ensure that the block can be added to the chain. The method used with Bitcoin and other leading cryptocurrencies is called proof of work. In a proof-of-work scenario, miners race each other to discover the correct value, and whoever gets there first is rewarded with bitcoins (BTC). The current reward for mining a new block is 6.25 BTC. At this moment, one bitcoin is worth around $42,000, so this reward is quite significant (more than $262,000), although Bitcoin miners often band together to share the work and split the prize. There is a lot of incentive to get there first, which has caused a rush to larger and more powerful computer systems.

Another feature built into the Bitcoin algorithm is that the puzzles used to validate the blocks are not supposed to get solved too quickly, so as the computers mining bitcoins get faster and more powerful, the problems automatically get more difficult. This increasing difficulty further expands the hardware and escalates power usage.

And of course, the rising value of Bitcoin builds in a big incentive for more powerful systems. By design, Bitcoin inventor Satoshi Nakamoto capped the number of bitcoins at 21 million units to make the currency scarce and control the inflation that would inevitably arise from an unlimited supply. This scarcity, and the current demand for bitcoins, has brought on a price that Satoshi Nakamoto probably never imagined when the system was invented. The amount of the reward halves after the creation of every 210,000 blocks (which equates to roughly four years), so the reward in bitcoins has dropped over the years, but the equivalent reward in dollars has exploded due to the explosion in the Bitcoin price.

That halving of the reward means that the best time to mine bitcoin is *right now*, not later, which only adds to the current frenzy. It could also mean that, after the next halving, miners will need to throw even more hardware at the problem to try to earn an equal amount.

To stay competitive, companies must employ the newest and fastest hardware to keep up. Another effect of the cryptocurrency gold rush is that hardware turnover is ramping up. As machines become obsolete, more and more electronic waste is produced. According to Digiconomist (Figure 1), the annual total electronic waste created by cryptocurrency mining is at 34.14 kilotons per year, or 382.6 grams of e-waste per transaction [3].

What Kind of Energy

When it comes to long-term climate effects, the question isn't just "how much energy?" but also "where did the energy come from?" Bitcoin mining that takes place in areas that rely on wind, solar, and hydropower has a smaller carbon footprint than mining in areas that depend on coal. However, the difference between "clean" and "dirty" bitcoin mining is sometimes difficult to quantify, given the interconnected nature of the electrical grid. A study in the energy journal *Joule* [4] reports that China's recent ban on crypto mining has forced miners to relocate to neighboring areas with less renewable power, resulting in an overall increase in the global carbon footprint of 17 percent.

Another recent development is to use energy that is currently going to waste for crypto mining. For instance, enterprising miners have begun locating portable crypto-mining rigs near oil fields to use the energy that is currently burned off as waste through a natural gas flare [5]. In that case, the carbon footprint is high because the energy source is natural gas, but the rig is running on energy that would otherwise have gone to waste. These kinds of innovations help, but the scale is too small to make an impact on the overall trend leading to more energy consumption for crypto mining.

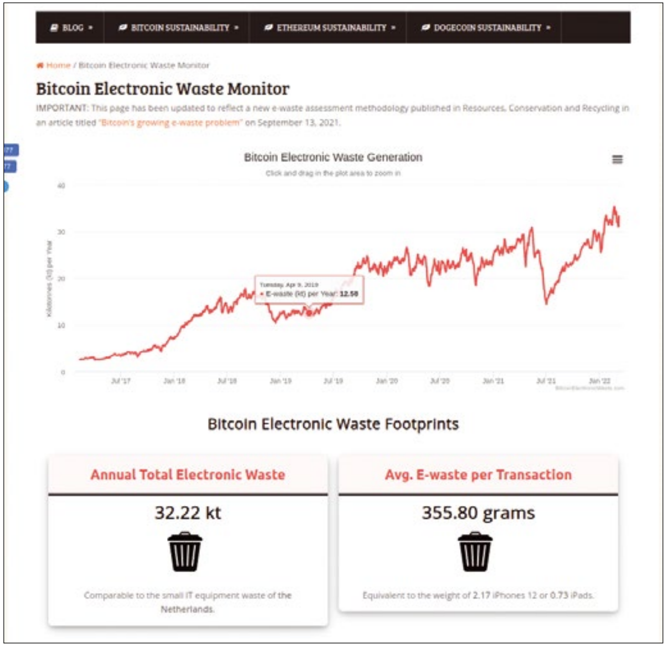
What Is Being Done?

Governments around the world are starting to wake up to the need to address the energy cost of crypto mining. China has banned crypto mining outright. As this issue goes to print, the Biden administration has just released a comprehensive executive order on cryptocurrencies, which includes a directive to "support technological advances that promote responsible development…" including the need to reduce climate impacts. Meanwhile US, Senator Elizabeth Warren and others have raised the alarm about the energy cost [6], and a bipartisan bill to provide for more regulation of the cryptocurrency industry is currently being debated. In the state of New York, a bill to establish a moratorium on proof-of-work crypto-mining techniques is currently in committee [7]. In the EU, European Securities and Markets Vice-Chair Erik Thedéen called for a ban on proof-of-work crypto mining [8], stating that too much of the renewable energy gain in his native Sweden is getting used up by bitcoin mining, making it difficult for the country to migrate its overall footprint to more renewable sources.

Several alternative cryptocurrencies make an effort to address sustainability issues in various ways. According to LeafScore (Figure 2), a site that tracks and ranks cryptocurrencies by sustainability [9], the most sustainable cryptocurrency is SolarCoin [10], which creates one SolarCoin for every megawatt-hour generated from solar technology. This cryptocurrency is a service that rewards users in cryptocurrency for solar installations. The company goal is to incentivize solar electricity production.

The next on the sustainability list is Powerledger [11], which offers a customized design based on the Solana blockchain to take advantage of the Solana's scalability. Powerledger has partnered with Midwest Renewable Energy Tracking System to help facilitate the trending of Renewable Energy Certificates (a market-based instrument that certifies the bearer owns one megawatt-hour of electricity that was generated by a renewable source).

As of now, these sustainable cryptocurrencies trade at very low volumes. At this time, the best way to truly make a



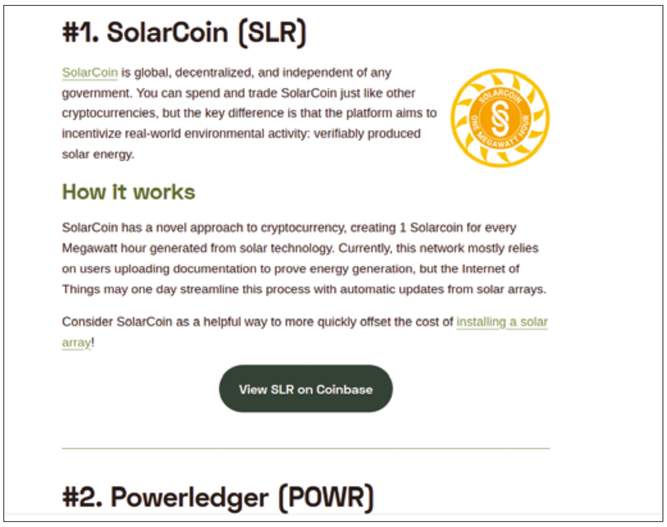
Bitcoin Electronic Waste Monitor

IMPORTANT: This page has been updated to reflect a new e-waste assessment methodology published in Resources, Conservation and Recycling in an article titled "Bitcoin's growing e-waste problem" on September 13, 2021.

Bitcoin Electronic Waste Footprints

Annual Total Electronic Waste	Avg. E-waste per Transaction
32.22 kt	355.80 grams
Comparable to the small IT equipment waste of the Netherlands.	Equivalent to the weight of 2.17 iPhones 12 or 0.73 iPads.

Figure 1: The electronic waste monitor at the Digiconomist site tracks the waste caused by Bitcoin, Ethereum, and Dogecoin.



#1. SolarCoin (SLR)

SolarCoin is global, decentralized, and independent of any government. You can spend and trade SolarCoin just like other cryptocurrencies, but the key difference is that the platform aims to incentivize real-world environmental activity: verifiably produced solar energy.

How it works

SolarCoin has a novel approach to cryptocurrency, creating 1 Solarcoin for every Megawatt hour generated from solar technology. Currently, this network mostly relies on users uploading documentation to prove energy generation, but the Internet of Things may one day streamline this process with automatic updates from solar arrays.

Consider SolarCoin as a helpful way to more quickly offset the cost of installing a solar array!

#2. Powerledger (POWR)

Figure 2: The LeafScore website keeps a list of the most sustainable cryptocurrencies.

difference to the overall power usage of the cryptocurrency industry is to take on the industry heavyweights. Most experts believe that will require a change to the proof-of-work model used for mining cryptocurrencies such as Bitcoin and Ethereum. One promising option that has emerged as an alternative to proof of work is the proof-of-stake mechanism. Rather than allowing a multitude of miners to race each other to the solution, as is the case with proof of work, the proof-of-stake model calls for miners to put up a *stake*, a pledge that acts as a kind of deposit for the right to mine the block. Features built into the proof-of-stake model provide the necessary redundancy and ensure that a single operator can't monopolize the system. The security and stability of this model largely depend on the details of the implementation and how the work is incentivized.

The Ethereum cryptocurrency currently uses proof of work but is in the process of migrating to a proof-of-stake protocol. In Ethereum's version of proof of stake [12], a user must put up 32 ETH to become a validator. (An ether, abbreviated as ETH, is the unit of exchange in Ethereum – currently trading at around $2,700.) According to the Ethereum website, "Validators are chosen at random to create blocks and are responsible for checking and confirming blocks they don't create. A user's stake is also used as a way to incentivise good validator behavior. For example, a user can lose a portion of their stake for things like going offline (failing to validate) or their entire stake for deliberate collusion."

Several smaller cryptocurrencies are already using proof of stake. In 2021, Cardano [13] was reported to be the largest proof-of-stake currency in terms of market capitalization. Cardano uses the Ouroboros proof-of-stake protocol, which the Cardano project calls "… an environmentally sustainable, verifiably secure proof of stake protocol with rigorous security guarantees." Cardano believes that, thanks to Ouroborus, their network can sustainably scale to meet global demand, while consuming only 6GWh of power and without compromising speed or efficiency.

The proof-of-stake model goes a long way to reducing the amount of computation power required to verify blocks and transactions. Instead of using a competition-based mechanism, the random selection method means fewer miners will feel the need to continually use more and more compute power to stay competitive.

Unfortunately, the conversion from proof of work to proof of stake can take considerable time to implement, and as long as Bitcoin holds off on committing to the transition, cryptocurrency power usage will continue to rise.

Proof-of-stake protocols are still relatively new and untested, and it isn't yet clear if they will provide a complete replacement for proof of work. Some commentators believe that proof-of-stake protocols could lead back to more centralization. Catherine Mulligan, a professor of computer science at the University of Lisbon's Instituto Superior Técnico, says of proof of stake, "A key disadvantage is that, in some systems, you are only selecting validators that have the most money. This means that proof of stake is likely to be significantly less democratic in many cases than Bitcoin" [14]. However, other proof-of-stake systems, such as the Ethereum system, require a minimum stake to be in the game but then choose the validator randomly to avoid a bidding war. Also, although you need to put up a stake to act as a validator, you might not need to buy as much expensive hardware as you would with proof of work, so it is possible that proof of stake could be *more* democratic if implemented properly.

The other issue proof of stake will face is that it simply hasn't been proven at scale. Yes, in theory, it should have considerably less impact on the environment, but until we see it working at the same scale as proof of work, there's no way of knowing if the theory will transition smoothly to practice. The Ethereum and Bitcoin proof-of-work blockchains currently have a stored value of more than $1 trillion, whereas the proof-of-stake currencies are only beginning to attract real-world users.

Conclusion

The big question is whether proof of stake can scale with the necessary security, as well as whether it will be able to significantly lower the environmental impact of crypto mining. Proof of stake needs to be tested at scale soon; otherwise, the ever-increasing energy demands of crypto mining could have a lasting, detrimental effect. Until cryptocurrencies begin the shift away from proof of work to proof of stake and other alternative methods, the technology will continue to undermine all efforts to reduce energy usage and forestall climate change.

Fortunately, cryptocurrencies such as Ethereum are in the process of making this migration. But until Bitcoin agrees to change its ways, power usage will continue to be a serious problem in the cryptocurrency industry. ■■■

Info

[1] Bitcoin mining consumes 0.5% of all electricity used globally: *https://www.businessinsider.com/bitcoin-mining-electricity-usage-more-than-google-2021-9*

[2] Every single Bitcoin transaction – even buying a latte – consumes over $100 in electricity: *https://fortune.com/2021/10/26/bitcoin-electricity-consumption-carbon-footprin/*

[3] Bitcoin Electronic Waste Monitor: *https://digiconomist.net/bitcoin-electronic-waste-monitor/*

[4] Revisiting Bitcoin's carbon footprint: *https://www.cell.com/joule/fulltext/S2542-4351(22)00086-1*

[5] Oil drillers and Bitcoin miners bond over natural gas: *https://www.reuters.com/business/sustainable-business/oil-drillers-bitcoin-miners-bond-over-natural-gas-2021-05-21/*

[6] Elizabeth Warren Senate webpage: *https://www.warren.senate.gov/newsroom/press-releases/as-crypto mining-operations-grow-in-the-us-senator-warren-raises-concerns-over-exponentially-growing-energy-use-climate-impact-and-costs-to-consumers*

[7] New York Senate Proof of Work Bill: *https://www.nysenate.gov/legislation/bills/2021/s6486*

[8] EU regulator calls for ban on proof of work bitcoin mining to save renewable energy: *https://www.euronews.com/next/2022/01/19/eu-regulator-calls-for-a-ban-on-proof-of-work-bitcoin-mining-to-save-renewable-energy*

[9] LeafScore: *https://www.leafscore.com/blog/the-9-most-sustainable-cryptocurrencies-for-2021/*

[10] SolarCoin: *https://solarcoin.org/*

[11] Powerledger: *https://www.powerledger.io/*

[12] Ethereum Proof of Stake: *https://ethereum.org/en/developers/docs/consensus-mechanisms/pos/*

[13] Cardano: *https://cardano.org/*

[14] Proof of stake vs proof of work: *https://www.businessinsider.com/personal-finance/proof-of-stake-vs-proof-of-work*

JOIN THE LINUX REVOLUTION!
ALL THE SOFTWARE YOU NEED!
GETTING STARTED WITH LINUX
301 BEST BASH COMMANDS
LINUX SHELL HANDBOOK
SUPERCHARGE YOUR LINUX SKILLS
Dive deep into the world's greatest free office suite
LibreOffice Expert
TUNE YOUR LINUX SYSTEM
101 COOL LINUX HACKS
RASPBERRY PI GEEK
MakerSpace
HANDS-ON PROJECTS FOR MAKERS

The MusE 4 MIDI sequencer

Music Maker

MusE, a digital audio workstation, offers a free software solution for MIDI projects on Linux.

By Hartmut Noack

If you want to create music with free software on Linux, you can choose from a few digital audio workstations (DAWs). If you play live music, Ardour is usually a good choice. However, if you primarily compose music in MIDI notation with virtual or hardware synthesizers, you may want to consider MusE 4. Of the free DAWs for Linux, MusE 4 puts the most emphasis on full support for the many methods and standards that have found their way into MIDI technology over the past 50 years.

Where to Get It

A few years ago, MusE was in the standard feature set of popular distributions related to music production, which is true of Ubuntu Studio (used to test MusE in our lab). Today, the major distributions often only have an outdated version in their package sources because work on MusE has been pretty slow at times.

To get the latest version, you can download an AppImage from the MusE website [1]. AppImages (and containers like them) are not necessarily the perfect solution for real-time audio. The elaborate wrapper increases the system load and the compartmentalized image often also prevents correct integration with the JACK audio server and ASLA MIDI. However, the MusE team has done a good job of testing the AppImages very carefully. A tolerable delay at startup time is the only side effect of running MusE from the AppImage. Once Muse4 is running, it responds quickly and integrates into the environment in an exemplary manner.

You also have the option of building MusE from source code. If you go this route, you will need a complete, up-to-date build environment for Qt5, as well as about a 100 other audio development packages. If you have built Ardour, these packages will already be in place on your system. However, because the AppImage contains the current release of the source code and works very well, the overhead of compiling the source code is only worthwhile in exceptional cases.

After downloading, the AppImage only needs the right permissions to run; when it has them, the program starts up nicely (Figure 1). In our lab, it detected a JACK server that was already running, without any manual intervention.

MusE does not provide the pre-made loops and examples you may be familiar with from commercial software. Instead, MusE supports everything installed on the system, including plugins in the native Linux VSTx format. If you want to use VSTs available as Windows DLL files, you will need a converter such as yabridge [2] (Figure 2). On Ubuntu 20.04 LTS, you will need a more recent version of Wine, which you can download from the Ubuntu Wine repository [3].

Peculiarities

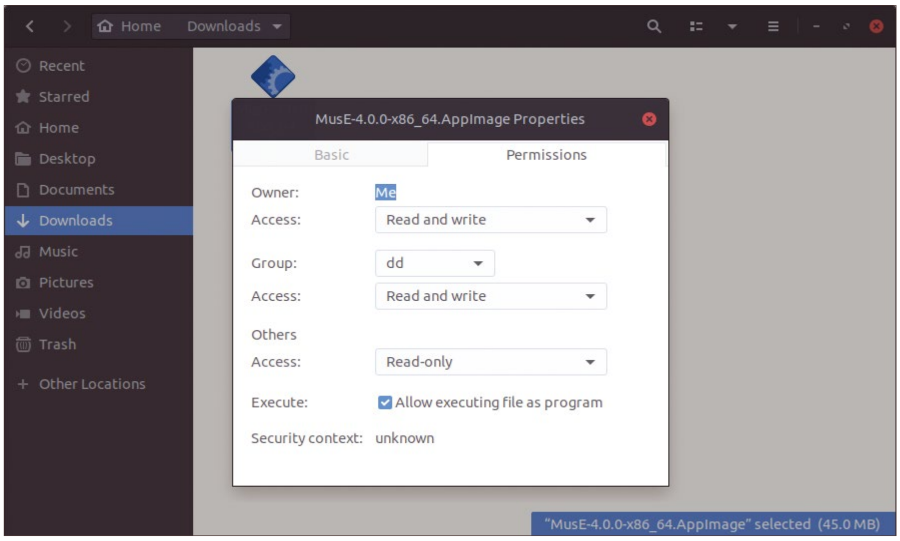
MusE follows the same principles as other DAWs: It provides an automatable tape recorder plus a signal mixer.

Photo by Marius Masalar on Unsplash

However, the way MusE implements these principles requires some getting used to for Ardour or Bitwig users.

At startup, MuseE offers to initialize external and virtual MIDI sound generators. You can send setup commands to these devices to configure them individually, which shows that MusE isolates the sound generators from the actual music data more than other DAWs do. As your first step, you need to set up the sound generators and the physical audio inputs and outputs. You can then create tracks that feed these devices with MIDI signals and audio data (Figure 3).

The configuration tool for the MIDI port connections (Figure 4) turns out to be anything but intuitive (which might cause some people to quickly give up on the program). While the configuration tool displays all possibilities, it does not let you connect the detected MIDI devices to anything in the project – despite a tooltip prompting you to do exactly that. The tooltip fails to point out that you first need to set up the devices as ports for MusE before the program offers them as inputs for MIDI tracks (see the “Setting Up a Studio” box).



MusE-4.0.0-x86_64.AppImage Properties

Basic | Permissions

Owner: Me
Access: Read and write
Group: dd
Access: Read and write
Others
Access: Read-only
Execute: ☑ Allow executing file as program
Security context: unknown

"MusE-4.0.0-x86_64.AppImage" selected (45.0 MB)

Figure 1: After making the AppImage file executable, you can launch MusE immediately.

Figure 2: Thanks to yabridge, the Feldspar freeware software synth in the form of a Windows DLL works flawlessly in MusE.

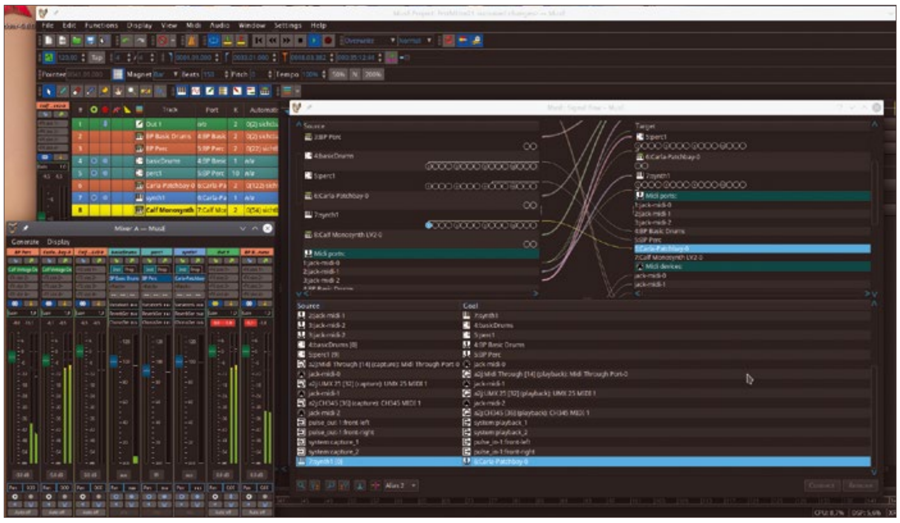


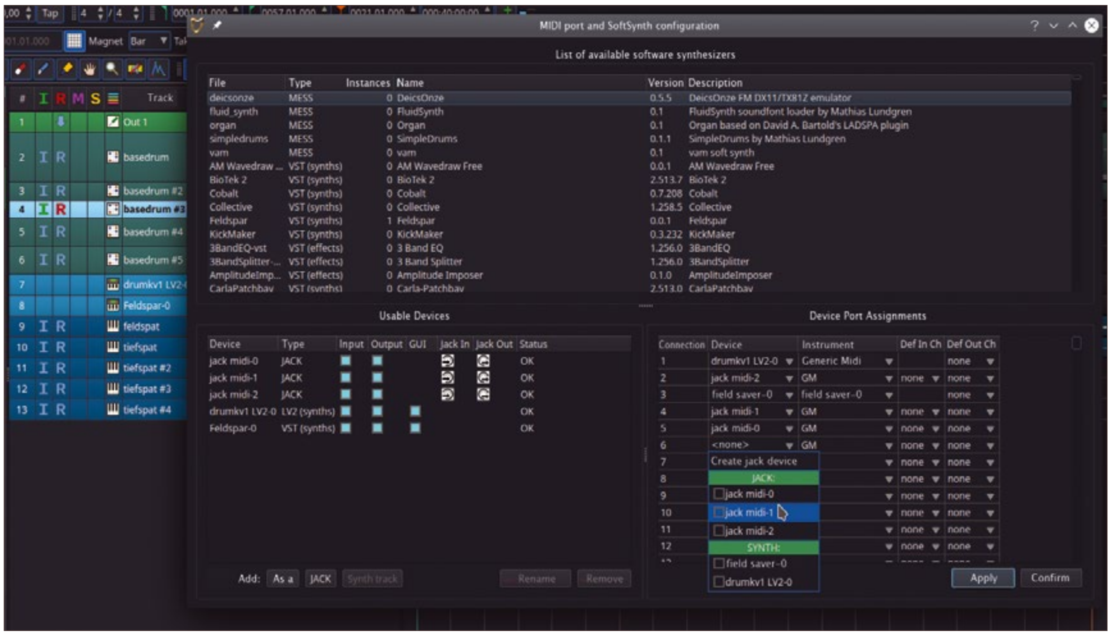
Figure 3: MusE’s routing tool lets you wire signal sources (left) and sound generators and audio outputs (right). You load plugins individually to access them from the list on the right.

Setting Up a Studio

Before you record MIDI tracks and send them to the sound generators, you need to set up a device in MusE for each generator. The device will appear as a track in the Arranger. MusE gives you two options for setting up a device – they differ greatly in some respects. The first option lets you insert synth tracks in the Arranger simply by right-clicking. When you do this, MusE displays the list of plugins that are available and supported by the system. After selecting the desired plugin from the list, you can then use it as an instrument. In the background, MusE creates a new entry in the list, which you will find in *Edit* | *Midi Ports* | *Soft Synths*.

However, the first option prevents you from using the same plugin more than once. This would be a very strange restriction if you could not use multiple instances of the same plugin. Especially with samplers, it’s quite common to have half a dozen individual noise spinners. From a purely technical point of view, this is not a problem in all of the current plugin systems.

You can get around this restriction with the second option. If you create the synth directly in the list below *Midi Ports* | *Soft Synths*, the restriction disappears. Now when you create a new instance, it also appears immediately as a Synth track in the Arranger (Figure 5).



List of available software synthesizers

File	Type	Instances	Name	Version	Description
deicsonze	MESS	0	DeicsOnze	0.5.5	DeicsOnze FM DX11/TX81Z emulator
fluid_synth	MESS	0	FluidSynth	0.1	FluidSynth soundfont loader by Mathias Lundgren
organ	MESS	0	Organ	0.1	Organ based on David A. Bartold's LADSPA plugin
simpledrums	MESS	0	SimpleDrums	0.1.1	SimpleDrums by Mathias Lundgren
vam	MESS	0	vam	0.1	vam soft synth
AM Wavedraw ...	VST (synths)	0	AM Wavedraw Free	0.0.1	AM Wavedraw Free
BioTek 2	VST (synths)	0	BioTek 2	2.513.7	BioTek 2
Cobalt	VST (synths)	0	Cobalt	0.7.208	Cobalt
Collective	VST (synths)	0	Collective	1.258.5	Collective
Feldspar	VST (synths)	1	Feldspar	0.0.1	Feldspar
KickMaker	VST (synths)	0	KickMaker	0.3.232	KickMaker
3BandEQ-vst	VST (effects)	0	3 Band EQ	1.256.0	3BandEQ
3BandSplitter-...	VST (effects)	0	3 Band Splitter	1.256.0	3BandSplitter
AmplitudeImp...	VST (effects)	0	Amplitude Imposer	0.1.0	AmplitudeImposer
CarlaPatchbay	VST (synths)	0	Carla-Patchbay	2.513.0	CarlaPatchbay

Usable Devices

Device	Type	Input	Output	GUI	Jack In	Jack Out	Status
jack midi-0	JACK	■	■				OK
jack midi-1	JACK	■	■				OK
jack midi-2	JACK	■	■				OK
drumkv1 LV2-0	LV2 (synths)	■	■	■			OK
Feldspar-0	VST (synths)	■	■	■			OK

Add: As a JACK Synth track Rename Remove

Device Port Assignments

Connection	Device	Instrument	Def In Ch	Def Out Ch
1	drumkv1 LV2-0	Generic Midi		none
2	jack midi-2	GM	none	none
3	field saver-0	field saver-0		none
4	jack midi-1	GM	none	none
5	jack midi-0	GM	none	none
6	<none>	GM	none	none
7	Create jack device		none	none
8	JACK:		none	none
9	jack midi-0		none	none
10	jack midi-1		none	none
11	jack midi-2		none	none
12	SYNTH:		none	none
	field saver-0			
	drumkv1 LV2-0			

Apply Confirm

Figure 4: In the MIDI Port and Soft Synth Configuration tool, a list located bottom right lets you map the existing devices to ports for the tracks in MusE.

and simple actions such as normalizing. However, for anything beyond that, the wave editor's design is virtually incomprehensible, and the tool sometimes crashes (see the box "Wave Editor Issues").

In comparison to the wave editor, the MIDI editor tabs are state-of-art and contain a multitude of useful functions and tools, as you might expect. The only downside is the graphical controller view at the bottom: It looks intuitive and practical, but the pencil tool only lets you edit the controller values for all notes. The tool does not take into account the selection of notes at the top of the canvas. This proves to be particularly annoying

For editing details in the recorded material, MusE offers its own editors; you can open an editor in the tabs by double-clicking on a recorded or imported *part* (a chunk of coherent notes or wave data in MusE terminology). The main Arranger view always stays open as the first tab bottom left.

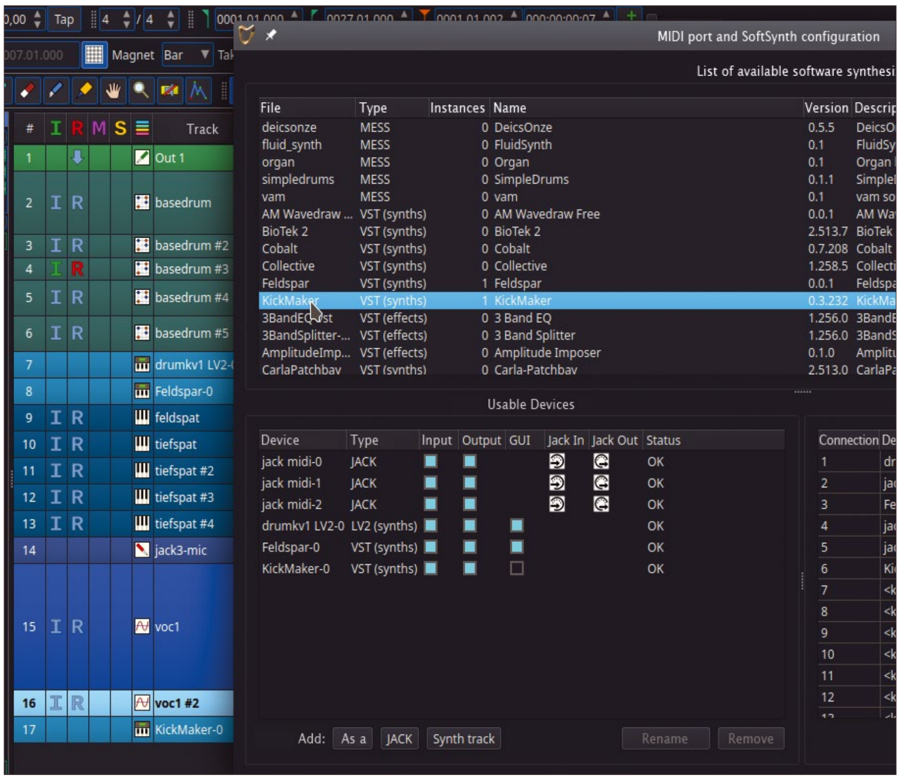
MusE removes the zoom limit in the Arranger view. Depending on the task, you will see various new toolbars and functions. MusE's wave editor, however, turns out to be very unwieldy and not very intuitive (Figure 6). After some familiarization, the wave editor can be used to perform precise cutting

Wave Editor Issues

The wave editor features two tools, time-stretch and samplerate-stretch, that behave in unexpected ways. For instance, you might expect the time-stretch tool to simply make a sound wave a little longer or shorter, but instead the tool seems to drag in material from the recorded file that is not even visible in the selected part.

The samplerate-stretch tool acts even stranger. When you first use it, the samplerate-stretch tool seems to generate useless vertical lines. However, at playback time (and only then), these lines seem to create new wave graphs that have somehow been changed in the runtime. I would have liked to explore this in more detail in our lab, but the samplerate-stretch function caused one of MuseE's rare crashes.

If the wave editor's time-stretch and samplerate-stretch tools worked as expected, they would undoubtedly be another big point in favor of MusE, because you currently won't find anything comparable in free software. Ardour's stretcher works quite well, but it does not offer the real-time preview of the new wave graph available in MusE, which is indispensable for precisely matching musical material to bars and beats in compositions.



List of available software synthesi

File	Type	Instances	Name	Version	Descrip
deicsonze	MESS	0	DeicsOnze	0.5.5	DeicsO
fluid_synth	MESS	0	FluidSynth	0.1	FluidSy
organ	MESS	0	Organ	0.1	Organ
simpledrums	MESS	0	SimpleDrums	0.1.1	Simple
vam	MESS	0	vam	0.1	vam so
AM Wavedraw ...	VST (synths)	0	AM Wavedraw Free	0.0.1	AM Wa
BioTek 2	VST (synths)	0	BioTek 2	2.513.7	BioTek
Cobalt	VST (synths)	0	Cobalt	0.7.208	Cobalt
Collective	VST (synths)	0	Collective	1.258.5	Collecti
Feldspar	VST (synths)	1	Feldspar	0.0.1	Feldspa
KickMaker	VST (synths)	1	KickMaker	0.3.232	KickMa
3BandEQ-vst	VST (effects)	0	3 Band EQ	1.256.0	3BandE
3BandSplitter-...	VST (effects)	0	3 Band Splitter	1.256.0	3BandS
AmplitudeImp...	VST (effects)	0	Amplitude Imposer	0.1.0	Amplitu
CarlaPatchbay	VST (synths)	0	Carla-Patchbay	2.513.0	CarlaPa

Usable Devices

Device	Type	Input	Output	GUI	Jack In	Jack Out	Status
jack midi-0	JACK	■	■				OK
jack midi-1	JACK	■	■				OK
jack midi-2	JACK	■	■				OK
drumkv1 LV2-0	LV2 (synths)	■	■	■			OK
Feldspar-0	VST (synths)	■	■	■			OK
KickMaker-0	VST (synths)	■	■	□			OK

Add: As a JACK Synth track Rename Remove

Connection	De
1	dr
2	ja
3	Fe
4	ja
5	ja
6	Ki
7	<k
8	<k
9	<k
10	<k
11	<k
12	<k

Figure 5: If you double-click on the KickMaker plugin entry in the list, it will be added to the project and automatically preconfigured correctly for use as a MIDI note target.

for drum notes because it is not possible to individually set the beat for a snare drum if it is in the same bar as a bass drum. While this type of important operation is not impossible in MusE, it is unnecessarily awkward.

Some important functions are missing in the menus, but you can access them via keyboard shortcuts. For example, Ctrl + B lets you insert MIDI parts as clones (Figure 7). Changes to the original have the same effect on the clones. Ctrl + V lets you create independent copies for individual editing. If you are working with bass/drum loops, working with clones is far more efficient. A note that is not quite perfect in a part will be in the right place in the clones after correcting it in the editor.

Finally, MusE's score editor offers a unique selling point (Figure 8). Even the

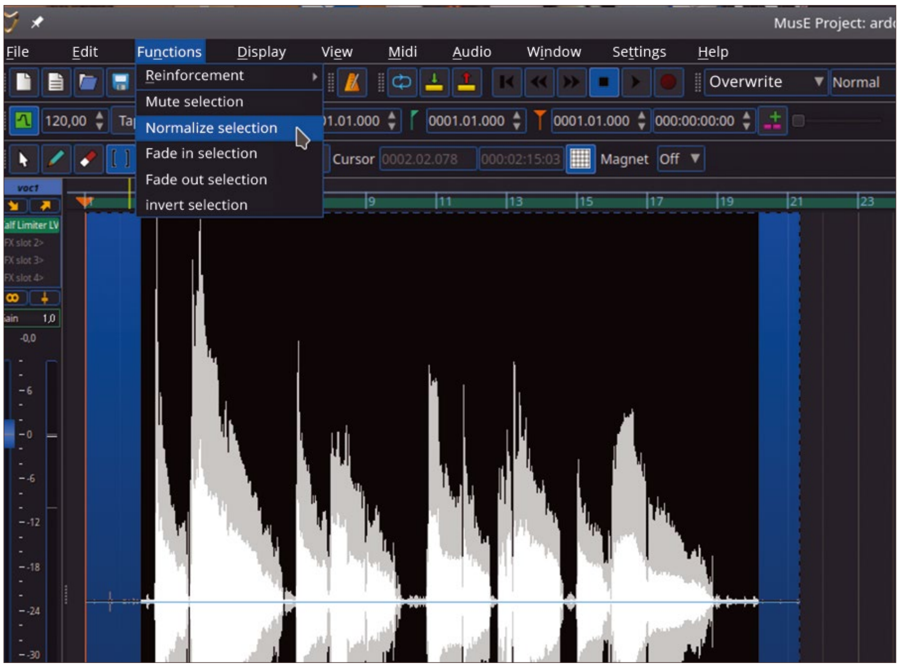


Figure 6: Selecting *Wave edit* from the context menu (right-click) lets you call up the wave editor, which is somewhat awkward to use. However, you can find important functions such as normalizing in the Functions menu top left.

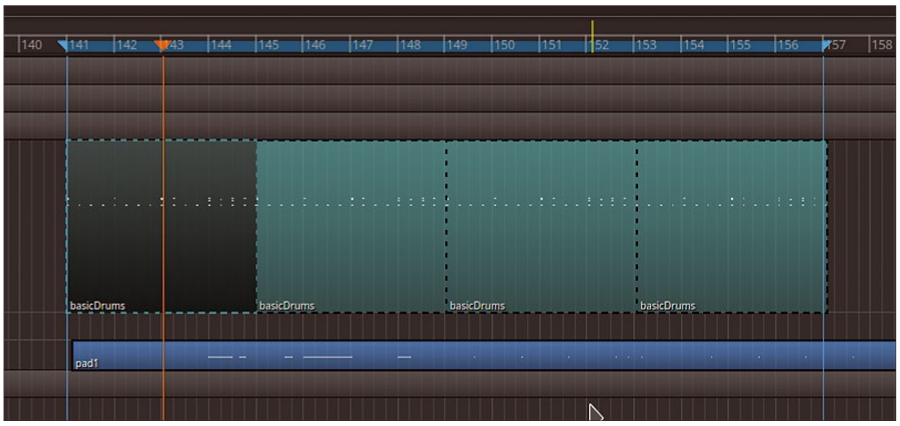


Figure 7: MusE shows clones as a dashed border instead of a solid one. Press Ctrl+B to turn the currently selected part into a clone.

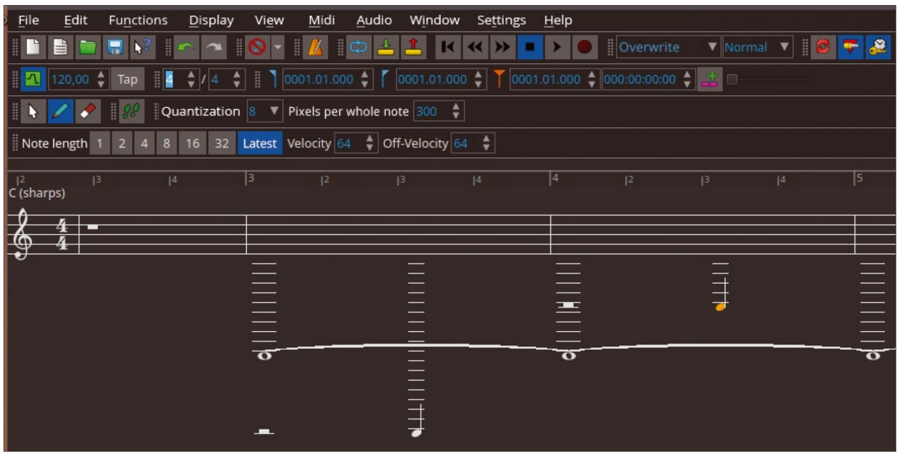


Figure 8: A part for a monophonic synthesizer in the score editor: Classic techniques such as legato are far easier to implement with the score editor than in the simple bar matrix.

History

I met the MusE project founder, Werner Schweer, in 2005 at the Linux Audio Conference at the ZKM Karlsruhe. Werner was already a very experienced developer at that time, had gathered a young team around him, and radiated a remarkable level of expertise – both in software and music theory. As a pianist at a professional level, he knows how the mathematical foundations of music work, and you definitely need this kind of knowledge to develop useful software for musicians.

The spin-off of MuseScore, which Schweer now works on, was actually a logical step. MuseScore is owned by the commercial Muse Group (see the box "Muse Confusion"). In the spirit of the Unix principle "do one thing and do it right," this was intended to separate complex music theory from the complexity of a universal DAW.

Robert "spamatica" Jonsson [5], MusE's current main developer, plays in rock and metal bands and produces quite impressive music in MusE. MusE's rather limited expansion in terms of audio processing may be related in part to Jonsson's belief that it's better to rerecord a failed recording rather than digitally reprocess poorly recorded audio tracks. What's remarkable is the perseverance of the people involved in MusE: Instead of giving up during lean periods, they kept starting over and adapting the software to the rapidly evolving environment.

expensive Bitwig Studio does not offer a classic score view, let alone a score editor. A popular alternative on Linux, MuseScore [4] (which, as the name suggests, comes from the MusE project) offers even more functionality for classical composition (see the box "History").

Many Tweaks

Many DAW functions can already be controlled directly in the MusE interface. To compensate for the rather sparsely populated main menus, however, very rich context menus pop up everywhere. The few tools that you open via the main menus offer a variety of additional functions in their own dialogs.

To save space, the function table at the top of the Arranger on the left only expands by one third in the default setting. You will find entries here that offer many functions you may have missed at first glance. One of them, for example, lets you call up the plugin's graphical interface as well automate all of the parameters in synth tracks. The ability to automate the parameters in synth tracks clarifies the purpose of the track for a sound generator because it does not contain any music data in MusE (Figure 9).

Assigning the automation curves directly to the synth makes them obsolete in the recorded tracks. If you want a different kind of automation for another recorded track, you have to create an individual plugin instance for it. If you are familiar with the playlist concept in Ardour, you will already know about this concept.

Score tracks in MusE basically work like playlist tracks in other applications: one sound generator, multiple data tracks. Somewhat confusingly, the data tracks can contain individual automations, such as those you apply with the keyboard's modwheel during recording. This is conceptually consistent because these automations do not belong to an individual synthesizer but are generic MIDI parameters, such as vibrato or volume, that are the same for each instrument.

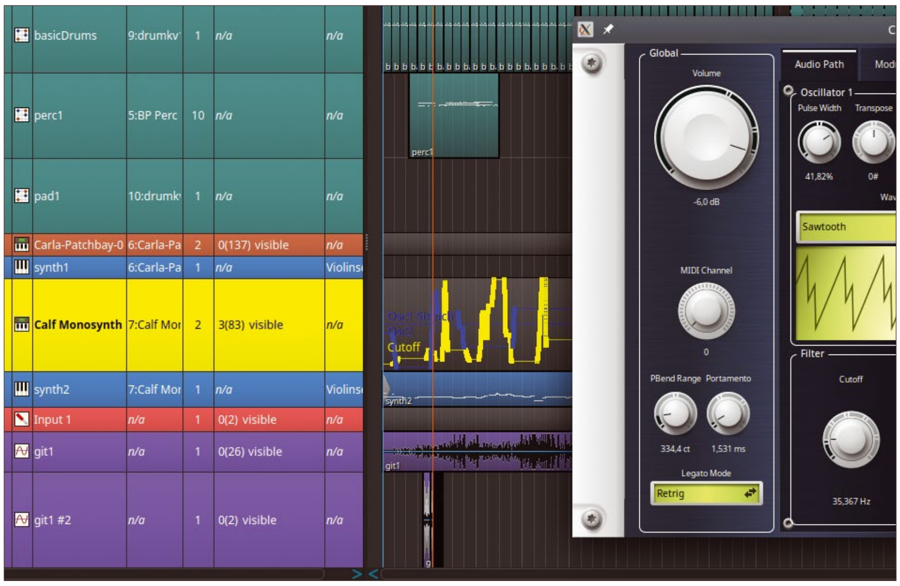
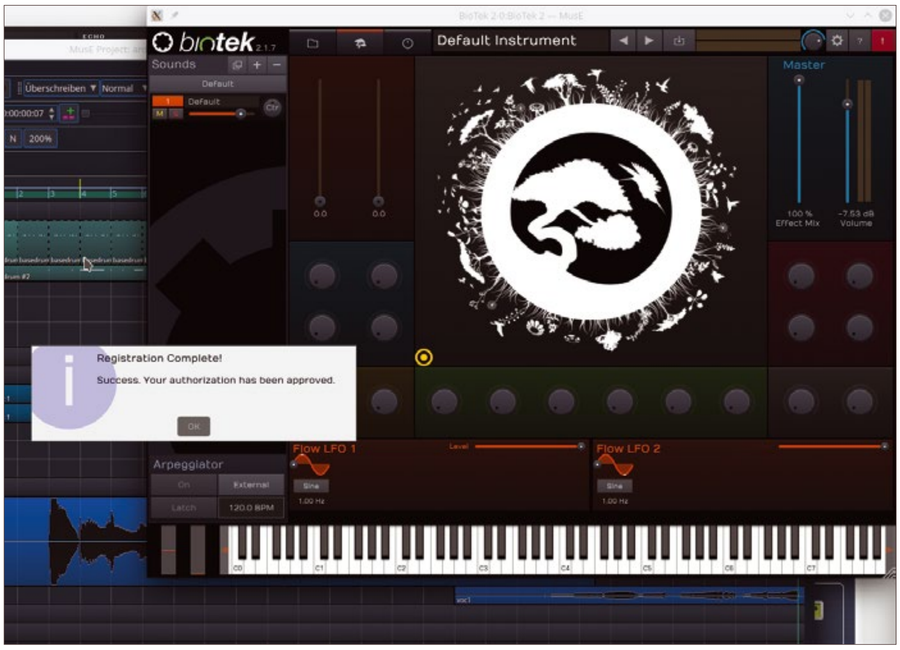


Figure 9: The automation track for the complex Calf Monosynth sits in its own plugin track, instead of the MIDI tracks controlling it. You can control anything that the synth's interface lets you control here.



Registration Complete!
Success. Your authorization has been approved.
OK

Figure 10: After registering with your email address and a password for your Mackie account on the Tracktion website, BioTek works flawlessly in MusE.

Muse Confusion

In recent years, there has been some confusion about software that has "Muse" in its name. While an Internet search for *muse* might turn up the British rock band of the same name, far more confusion is caused by *muse-sequencer.org*, which seems to refer to a non-commercial sequencer. In reality, *muse-sequencer.org* is a sneaky advertising site for proprietary music production software.

In addition, other software vendors have named their products Muse, ranging from a healthcare contact management system to a note-taking app for iOS. While *mu.se* [6] actually has something to do with music software, it's the website for the US-based makers of the proprietary Windows software Ultimate Guitar. Because they use a Swedish top-level domain name, it's easy to assume that they have something to do with Robert Jonsson, who is Swedish.

Finally, MusE should not be confused with the Muse Group (publisher of MuseScore), which caused a stir when it acquired the free wave editor Audacity in April 2021 [7]. The Muse Group also offers a website [8] that serves as a kind of social network for musicians.

MusE comes with dozens of presets for popular classic synths and many for modern plugins. When you select a preset for your instrument, the program embeds the control signals for the instruments into the data tracks. In addition, individual initializations can be implemented as scripts. There are no special tools for this, and it requires a good deal of prior knowledge, but if you know your synths and their interfaces inside out, you can do practically anything with them. This also applies to hardware synths connected via the MIDI sequencer in JACK or ALSA.

For newcomers, it makes more sense to rely on MusE's sensible automations and ignore the complexity. The usual synth plugins such as Yoshimi or Calf Monosynth work flawlessly even if you don't manually tweak the settings. The same applies to commercial Linux VST modules such as Mackie's complex BioTek synth. BioTek's registration lets you use it in MusE even if you bought BioTek in a bundle with Mackie's Waveform (Figure 10).

Conclusions

MusE 4 not only impresses with a new contemporary interface, it also appears more mature and carefully built than its predecessors. The way simple audio tracks work is largely intuitive.

For projects that primarily work with audio material, however, Ardour certainly offers more possibilities. For MIDI compositions, MusE currently outshines other DAWs on Linux. Bitwig might offer more options for loops, but it lacks the note editor offered by MusE.

The numerous setting options for initializing MIDI instruments in MusE are likely to appeal to musicians with in-depth knowledge in this field. Also, the simple way MusE cooperates with yabridge to let users integrate VST plugins in the Windows DLL format will please many users. Hopefully, Robert Jonsson will continue to work as patiently and persistently as he has thus far on the future roadmap of this interesting Linux DAW. ■■■

Author

Hartmut Noack works as a lecturer, author, and musician. He thinks that free software and self-made music go together well. Hartmut's own server on *http://lapoc.de* has some Creative Commons-licensed goodies resulting from his work with free music software.

Info

[1] MusE: *https://muse-sequencer.github.io/*

[2] yabridge: *https://github.com/robbert-vdh/yabridge/releases*

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

[4] MuseScore: *https://musescore.org/en*

[5] Robert Jonsson: *https://xouba.net/post/25297044437/floss-your-music-robert-jonsson*

[6] Muse Group: *https://mu.se*

[7] Audacity acquisition: *https://www.musicradar.com/news/audacity-spyware-claims*

[8] Commercial MuseScore community: *https://musescore.com*

Rethinking Unix-like systems

DragonFly BSD

Matthew Dillon, founder of DragonFly BSD, discusses the past, present, and future of BSDs. *By Bruce Byfield*

It's easy to forget that Linux is only one member of a whole family of Unix-like systems, especially the Berkeley Software Distribution (BSD). Among the many BSDs, one of the most interesting is DragonFly BSD (featured on this month's DVD). DragonFly BSD has a history of rethinking Unix-like subsystems and processes in a never-ending effort to make them more efficient. Matthew Dillon, the founder of DragonFly BSD, learned his craft in AmigaOS and remains active today in the Dragonfly project. Dillon offers an expert's view of not only BSD but of Linux and open source as well.

Linux Magazine: Tell us a bit about how you became involved in BSD.

Matthew Dillon: In the early days (and now I'm talking about the 1980s), open source was far more personal, and individuals could put together and sell wonderful programs via shareware, as well as give away code. I wrote DICE for the Amiga, for example. DICE is a full 68000 compiler – editor, preprocessor, C compiler, assembler, and linker – the whole thing. I sold that back in the 1980s as shareware, but I also included the full and complete sources to the whole thing. Many young programmers thanked me, even decades later, for being able to mess around with that source code. And for the Amiga, I contributed lots of really useful programs like DME (an editor) and DMouse (a mouse accelerator), and a few other things. In the 1990s, I leveraged my experience to start (along with three other founders) one of the first real Internet ISPs in the South Bay area (California), called BEST Internet. We started with SGI platforms but later switched to BSDi, and eventually to FreeBSD. That is how I really got involved with FreeBSD, because we had lots of panics and I was running a business so I had to fix them. And of course I began submitting the fixes at a furious pace.

LM: How did DragonFly BSD get started?

MD: Many projects at the time (late 1980s going into the 1990s) were still very small and experiencing serious growing pains as their user bases exploded. Linux, BSDi, FreeBSD, NetBSD, and OpenBSD were all highly relevant, but personalities made any sort of merging impossible (though numerous committers wound up working on multiple projects, many to this day). In fact, at the time, Linux was considerably more primitive than the BSDs. But it also had a much younger following, a more open nature, and the promise of GNUism, while the BSDs tended to be far more insular. I did work on Linux a bit in the early days, but having gone to school at Berkeley I had basically grown up on BSD, so I was naturally attracted to the BSD codebase.

Ultimately my attraction culminated in becoming a committer in the FreeBSD project. Nobody at the time really knew how the cards would fall.

Ultimately the BSD projects' internal problems caused them to lose ground, and younger generations became enamored with GNU licensing verses the more realistic BSD give-it-away style of license. Linux became dominant.

LM: Why was DragonFly BSD forked?

MD: DragonFly BSD was forked off of FreeBSD for several reasons. Personal interactions coupled with secrecy and backroom dealings had caused the internal mailing lists of the FreeBSD project to become quite toxic (before, during, and after my tenure). There was a great deal of resistance to ripping up and rewriting subsystems as multicore CPUs became more prevalent – lots of people wanting their opinions heard, but few of them were the ones actually doing the hard work. That combination led to a falling out, so the project and I parted ways and DragonFly was born.

FreeBSD ultimately went a more corporate route which kept it in the game, and many of its principals were able to put together small businesses or go to work for large businesses (for example, Netflix) around their project work.

The DragonFly project was more relaxed and open and maintained more of its open source/small-project social roots, remaining so to this day. Other BSDs also exist to this day, but to be completely honest all the BSD projects have been aging out slowly, and even the Linux core is suffering similar pain, albeit at a much larger scale.

LM: What are DragonFly BSD's priorities for development?

MD: Originally our focus was on getting symmetric multiprocessing (SMP) unlocked as scalable as possible. Our motto is that the best type of SMP lock is the one you don't need to have in the first place. We focused on partitioning the CPUs to remove locking entirely. Of course, there is still a lot of locking. There are still many shared resources. But DragonFly also runs a number of subsystems lockless, such as the network protocol stacks and packet filters. Not just lockless, but completely unshared, so the Ethernet hardware's Toeplitz Hash routes a packet directly to the correct CPU and the network stack on that CPU operates on the connection (stream or packet connection) just about as close to lockless as it is possible to get.

Over time, though, Linux, DragonFly, and FreeBSD all came to nearly the same place. Not quite the same – Linux focuses a lot more on the ultimate performance, but its kernel mechanisms are considerably more fragile because of that. DragonFly focuses on a combination of SMP performance, stability, and reliability. FreeBSD is similar. And all the projects have to compile the huge library of third-party applications that end users desire. In that respect, Linux is well ahead of all the BSDs. FreeBSD is ahead of DragonFly on drivers (more eyeballs), but at the raw programmer level all the projects still share a great deal of intellectual conversation, sharing, and benefit. That's just the reality of how it all evolved.

LM: What would newcomers from Linux find different in DragonFly BSD?

MD: A Linux user would find a more complete base system on initial installation but might be taken aback at the lagging hardware support. Hardware changes all the time, is mostly proprietary, and requires a lot of eyes to reverse engineer. Linux has eyes that the BSDs do not.

LM: What are DragonFly's distinguishing features? To whom does it appeal?

MD: There aren't as many distinguishing features these days beyond more ephemeral stability, performance, and filesystems. We do have HAMMER2, which is an excellent single-image filesystem, but we have fallen behind on block-level redundancy mechanisms. On the other hand, most end users abuse such mechanisms and wind up with less stability than they would just using a typical live independent backup and snapshotting model.

The DragonFly subsystems are quite refined, so we do get some users coming into DragonFly from other operating systems for specific reasons. Dominant among these is strong performance on machines with limited amounts of RAM and a really excellent process scheduler. Our swap and paging system is one of the best of all the projects, so DragonFly can run on low-memory machines and still give users an excellent experience (even when they leave Chrome tabs open for weeks, which takes a lot of swap space because of Chrome's memory leakage). In terms of appeal, it comes down to hardware compatibility and how the user wants to interact with the system. Hardware support is probably our weakest point.

LM: How is DragonFly BSD organized and governed?

MD: It is a very small project so we mostly just talk things through on IRC these days, with mailing lists primarily for announcements. The social atmosphere has always been quite laid back. It's very nontoxic, so conversations are easy. Generally speaking, if someone wants to do a particular piece of work, they really are in the driver's seat as to how it should be done. The rest of us will help and make sure that the code is up to our standards, but we do that by actually working on the code cooperatively back and forth, not just throwing ideas out on internal mailing lists. We do have a rack at a co-location and some

significant processing power for doing bulk builds and such.

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

MD: Regular contributors are one or two dozen or so. I don't know about users, but I am constantly surprised at the number of emails I get from people quietly using the system.

LM: Looking back, how would you say BSD in general and DragonFly specifically have grown, both technically and in terms of audience?

MD: As projects go, I think all the BSDs are aging out, including DragonFly, but can remain relevant in this world of Linux as long as we are able to offer installation on modern systems. The actual underlying code, algorithms, and concepts, however, are strongly shared across all the projects. People who need ideas reference all the major codebases to see how a problem has been solved, even if they don't pull the actual code. We intend to continue on as long as interest remains. All DragonFly devs have their own lives … the project does not require 24x7 attention.

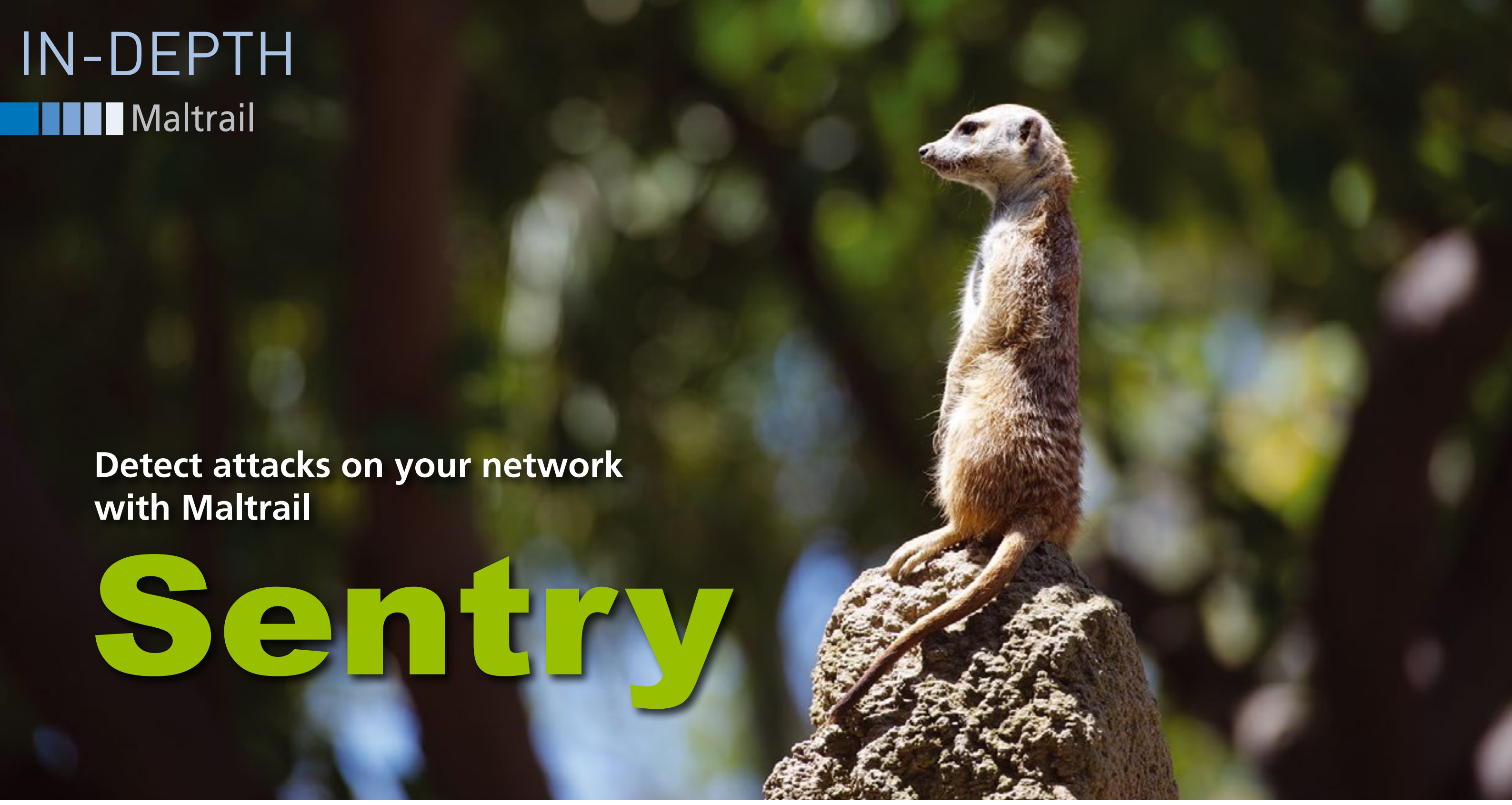
LM: Would you like to say anything else?

MD: I think open source in general needs a bit of a revisit. I think a lot of open source programmers who didn't have the benefit of the early Wild, Wild West days have become disillusioned. In modern times, projects tend to be much, much, much larger, and no single programmer can write or maintain it. The open source model for a project often does not survive its creator or, at least, does not see significant development after the original authors move on to other pastures. Nobody wants to work on or bug hunt someone else's project – there is precious little recognition in such endeavors. The GNU license winds up being a ball and chain as much as it opens code up to the wider world, and the BSD license doesn't help a whole lot, either. There is a massive amount of open source out there, but precious little compensation for the authors. Even successful projects tend to have limited scale because if something becomes too successful its open source nature makes exploitation too easy (just by a project involving many contributors at a large company vs. being contained to a single person). A great deal of the spirit has been lost in modern times.

Open source programmers, for the most part, have to make their livelihoods in other ways – either leveraging the experience to obtain a good job, or by some other means (sometimes not even related to the original work). It is not a failure, as such, because open source has created a technological baseline for the entire world – a very, very high bar for commercial companies that would otherwise sock society with junk. But it is, perhaps, not really what today's open source authors were hoping for. ■■■

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/*.



Detect attacks on your network with Maltrail

Sentry

Maltrail is a lightweight analysis tool that examines network traffic and raises the alarm if it detects suspicious access or dubious name resolution. *By Markus Stubbig*

Hundreds of security products vie for the favor of users on the Internet, promising the highest levels of protection. Along with the numerous commercial offerings available for a monthly rate are some free open-source products that aim to expand the basic protection that might already be in place.

Maltrail [1] is an open source tool that lays in wait on the network and sounds the alarm if a package appears suspicious. It reports its findings but does not intervene. The way Maltrail works is somewhere between an intrusion detection system and a malware scanner. Maltrail uses public blacklists to examine the packages. In Maltrail jargon, the description of a suspicious IP address, web URL, or domain is known as a *trail*. *Feeds* are lists of known trails that the Maltrail community keeps up to date.

Structure

Maltrail consists of two components. The sensor component sniffs the packets, and the server component collects the alarms from the sensor. In a perfect setup, the sensor component resides on a router or firewall, because these devices get to see the data streams of all network participants. In Figure 1, the sensor resides on a firewall and therefore has access to all the packets passing through. The position of the server does not matter much as long as the sensor and the admin can access it.

Installation

The Maltrail program code is written entirely in Python. Maltrail is not picky about the Python version. Basically, all interpreters with a version number of 2.6 or newer will work, and this means that even older Linux servers can be used as sensors. The sensor also needs the Python *pcapy* package to intercept the IP packets from the network adapter. The software itself is available from Github under a free license.

For the install, use your distribution's package manager and install the required packages (Listing 1, first line). Afterwards, retrieve the program code from Github and store it locally (line 2). By default, the sensor listens on all available network adapters and dumps its warnings into a local file.

When first launched (Line 3), the sensor fetches all available blacklists from the net and drops them into the `~/.maltrail/` subfolder. Maltrail then starts its magic. To enable the sensor to report to the server, you need to add its IP address and port to the Maltrail configuration file `~/.maltrail/maltrail.conf` (Listing 2, Lines 2 and 3). Then launch the server (Listing 1, Line 5). The sensor sends its results to the DNS name or to the previously resolved IP address of the server (Listing 2, last line).

It is easy to test whether the communication link between the sensor and the server is open. To test the link, resolve a domain that is blacklisted by Maltrail on the system running the sensor (Listing 1, Line 6). The sensor detects the action and reports the incident to the server.

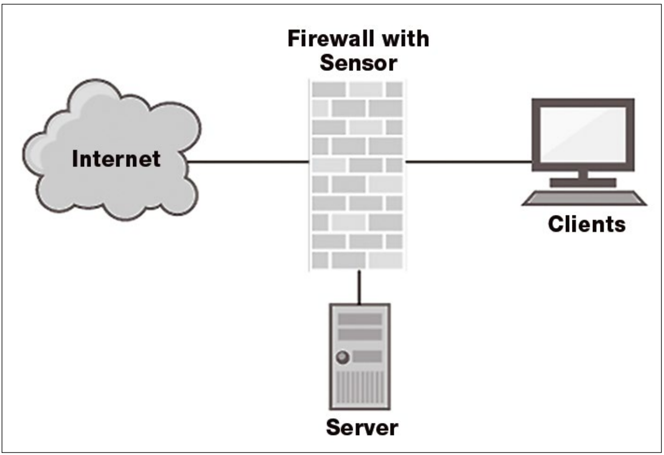


Figure 1: The Maltrail sensor sits in the data stream and reports to its server.

Evaluation

The server delivers the results on a pretty web page. *http://server.example.net:8338* gives you access to graphical processing of the reported alarms, grouped by message type, sensor, and IP address of the originating client. You have to judge for yourself whether the message is actually a real threat. For example, if the Windows computers on the local network use the Remote Management API, Maltrail will report suspicious actions, but this is no reason for you to worry.

Heads up to data protection officers: Maltrail does not comply with rules for protection of personal data. For example, the IP addresses of the clients can be found in the analysis, and user names are also occasionally found in the HTTP access reports. If you don't want to get into trouble, black out the client address in the log file afterwards with the commands from Listing 3.

The server's web interface presents a daily report with all alert messages received (Figure 2). The report summarizes the threats, the events, the most suspicious IP addresses, and the most frequent trails in colorful charts. You can obtain details of the recorded addresses by hovering the mouse over the text.

Many programs and operating systems phone home in the background, and Maltrail shows in its overview which of the messages seem suspicious. In addition, Maltrail lets you know which machines attempt to communicate with each other on the local network without you knowing about it or wanting it to happen.

The project presents a collection of examples of real-life attacks on its website in the *Real-life cases* section. If you find similar entries in the reports of your Maltrail server, your network may have been the victim of a mass scan, a port scan, malware, or a data leak. Of course, not every report will automatically trigger a Class 1 alert. Sensible use of Maltrail requires you to adapt the alerting system to your own environment and minimize the false positives by doing so.

Place of Use

If you are not able to install Maltrail on a router, switch, or firewall, you need to run the software in passive mode. To operate Maltrail in passive mode, set up the sensor on any computer and then configure it to receive a copy of each packet via a mirror port on the core switch. In this configuration, the sensor is not sitting in the data path, but it will still get enough information to work effectively.

Small networks with a Fritzbox or Speedport as the central component require more effort. The popular DSL routers only allow insights into the incoming packets via a backdoor. This is where manufacturer support ends and tinkering begins. Fortunately, various scripting geniuses have shared their success in accessing the hidden areas of the Fritzbox [2] web interface and other devices [3]. If your router does provide the packet stream in PCAP format, the Maltrail sensor can read and analyze the resulting file with the `-i` switch.

Listing 1: Set up Maltrail

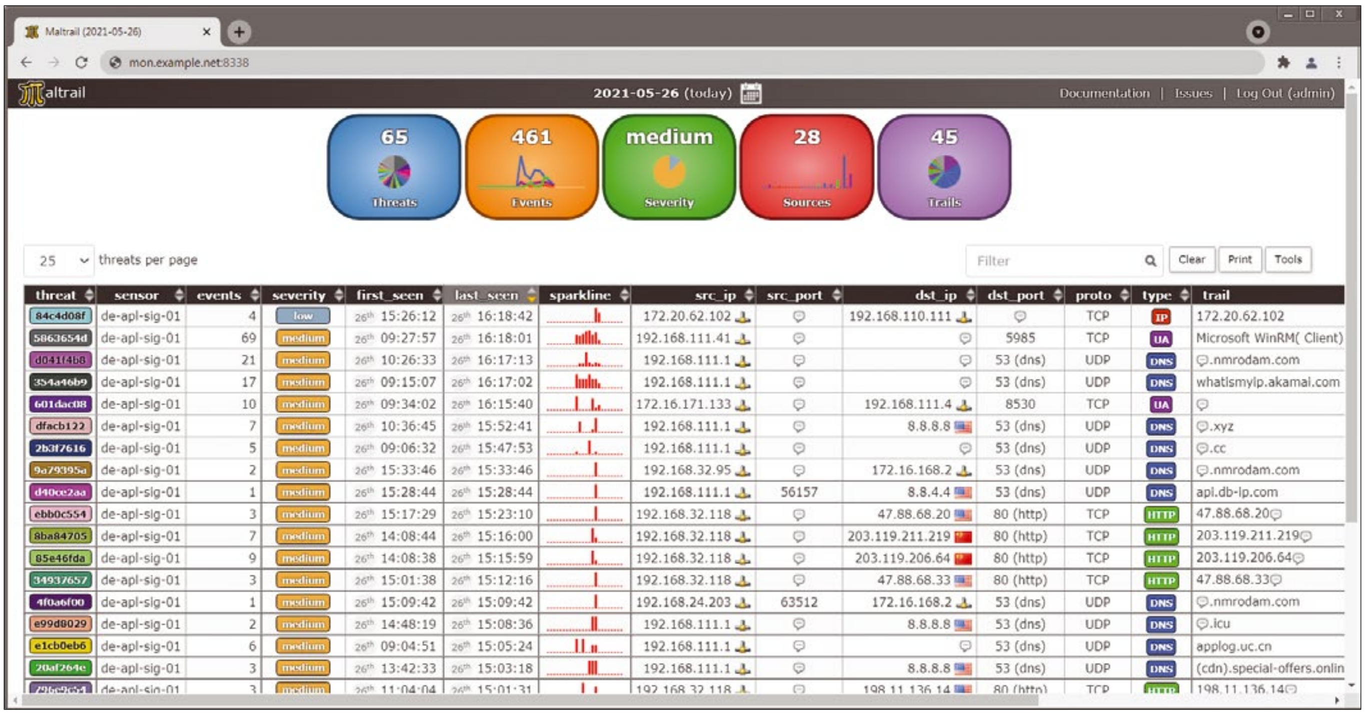
```
$ sudo apt install python-pcapy git
$ git clone
  --depth 1 https://github.com/
  stamparm/maltrail.git
cd maltrail
$ sudo python sensor.py
$ sudo python server.py
$ nslookup kshield.net
```

Listing 2: maltrail.conf

```
# Address and port of the server
UDP_ADDRESS 0.0.0.0
UDP_PORT 8337
# log server
LOG_SERVER server.example.net:8337
```

Listing 3: Anonymization

```
$ cat $(date +"%Y-%m-%d").log |
  awk '{ $4 = "-"; print }' > temp
$ mv temp $(date +"%Y-%m-%d").log
```



threat	sensor	events	severity	first_seen	last_seen	sparkline	src_ip	src_port	dst_ip	dst_port	proto	type	trail
84c4d08f	de-apl-sig-01	4	low	26th 15:26:12	26th 16:18:42		172.20.62.102	💬	192.168.110.111	💬	TCP	IP	172.20.62.102
5863654d	de-apl-sig-01	69	medium	26th 09:27:57	26th 16:18:01		192.168.111.41	💬	💬	5985	TCP	UA	Microsoft WinRM(Client)
d041f4b8	de-apl-sig-01	21	medium	26th 10:26:33	26th 16:17:13		192.168.111.1	💬	💬	53 (dns)	UDP	DNS	💬.nmrodam.com
354a46b9	de-apl-sig-01	17	medium	26th 09:15:07	26th 16:17:02		192.168.111.1	💬	💬	53 (dns)	UDP	DNS	whatismyip.akamai.com
601dac08	de-apl-sig-01	10	medium	26th 09:34:02	26th 16:15:40		172.16.171.133	💬	192.168.111.4	8530	TCP	UA	💬
dfacb122	de-apl-sig-01	7	medium	26th 10:36:45	26th 15:52:41		192.168.111.1	💬	8.8.8.8	53 (dns)	UDP	DNS	💬.xyz
2b3f7616	de-apl-sig-01	5	medium	26th 09:06:32	26th 15:47:53		192.168.111.1	💬	💬	53 (dns)	UDP	DNS	💬.cc
9a79395a	de-apl-sig-01	2	medium	26th 15:33:46	26th 15:33:46		192.168.32.95	💬	172.16.168.2	53 (dns)	UDP	DNS	💬.nmrodam.com
d40ce2aa	de-apl-sig-01	1	medium	26th 15:28:44	26th 15:28:44		192.168.111.1	56157	8.8.4.4	53 (dns)	UDP	DNS	api.db-ip.com
ebb0c554	de-apl-sig-01	3	medium	26th 15:17:29	26th 15:23:10		192.168.32.118	💬	47.88.68.20	80 (http)	TCP	HTTP	47.88.68.20💬
8ba84705	de-apl-sig-01	7	medium	26th 14:08:44	26th 15:16:00		192.168.32.118	💬	203.119.211.219	80 (http)	TCP	HTTP	203.119.211.219💬
85e46fda	de-apl-sig-01	9	medium	26th 14:08:38	26th 15:15:59		192.168.32.118	💬	203.119.206.64	80 (http)	TCP	HTTP	203.119.206.64💬
34937657	de-apl-sig-01	3	medium	26th 15:01:38	26th 15:12:16		192.168.32.118	💬	47.88.68.33	80 (http)	TCP	HTTP	47.88.68.33💬
4f0a6f00	de-apl-sig-01	1	medium	26th 15:09:42	26th 15:09:42		192.168.24.203	63512	172.16.168.2	53 (dns)	UDP	DNS	💬.nmrodam.com
e99d8029	de-apl-sig-01	2	medium	26th 14:48:19	26th 15:08:36		192.168.111.1	💬	8.8.8.8	53 (dns)	UDP	DNS	💬.icu
e1cb0eb6	de-apl-sig-01	6	medium	26th 09:04:51	26th 15:05:24		192.168.111.1	💬	💬	53 (dns)	UDP	DNS	applog.uc.cn
20af264e	de-apl-sig-01	3	medium	26th 13:42:33	26th 15:03:18		192.168.111.1	💬	8.8.8.8	53 (dns)	UDP	DNS	(cdn).special-offers.onlin
796c9c54	de-apl-sig-01	3	medium	26th 11:04:04	26th 15:01:31		192.168.32.118	💬	198.11.136.14	80 (http)	TCP	HTTP	198.11.136.14💬

Figure 2: The Maltrail server informs about all alarms.

If this kind of workaround is too much work for you, the only other option is to use Maltrail as a host IDS. In this case, the sensor runs locally on all the servers you want it to monitor. However, Maltrail will no longer be able to protect the entire network, only selected computers.

Exceptions

Not all trails on your network pose a threat. Maltrail also sees services like DynDNS and various HTTP user agents as potential risks. To prevent harmless messages from filling the entire log, you can exclude groups of trails or even entire feeds from the scan.

If you want individual trails to disappear, make them invisible by right-clicking and selecting *Hide Threat*. Conversely, Maltrail also accepts custom sources with trails that you specify in the configuration file via the `CUSTOM_TRAILS_DIR`.

Special Case: Windows

Officially Maltrail only runs on Linux and Unix, but with a few tweaks, you can also use a Windows computer as a sensor. The problem is not with Maltrail itself, but with the Pcapy package, which comes with a mass of dependencies. If you don't mind the work, download the Pcapy, Winpcap, and Python source code, as well as the Microsoft Visual C++ Compiler for Python, and compile the package yourself [4]. Once this step is completed, you can launch the sensor on Windows, where it will run in the usual way (Figure 3).

Watch and Guess

If the investigated data stream does not match any blocking list, Maltrail has to dig deeper into its bag of tricks and use heuristics. Maltrail looks for hints of malicious behavior, such as unusual user agents in the web requests, port scans, long domain names, and web access with code injection.

The results of these checks are not always precise and can lead to false positives. Whether or not a message made it into the web page due to the heuristics is shown in the *reference* column. If you see an accumulation of the *heuristic* label, you can use the `USE_HEURISTICS` configuration statement to disable this method.

Initiating Countermeasures

Maltrail can report, but it can't fight back. To block IP packets, Linux has iptables/nftables, FreeBSD has Pf, and Windows relies on the Microsoft Defender firewall. At least the Maltrail server provides its findings as a list of suspicious IP addresses via HTTP. The project page describes how to feed this information to the local iptables policy with a script. Basically, the IP list can be used to extend any firewall accessible via API or scripting, for example OPNsense and even the Windows firewall.

Limitations

Maltrail is not a full-fledged IDS but merely a packet scanner that makes use of public blacklists. Maltrail cannot detect complex, application-level attacks, which means it won't come close to the detection rate of a real IDS.

Another shortcoming is the communication between the sensor and server, which relies on the unencrypted UDP protocol. A man-in-the-middle attack could sniff, manipulate, or delete alerts to hide malicious activity. For undisturbed transport over the Internet, the admin needs to harden the alert packets with IPsec or SSH. Access to the Maltrail server's web interface uses TLS and a server certificate, which provides sufficient security.

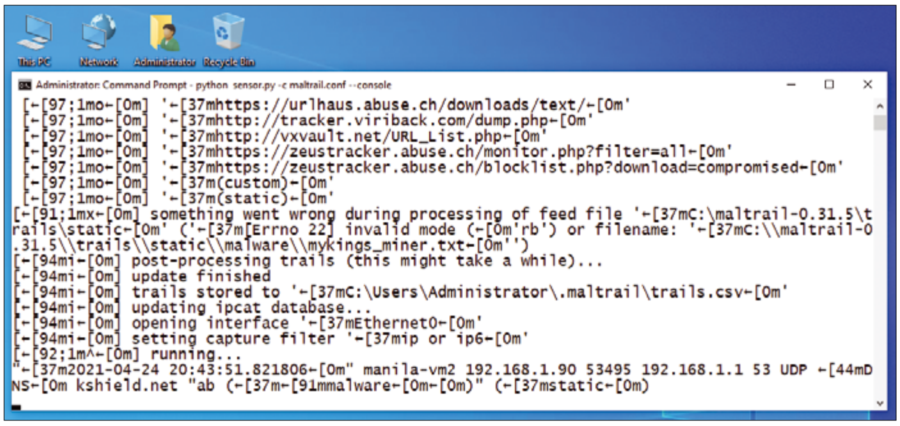
Outlook

Maltrail is also at home in virtual environments. For instance, you can analyze the network traffic of guest systems in a VMware infrastructure. The ESXi host has a Python interpreter on board, but Maltrail works better as a standalone virtual machine. The VM needs one network adapter per port group. Because Maltrail accesses the IP packets via PCAP, the port group or virtual switch needs permission to use promiscuous mode. In this setup, Maltrail works only with copies of the packets. This keeps the virtual servers accessible even if the sensor process throws errors or the Maltrail VM is not running.

The Maltrail sensor can, of course, report to its own server, but it can also feed Syslog and Logstash. To do this, it formats its messages as standardized syslogs or as structured JSON. This support for logging means you can integrate Maltrail into a larger log infrastructure or include it as part of a logging-as-a-service strategy.

Conclusions

The lightweight Maltrail network scanner analyzes network traffic for suspicious activity, gleaning information from freely available blacklists and reputation databases. Maltrail also acts like an intrusion detection system: It loads signatures and compares them with the inspected IP packets. If a match occurs, an alert appears on the dashboard to warn the admin. Maltrail is not an all-around no-worries package, but it is a useful building block in a security strategy. ■■■


This PC
Network
Administrator
Recycle Bin
Administrator: Command Prompt - python sensor.py -c maltrail.conf --console

```
 [←[97;1mo←[0m] '←[37mhttps://urlhaus.abuse.ch/downloads/text/←[0m'
 [←[97;1mo←[0m] '←[37mhttp://tracker.viriback.com/dump.php←[0m'
 [←[97;1mo←[0m] '←[37mhttp://vxvault.net/URL_List.php←[0m'
 [←[97;1mo←[0m] '←[37mhttps://zeustracker.abuse.ch/monitor.php?filter=all←[0m'
 [←[97;1mo←[0m] '←[37mhttps://zeustracker.abuse.ch/blocklist.php?download=compromised←[0m'
 [←[97;1mo←[0m] '←[37m(custom)←[0m'
 [←[97;1mo←[0m] '←[37m(static)←[0m'
[←[91;1mx←[0m] something went wrong during processing of feed file '←[37mC:\maltrail-0.31.5\t
rails\static←[0m' ('←[37m[Errno 22] invalid mode (←[0m'rb') or filename: '←[37mC:\\maltrail-0
.31.5\\trails\\static\\malware\\mykings_miner.txt←[0m'')
[←[94mi←[0m] post-processing trails (this might take a while)...
[←[94mi←[0m] update finished
[←[94mi←[0m] trails stored to '←[37mC:\Users\Administrator\.maltrail\trails.csv←[0m'
[←[94mi←[0m] updating ipcat database...
[←[94mi←[0m] opening interface '←[37mEthernet0←[0m'
[←[94mi←[0m] setting capture filter '←[37mip or ip6←[0m'
[←[92;1m^←[0m] running...
"←[37m2021-04-24 20:43:51.821806←[0m" manila-vm2 192.168.1.90 53495 192.168.1.1 53 UDP ←[44mD
NS←[0m kshield.net "ab (←[37m←[91mmalware←[0m←[0m)" (←[37mstatic←[0m)
```

Figure 3: In a roundabout way, Maltrail also runs under Windows.

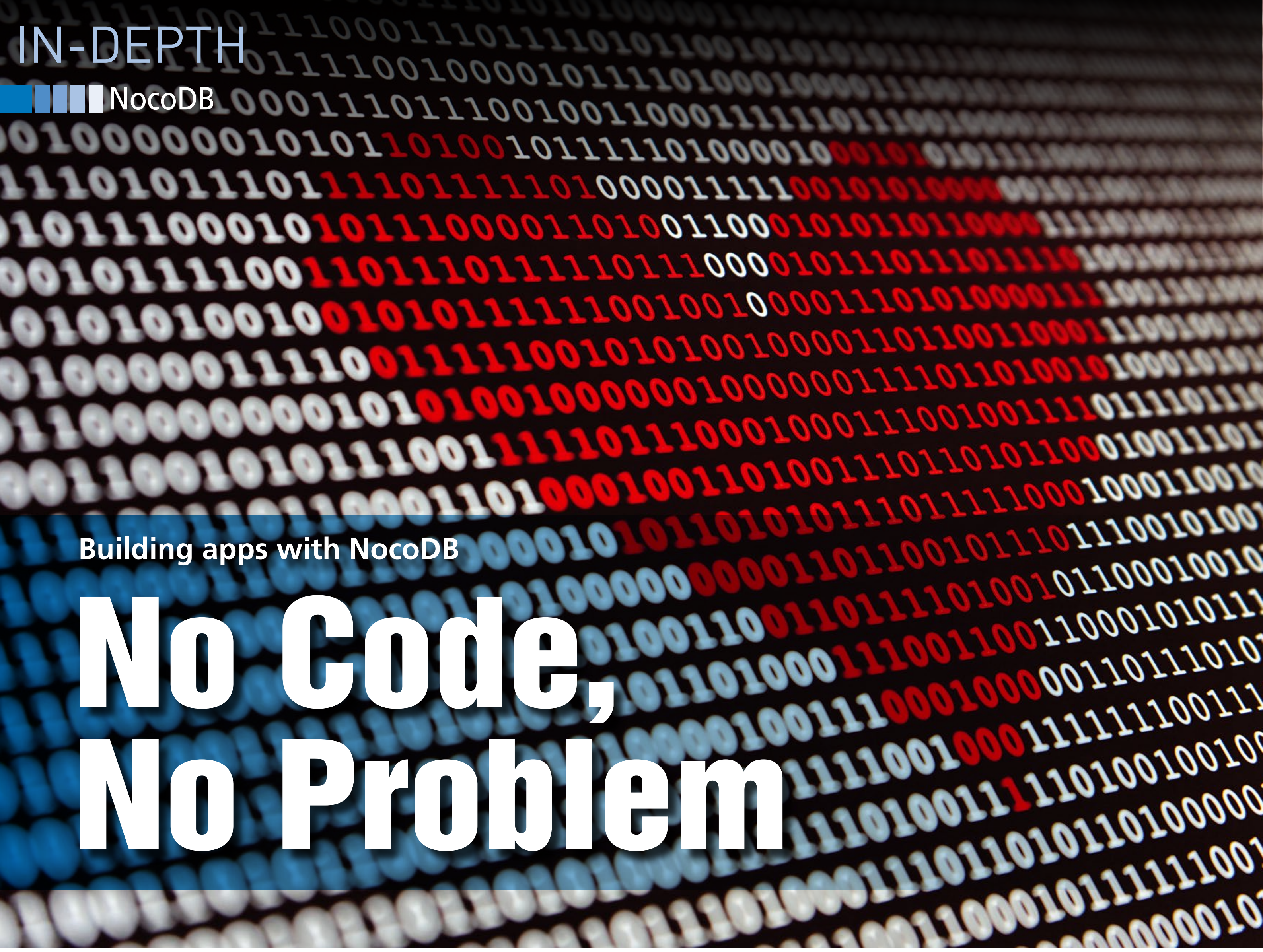
Info

[1] Maltrail: *https://github.com/stamparm/maltrail*

[2] Fritzdump: *https://github.com/ntop/ntopng/blob/dev/tools/fritzdump.sh*

[3] Script access to speedport: *https://github.com/koutheir/speedport-w724v-external-ip-address*

[4] Compiling Pcapy on Windows: *https://github.com/helpsystems/pcapy/wiki/Compiling-Pcapy-on-Windows-Guide*



Building apps with NocoDB

No Code, No Problem

NocoDB lets you build useful applications without writing a single line of code. *By Dmitri Popov*

No-code platforms are all the rage nowadays, and it's not that hard to see why. They make it possible for anyone to create applications for their specific needs in no time and save serious money in the process. Even if you have coding chops, you may find a no-code solution useful for whipping up a quick prototype or a simple application instead of building it from the ground up.

While there are several open source no-code platforms to choose from, NocoDB [1] strikes a perfect balance between functionality and user-friendliness. It's also supremely easy to deploy, which makes it a perfect platform for building simple and more advanced applications.

Deploying NocoDB

The easiest way to deploy NocoDB on a local machine is by using the NocoDB Docker container image. The first step is to install Docker. To do this on Ubuntu and Linux Mint, run:

```
sudo apt install docker
```

To be able to run Docker as a regular user, add the current user to the Docker group using the command:

```
sudo usermod -aG docker $USER
```

Reboot the machine, and you're done.

Next, create a dedicated directory for storing NocoDB data (e.g., `~/nocodb`). This ensures that even when you delete a NocoDB container, your data remains intact.

Finally, run the following command in the terminal (replacing `/path/to/nocodb` with the actual path to the data directory) to start the NocoDB container:

```
docker run -d -p 8080:8080 ⮐
  --name nocodb ⮐
  -v /path/to/nocodb:/usr/app/data/ ⮐
  nocodb/nocodb:latest
```

Upgrading NocoDB

Because NocoDB is deployed as a container, upgrading the platform is a matter of deleting both the container and the image:

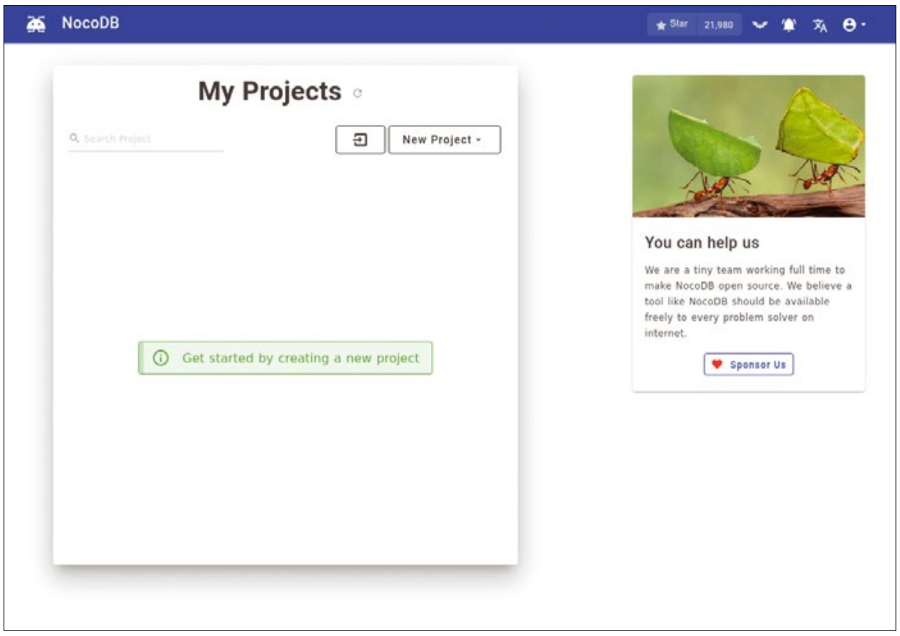
```
docker stop nocodb
docker rm nocodb
docker rmi nocodb/nocodb:latest
```

You then rerun the command you initially used to start the NocoDB container above to automatically fetch the latest NocoDB image.

Creating a NocoDB Application

Point your browser to *127.0.0.1:8080* (replace *127.0.0.1* with the actual IP address of the machine running NocoDB), and you should see the

Photo by Alexander Sinn on Unsplash


NocoDB

Star 21,980

My Projects

Search Project

New Project

Get started by creating a new project

You can help us

We are a tiny team working full time to make NocoDB open source. We believe a tool like NocoDB should be available freely to every problem solver on internet.

Sponsor Us

Figure 1: **Creating a new project in NocoDB.**

welcome page. When prompted, create a user account.

NocoDB makes it possible to create a wide range of applications, from a simple task manager to a more complex note-taking tool consisting of several connected tables. Building a note-taking tool as your first project may sound like a daunting proposition, but it demonstrates how easy it actually is to turn ideas into working applications using NocoDB. This project also provides a perfect opportunity to master most of NocoDB's features. In addition to storing notes, this note-taking application project will allow you to attach files to each note, assign tags, and group notes into categories.

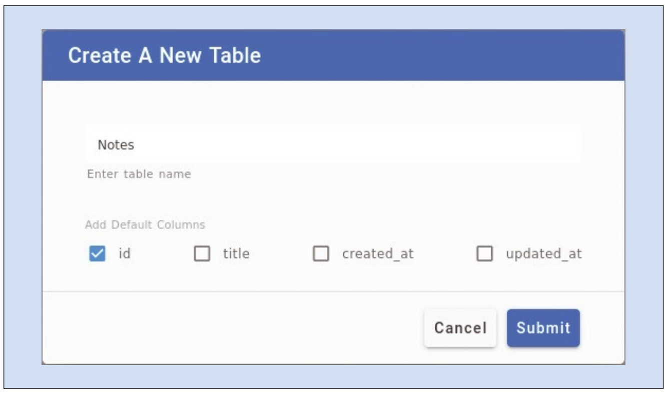
The first step is to create a new application and populate it with tables. After you've created an account, NocoDB displays the *My Projects* dialog (Figure 1). Click *New Project* and select *Create*. Give the project a descriptive name and press *Create*. This drops you into the *Team & Auth* section, with your owner account automatically added to the project. If you want to invite other users to the project, press *New User*, add their email addresses, choose the desired role, and press *Invite*.

Now it's time to create tables. Our application needs three tables: *Notes* for storing notes, *Categories* for storing notebook categories, and *Tags* for storing tags.

To create a table, click the *Add new table* icon (+) next to the *Tables* entry in the left sidebar. Name the table *Notes* and make sure that only the *id* field is enabled under the *Add Default Columns* section (Figure 2). Press *Submit* to create the table. This drops you into the *Notes* table, where your first task is to add the required fields (or columns in NocoDB's terminology). Click the + icon in the table's header row, name the column *Title*, choose *SingleLineText* from the *Column type* drop-down list, and press *Save*. In a similar manner, add another field named *Note*, but instead of *SingleLineText*, choose *LongText* as the column type. Create yet another column named *Attachments* with the *Attachment* column type.

The table needs two more columns, but before you can add them, you need to set up two more tables. Create as described a table named *Categories*. Add a field called *Category* with the *SingleLineText* column type. Now, click on the triangle next to the *Category* field and choose *Set as Primary value*. This sets the *Category* column as a unique identifier of each record (or row in NocoDB's terminology). In practical terms, this makes it possible to link from a column in the *Notes* table to the *Category* column in the *Categories* table. If this sounds a bit confusing, don't worry, it will become clear in a moment. Switch to the *Notes* table, add a new column called *Categories*, and set its type to *LinkToAnotherRecord*. Make sure the *Many to Many* option is enabled, choose *Categories* from the *Child Table* drop-down list, and press *Save* (Figure 3). This creates the *Notebook* column linked to the *Category* column in the *Categories* table.

Use the same procedure to set up a table called *Tags*. Add the *Tag* column to the table and set the column as the


Create A New Table

Notes

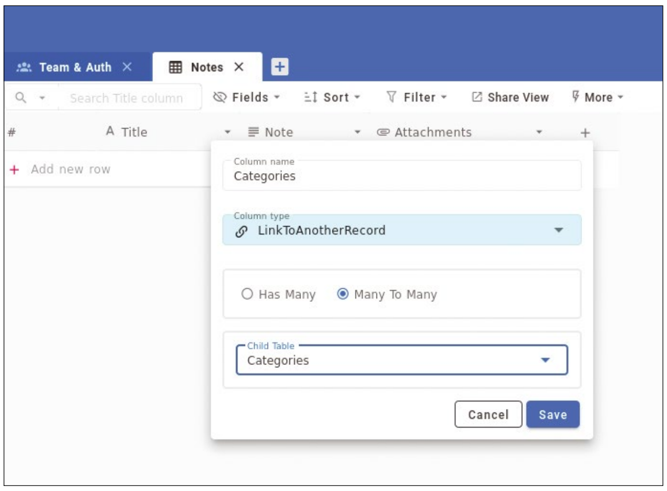
Enter table name

Add Default Columns

[✓ id] title created_at updated_at

Cancel Submit

Figure 2: **NocoDB makes it easy to populate your application with tables.**


Team & Auth | Notes

Search Title column

Fields | Sort | Filter | Share View | More

Title | Note | Attachments

Add new row

Column name: Categories

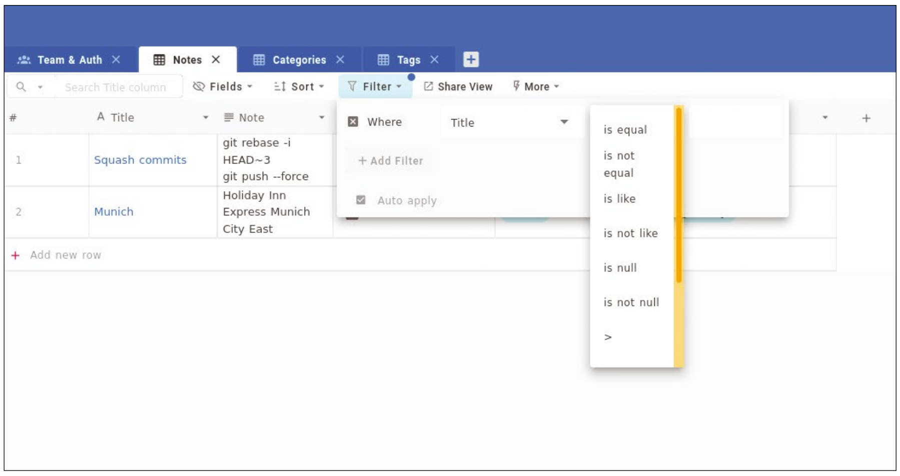
Column type: LinkToAnotherRecord

Has Many [✓ Many To Many]

Child Table: Categories

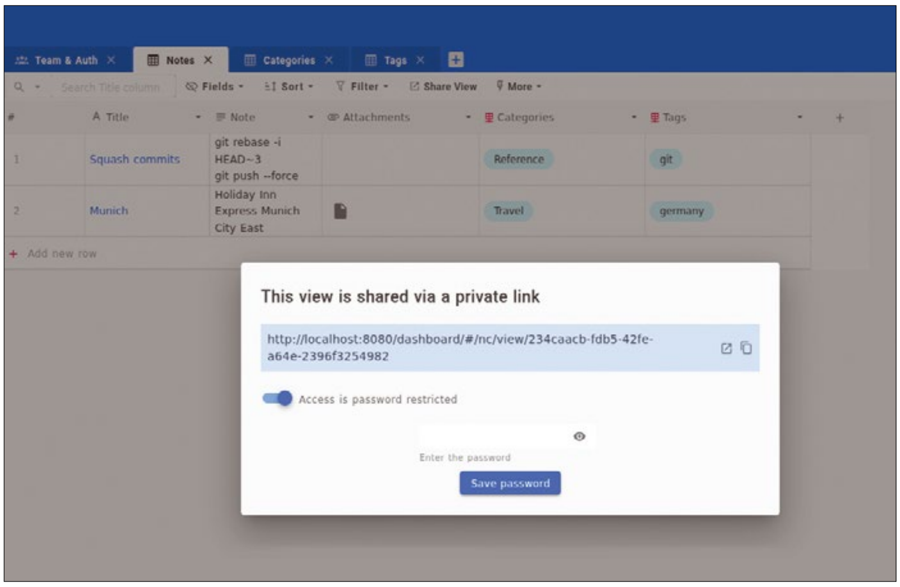
Cancel Save

Figure 3: **Specifying a link between columns in different tables.**



#	Title	Note
1	Squash commits	git rebase -i HEAD~3 git push --force
2	Munich	Holiday Inn Express Munich City East

Figure 4: NocoDB lets you create and apply filters.



#	Title	Note	Attachments	Categories	Tags
1	Squash commits	git rebase -i HEAD~3 git push --force		Reference	git
2	Munich	Holiday Inn Express Munich City East		Travel	germany

This view is shared via a private link

http://localhost:8080/dashboard/#/nc/view/234caacb-fdb5-42fe-a64e-2396f3254982

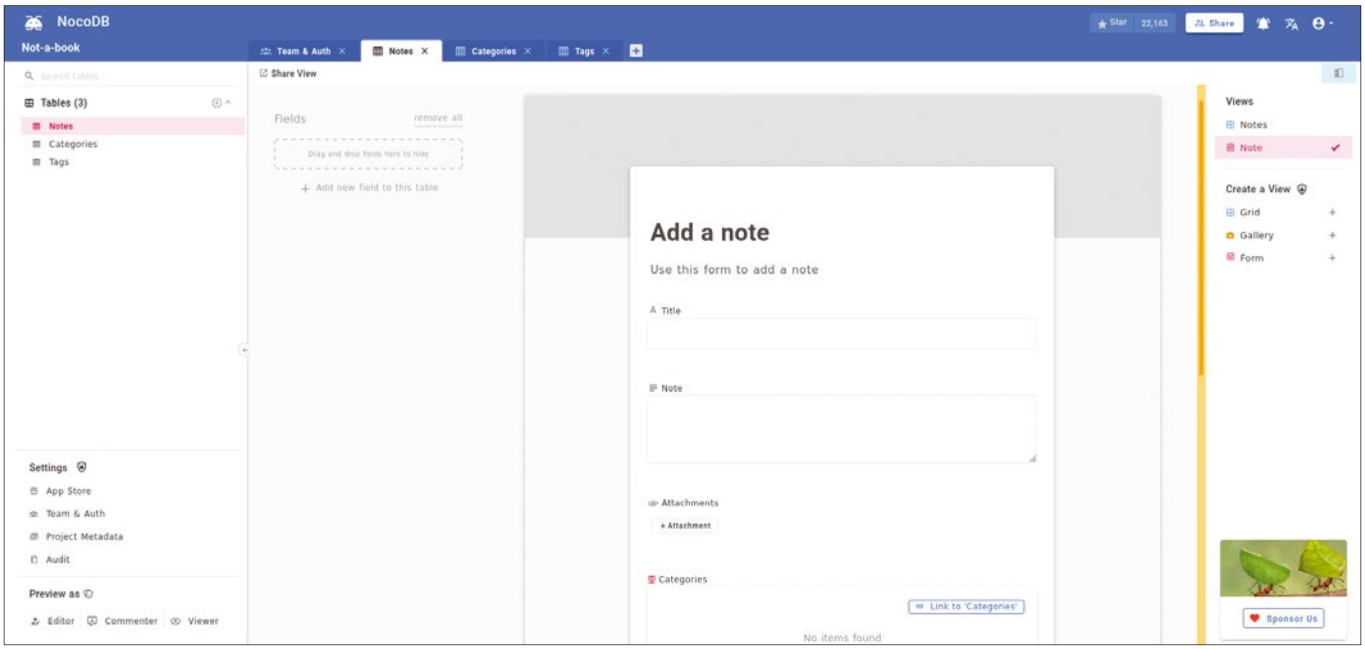
Access is password restricted

Figure 5: You can share specific views via private links that can be password protected, if needed.

primary value. Switch back to the *Notes* table, add a new column called *Tags*, and set its type to *LinkToAnotherRecord*. Make sure the *Many to Many* option is enabled, choose *Tags* from the *Child Table* drop-down list, and press *Save*.

The basic skeleton of the note-taking application is now ready. To test it, switch to the *Categories* table, and use the *Add new row* button in the table view to add two or three rows with the desired categories (e.g., Research, Ideas, Scraps, etc.). Add a few rows to the *Tags* table too. Switch to the *Notes* table and add a new row. Enter text in the *Title* and *Note* fields. To add an attachment, click on the + icon in the *Attachments* field and select one or more files. Because the *Categories* field is linked to the *Category* column in the *Categories* table, you cannot enter text directly into the *Categories* field. Instead, click the + icon and you should see a list of the existing values in the *Categories* table. Click on the desired value to add it to the row in the *Notes* table. With time, the number of rows in the *Categories* table can become rather long, but you can quickly locate the desired value by typing its name into the *Filter query* field and pressing *Enter*.

If the desired value in the *Categories* table doesn't exist, you can add it directly from the *Link Record* dialog. Press + *New Record*, specify the desired category name, and press *Save row*.



Add a note

Use this form to add a note

Figure 6: NocoDB can generate forms that make data entry easier.

At this point, the application looks and behaves pretty much like a spreadsheet. The *Search* field in the top toolbar can be used to quickly find records matching the search term. Using the *Sort* drop-down list, you can sort rows by the desired field, while the *Filter* feature allows you to define and apply filtering rules (Figure 4). While you can invite other users to collaborate on the application, you can also share the current view with anyone. This can be useful if you'd like to allow someone to access a specific part of the application without requiring the recipient to register or provide any information. Pressing the *Share View* button generates a private link that you can share with anyone; you also can protect the view with a password if so desired (Figure 5). Keep in mind, though, that because NocoDB is running on your local machine, the private link only works on your local network. You also need to replace the *localhost* part of the link shown in Figure 5 with the actual IP address of the machine running NocoDB.

Finally, using the options available under *More*, you can export data in a comma-separated file, as well as import existing data into the current table.

The default Grid view is fine for viewing, searching, and sorting rows, but it's less suitable for entering data. This is where the Form view comes into the picture. Select *Form* under the *Create a View* section in the right sidebar, give the view a descriptive name, press *Submit*, and NocoDB generates a ready-to-use form that requires only minor tweaking (Figure 6). Add a title and a short description, and you're pretty much done. Normally, you'd want to include all fields in the form, but if that's not the case, you can hide the individual fields by dragging them off of the form. You can also rearrange the fields in the form by dragging them up or down. It's also possible to configure the form's behavior after the data has been submitted. As with other views, you can share the Form view with anyone, thus giving external contributors the ability to submit data.

Wrap-Up

NocoDB is not the most advanced open source no-code platform out there, but what it lacks in functionality it makes up for in ease of use. Even if you have no experience working with databases, you'll be able to create versatile applications in NocoDB in no time. NocoDB's simplicity doesn't mean it's limited to building basic applications, though. The ability to link records, support for a wide range of field types (including formula and lookup fields), and integration with third-party services, make NocoDB a solid tool for developing advanced solutions. ■■■

Info

[1] NocoDB: *https://www.nocodb.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 Spanish magazines and websites. You can find more on his website at *tokyoma.de*.

Creating ready-to-print photo books with Ruby and TeX

Photo Book Engineering

Instead of composing photo books online, Mike Schilli prefers to use Ruby and TeX to program an application that generates print-ready PDF books. *By Mike Schilli*

I thought I had actually already given up printed books. Today, when I buy a paper book, typically used, I immediately send it to the paper cutting guillotine. The individual pages are then fed to my scanner and ultimately end up digitalized on my iPad, where I then read them.

But there is one type of paper book that I still cherish and even find preferable to digital formats: thick hardcover photography tomes that you might find scattered haphazardly on a little table in a hotel lobby or a vacation home. You just plop down in an armchair next to them, lean back, put your feet up, and start leafing through the high quality pages.

These books are known as coffee table books, probably because they are usually found on coffee tables. You can even create such works yourself with your own photos! Various online providers offer browser tools for uploading photos and laying them out across the pages of a virtual paper book. If you then press the send button and pay the bill, after roughly a week, your mailperson will deliver a high-quality book with a hard cover directly to your home (Figure 1).

Do it Yourself, Instead

The design options, however, are limited online. And, without versioning, you always worry that a poorly programmed application will drop the work you've invested so far into a black hole at any moment, forcing you to start all over again. How about a do-it-yourself program for home use that creates photo books as PDFs, which you can easily and

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.

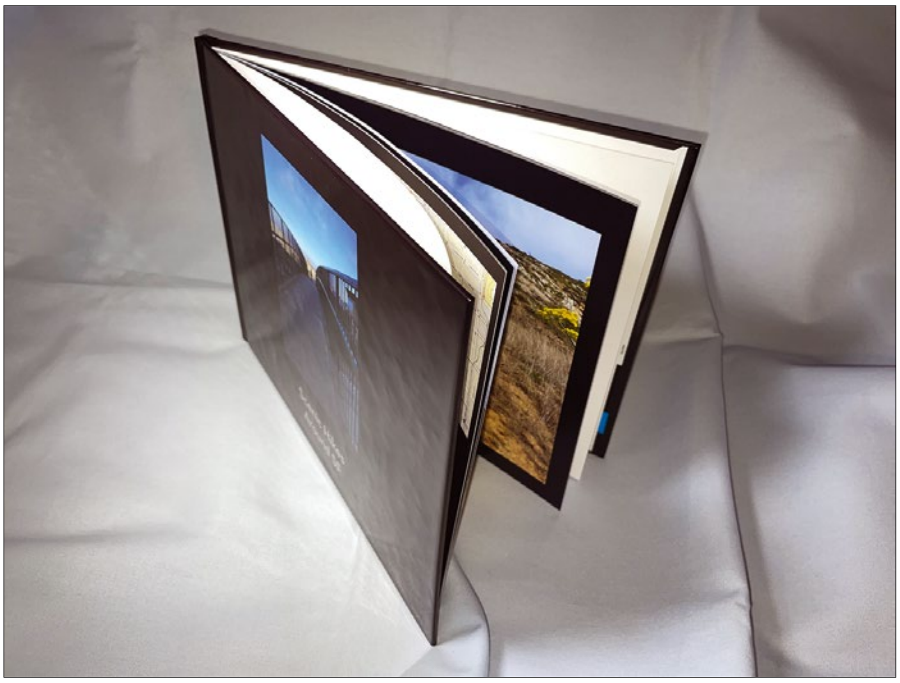


Figure 1: A photo book created by the author and printed online.

Lead Image © Eric Reis, 123RF.com

cheaply have printed and bound in one fell swoop by print services like `Lulu.com`?

In addition, why should a human being have to mess around with a mouse while arranging the pictures to get them all in line, when a typesetting program like TeX will automatically put the photos where they belong with a simple command? Of course, you can't expect users to battle with the quirky TeX syntax and type photo books in the text editor, using dangerous-looking commands that start with backslashes and require you to put any text body in curly braces.

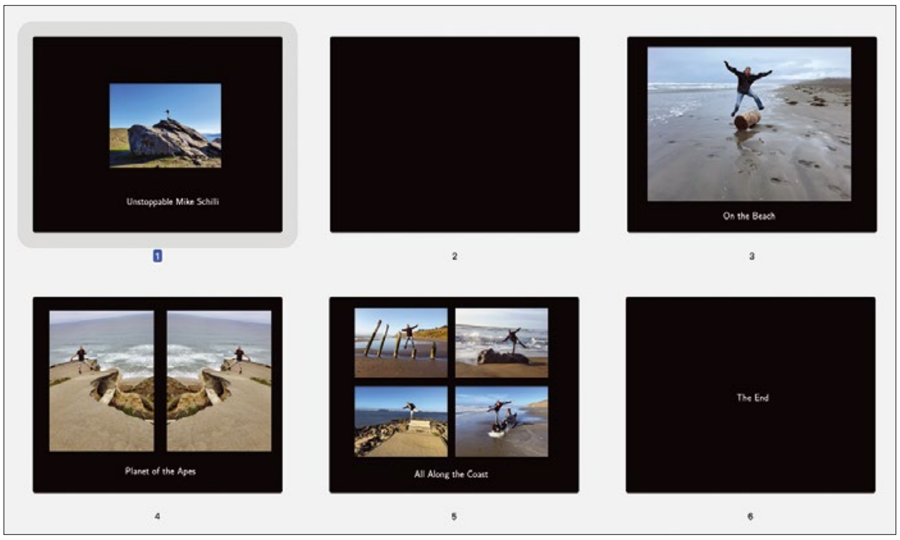


Figure 2: A PDF viewer displays the finished photo book, which is ready to be printed.

Ruby Glue

With a little glue code in Ruby, even non-programmers can create print templates for their coffee table books. Listing 1 shows the definition of a photo book, and the contact print in Figure 2 shows you the PDF generated from it. You need to create one TeX template per page type (such as `cover` for the cover and `four` for a page with four pictures) and then add the associated photos as `photos` and the captioning text as `text`. Boom, the `print()` method in line 35 outputs a PDF with the name defined in line 5 (`"mybook"`), which a PDF viewer such as Evince will render on screen after you type:

```
evince mybook.pdf
```

Any online printing service will turn this into an impressive hardcover book in next to no time. Not bad!

Under the hood, the Ruby interface generates TeX snippets based on the previously defined templates and adds explanatory text and paths to the files of the photos to be embedded in the TeX code using variables. The `print()` method in line 35 finally bundles everything together and calls the XeLaTeX program, which generates a professional-looking PDF from this. You can install XeLaTeX on Ubuntu, for example, by typing

```
sudo apt install texlive-xetex
```

XeLaTeX accepts TeX files, typesets the text, and keeps empty rectangles for the photos, matching their formats and typesetting specifications. The PDF driver then drops the JPEG photos into the rectangles, and you're done.

The advantage? Now the book description is available as code in a text file (new tech buzzword: CoC or Coffeebook as Code), which you can retrieve at any time, modify, or reset to a historical state using standard version control tools like `git`.

Old-Timers: TeX and LaTeX

Of course, what TeX does under the hood is so complicated that it is often difficult to predict whether the photo layout in the template is actually going to produce what the author ordered. Besides, no one (except perhaps TeX author Knuth himself) writes TeX directly to typeset books anymore – that's usually done by a macro package like LaTeX, which is based on TeX. But finding the right magic formula in LaTeX to have it render a nice-looking page without making uninformed decisions often requires nerve-wracking tinkering with the code to finally get the TeX compiler to the point of producing the intended layout.

But once a template is set up and the kinks have been worked out, it will work until the end of time. The original book by TeX inventor Knuth [1] and the work of his LaTeX followers [2] are still worth reading today, if you can find them in an antiquarian bookstore. Experts still publish mathematical papers in LaTeX, and I once wrote a Perl book in LaTeX more than 20 years ago [3]. As I can tell you

Listing 1: mkbook.rb

```
01 #!/usr/bin/ruby
02 require_relative "bm"
03
04 b = Bm.new
05 b.name = "mybook"
06
07 b.add "intro"
08
09 b.add "cover",
10   text: "Unstoppable Mike Schilli",
11   photos: ["ring.jpg"]
12
13 b.add "chapter"
14
15 b.add "single",
16   text: "On the Beach",
17   photos: ["ob-jump.jpg"]
18
19 b.add "twotowers",
20   text: "Planet of the Apes",
21   photos: ["icpf.jpg", "icp.jpg"]
22
23 b.add "four",
24   text: "All Along the Coast",
25   photos: ["thornton-jump.jpg",
26            "beach-rock.jpg",
27            "heron.jpg",
28            "beach-trunk.jpg"]
29
30 b.add "chapter",
31   text: "The End"
32
33 b.add "outro"
34
35 b.print
```

from experience, this particular choice definitely sets an author off on a path with an extremely steep learning curve, but if you value version control and repeatability without manual steps, you have very little choice – even today. Professional typesetting programs like Adobe InDesign or QuarkXPress still can't do anything automatically, even decades later.

Ruby Beats Go

Why use Ruby for the glue code and not Go, which has become the de facto choice in this Programming Snapshot column, you ask? Well, a scripting language like Ruby simply can't be beaten for plain vanilla text manipulation and simple, concise, easy-to-extend code that almost looks like a config file. On top of that, Ruby's template system ERB is great for dumping text or picture file names into predefined snippets with a mess of TeX code. There is a good reason why configuration management tools like Puppet and Chef use Ruby as a reference language for dynamically adapting variegated configuration files catering to individual environments.

Listing 2 shows you the Ruby glue code to turn the book description scripts into TeX. It defines a class that user scripts need to include to generate coffee table books, just like Listing 1 does in line 2. Scripts call the built-in Ruby function `require_relative bm` to find the library from Listing 2 in the same directory and can then immediately get started by calling `Bm.new`. The returned `bm` instance then lets you create sophisticated photo books with just a few commands.

Short Paths

To keep things tidy, the predefined TeX templates are all located in a subdirectory named `tmpl/`. To help the user call them by simple names, such as `cover`, Listing 2 defines an `@tmpldir` instance variable that points to the subdirectory containing the templates. When it's time to load a template, line 14 prepends the `tmpl` path to the provided file name and adds a `.tex` suffix thereafter. This means that the generated path points to a real file, which the ERB template processor can then read from disk in line 19.

The template processor uses ERB's `result_with_hash()` method in line 20 to insert the text supplied for the page caption, and the paths to the photo files, into the respective template. The placeholders in the templates are `<%= text %>` for the page text and `<%= photos[0] %>` for the first element of the `photos` array passed in, containing a list of JPEG paths to the photos to be included. For the image paths, there is no need to laboriously type in the entire path each time. As long as the `@photodir` instance variable is pointing to the directory where all the images are stored (`photos` in our example), you only need to specify the file name without a path in each case in Listing 1.

Defining the Name

To define the name of the book file to be created, the calling script sets the `name` attribute of a `Bm` object as shown in line 5 of Listing 1. To make this work, Listing 2 uses `attr_accessor` in line 5 to define `:name` and with it a read-write attribute `name` for all objects of the class.

This means that an instance of the `Bm` class in `bm` can simply call `bm.name = "mybook"`, like it does in line 5 of Listing 1, to set the book name by assignment. Line 25 in Listing 2 uses `@name` to obtain the name from the instance variable and interpolates it within a string with `"#{@name}"`. This gives you the name of the TeX file with all the piled up TeX code and will send the file to the XeLaTeX LaTeX compiler in the next step.

This means that line 26 only needs to open the `.tex` file for writing and drop the content of the `@tex` instance variable there, after having assembled all the necessary TeX instructions in it. The `system()` call in line 28 then runs XeLaTeX against this, which produces a PDF document of the same name with the finished photo book, if all goes well.

Beginning and End

So what do the template files actually look like? TeX, or rather LaTeX, code is teeming with backslashes and uses curly braces to delimit text areas. After a preamble with the document type and some additional required TeX macro packages, the actual document always begins with a `\begin{document}` statement and ends with `\end{document}`. The `intro.tex` and `outro.tex` templates in Listing 3 and 4 do exactly this, defining the beginning and end of each photo album you intend to generate.

The LaTeX *color* and *pagecolor* packages define the font and background colors in the document. Because photo books somehow look more professional in dark mode (i.e., with a white font on a black background) – or not, the choice is yours – the template sets `\color{white}` and `\pagecolor{black}`.

LaTeX documents usually print page numbers at the bottom of every page, but that's not a good look for a photo book,

Listing 2: bm.rb

```
01 #!/usr/bin/ruby
02 require 'erb'
03
04 class Bm
05   attr_accessor :name
06
07   def initialize
08     @tmpldir = "tmpl"
09     @photodir = "photos"
10     @tex = ""
11   end
12
13   def add(tmpl, text: "", photos: [])
14     tpath = "#{@tmpldir}/#{tmpl}.tex"
15
16     photos = photos.map {
17       |path| "#{@photodir}/#{path}"}
18
19     t = ERB.new(File.read(tpath))
20     @tex += t.result_with_hash(
21       text: text, photos: photos)
22   end
23
24   def print
25     texf = "#{@name}.tex"
26     File.open(texf, "w") {
27       |f| f.write(@tex) }
28     system("xelatex", texf)
29   end
30 end
```

Listing 3: intro.tex

```
01 \documentclass[10pt]{book}
02 \usepackage[export]{adjustbox}
03 \usepackage{color}
04 \pagecolor{black}
05 \color{white}
06 \usepackage[paperwidth=9in,
07   paperheight=7in,
08   total={8in,6.5in}]{geometry}
09 \pagestyle{empty}
10 \begin{document}
11 \sffamily
```

Listing 4: outro.tex

```
\end{document}
```

Listing 5: chapter.tex

```
01 \null\newpage
02 \vspace*{\fill}
03 \center{\Huge{<%= text %>}}
04 \vspace*{\fill}
```

so the command `\pagestyle{empty}` turns off this feature. In addition, TeX's standard font, Computer Modern, looks somewhat antiquated nowadays, but the sans serif version (selected by the command `\sffamily`) is still pretty modern, at least to my untrained eyes. The *adjustbox* package lays the foundation for the *graphics* package to include photos in the text and supports features like scaling to make sure the photos fit into the boxes left blank by TeX in the layout stage.

The photo book's page format is specified by the *geometry* package starting in line 6 of Listing 3. In line with the specifications of the print service provider I use, the pages measure 9 by 7 inches in landscape format, with the actual printed area being slightly smaller at 8 by 6.5 inches to allow for a margin. For other service providers, you may need to make adjustments to the page format here. Also, some printers require the PDF to set page dimensions beyond the actual paper size being used to allow for the ink to extend all the way to the page border, even after a somewhat inaccurate cutting process trims off the edge. This is known as "full bleed" in the printing process.

The end of the TeX document is heralded by the `outro.tex` template, which merely fires off an `\end{document}` to tell TeX that it's done.

Floating in the Center

Listing 5 shows you a TeX template for an empty page with an optional, centered text module. The template processor replaces the placeholder `<%= text %>` in line 3 with the value contained in the ERB variable `text`. To ensure that the `Huge`-sized text output is centered vertically as a chapter heading, Line 2 uses `\vspace*` to set a fill space above it, while line 4 does the same thing underneath the text. Because the two fill spaces above and below occupy equal areas in line with the rules of fair space allocation, the text ends up at the middle of the page. TeX, by the way, distinguishes between `\vspace` and `\vspace*`: The command without the asterisk is ignored at the top of the page, which – somewhat surprisingly – would make the chapter heading float up to the top of the page.

Good First Impression

The photo book's cover page in Figure 1 displays an image with a line of text in a large font below it; the text is aligned vertically with the image's right margin. Listing 6 introduces a horizontally centered environment to the `cover.tex` template using `\begin{center}` and includes a photo set to a height of three inches thanks to `\includegraphics` from the LaTeX library's *graphics* package. The template processor gets handed an array of photo files later on, and the `<= photos[0] %>` placeholder defines the path to the first file in the list. LaTeX then reserves the space occupied by the image in the document. The PDF driver sets the photo later on after scaling it to the required size.

For photos that need to stand out in a big way, Listing 7 shows a template for a page-filling landscape photo with a caption underneath, as seen on page 3 of the finished photo book in Figure 2. The `\includegraphics{}` command reserves space for an image in the layout. Specifying `height` limits the area vertically so that there is still room on the page for the image caption at the bottom. Line 7 sets the text below the photo with half a centimeter of extra space.

Two Towers

If you want to lay out two photos in portrait format side by side on one page of a book in landscape format, just use the `twotowers.tex` template in Listing 8 (two towers being a metaphor for two images in portrait format). The template references the two photos passed in as parameters by `<%= photos[0] %>` and `<%= photos[1] %>`. The photos on page 4 of the photo book in Figure 2 are two mirror images, the first one a real photo of a piece of broken road, right by the ocean south of San Francisco; its mirrored counterpart was created by the ImageMagick Convert tool's `-flop` option.

Of course, a page of a book in landscape format can also accommodate four landscape photos arranged as a rectangular box. The `four.tex` template in Listing 9 shows the corresponding TeX code. For some attractive spacing between the images, the template shrinks the photos to a

Listing 6: cover.tex

```
01 \vspace*{\fill}
02 \begin{center}
03   \hfill
04   \includegraphics[height=3in,valign=t]{
05     <%= photos[0] %>}
06   \hspace {4cm}
07 \end{center}
08 \vspace{2cm}
09 \begin{center}
10   \hfill \Huge{<%= text %>}
11   \hspace {4cm}
12 \end{center}
13 \vspace{1cm}
```

Listing 7: single.tex

```
01 \null
02 \begin{figure}
03   \centering
04   \includegraphics[height=5.5in,valign=t]{
05     <%= photos[0] %>}
06   \linebreak
07     \par\vspace{.5cm}\par
08   \Huge{\sffamily <%= text %>}
09 \end{figure}
```

Listing 8: twotowers.tex

```
\null
\begin{figure}
  \centering
  \includegraphics[height=5in,valign=t]{
    <%= photos[0] %>}
  \hspace{1cm}
  \includegraphics[height=5in,valign=t]{
    <%= photos[1] %>}
  \linebreak
  \par \vspace{1cm} \par
  \Huge{\sffamily <%= text %>}
\end{figure}
```

Listing 9: four.tex

```
01 \begin{figure}
02   \centering
03   \includegraphics[height=2.5in,valign=t]{
04     <%= photos[0] %>}
05   \hspace{.5cm}
06   \includegraphics[height=2.5in,valign=t]{
07     <%= photos[1] %>}
08   \par
09   \vspace{.75cm}
10   \par
11   \includegraphics[height=2.5in,valign=b]{
12     <%= photos[2] %>}
13   \hspace{.5cm}
14   \includegraphics[height=2.5in,valign=b]{
15     <%= photos[3] %>}
16   \linebreak
17   \par\vspace{.5cm}\par
18   \Huge{\sffamily <%= text %>}
19 \end{figure}
```

height of 2.5 inches and sets the horizontal spacing to 0.5 centimeters in lines 5 and 13. The vertical spacing is set to 0.75 centimeters in line 9.

The underlying text in the `\Huge` font is also centered within the `\centering` environment and ends up 0.5 centimeters below the lower image pair, as specified by line 17. The `\par` statement introduces a new paragraph in TeX, and `\vspace{}` tells the typesetting engine to leave space between paragraphs – in this case, 0.5 centimeters.

Your Own Templates

The templates presented thus far will help you create an attractive photo book. The book consists of the cover, blank pages, chapter headings, and photo pages with one photo in landscape format, two photos in portrait format, and four photos in landscape format – all of them with optional page text. To output the finished photo book in PDF format, all that you need is a script like Listing 1 that retrieves the templates and includes appropriate photos from your hard disk. After cobbling together the code for a TeX document, the final `b.print` will invoke the `xelatex` compiler to turn it into a print-ready PDF.

Of course, you can let your creativity run wild and create arbitrary new TeX templates that you can then call in the Ruby script. Because the entire book definition is a text file and managed by a version control system like Git, you can incorporate changes without worrying about failed attempts or even add parts of an existing book to a new one – I guess you could call it photo book engineering.

Tempus Curat Omnia

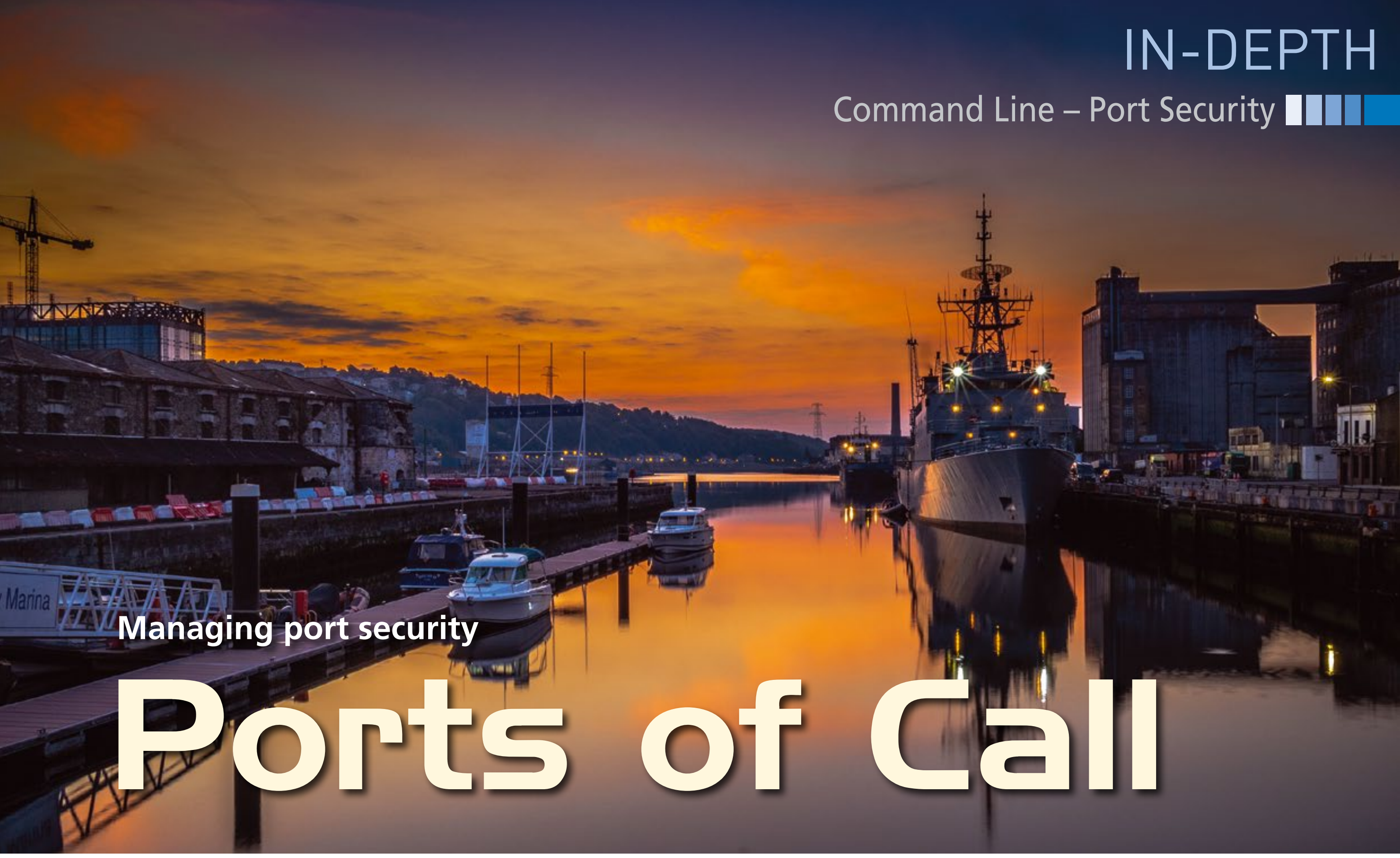
By the way, I found a nice quote in Donald Knuth's TeX book [1], which I read as a refresher for this article – imagine, the first edition was published in 1984! On page 110, he explains why TeX meticulously tries to rearrange paragraphs such that the word spacing in all lines looks as equal as possible – up to the point where a single word at the end of a paragraph can force the entire paragraph text to be reworked. On the other hand, TeX won't look several pages ahead, in terms of pagination, in order to distribute illustrations and text paragraphs as smartly as possible across pages. Knuth also gives a reason for this design decision: The RAM of a typical desktop computer was simply not big enough at the time to hold several pages at once – how times change. ■■■

Info

[1] Knuth, Donald E. *The TeXbook*. Addison-Wesley Professional, 1984: *https://www.amazon.com/TeXbook-Donald-Knuth/dp/0201134489*

[2] Goossens, Michel, Sebastian Rahtz, and Frank Mittelbach. *The LaTeX Graphics Companion*. Addison-Wesley, 1997: *https://www.amazon.com/LaTeX-Graphics-Companion-Illustrating-Postscript/dp/0201854694/*

[3] Schilli, Michael. *Perl Power!* Longman Group, 1998: *https://www.amazon.com/Perl-Power-JumpStart-Guide-Programming/dp/0201360683/*



Managing port security

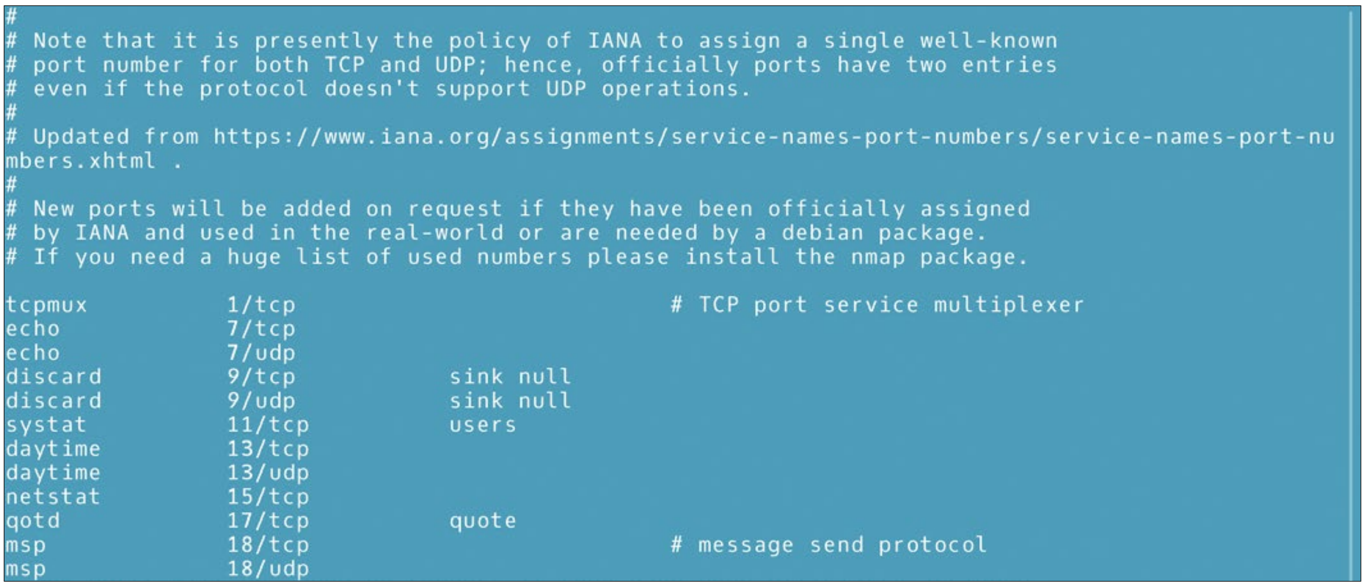
Ports of Call

A few basic commands for working with ports can help you make your small network or standalone system more secure. *By Bruce Byfield*

Ports are a core feature of modern computing. I'm not talking about transferring versions of applications to another architecture or operating system. Instead, I mean the kind of port that is an address for a virtual connection point to or from a computer and another device or server, including the Internet. Ports direct external traffic to the correct application, and this function makes them important for troubleshooting and security. How do you find which ports are open or listening (i.e., currently in use) when there is no need for them to be? What ports are associated with which application or server? How do you know whether any ports are hidden and being used by an intruder? Even if you are working on a standalone computer, knowing how to answer these questions is a basic administrative skill.

Ports can be either hardware or software. Either way, they are treated similarly. Most ports are managed by two protocols: Transmission Control Protocol (TCP) and User Datagram Protocol (UDP). Both TCP and UDP have a range of port numbers, divided into three categories:

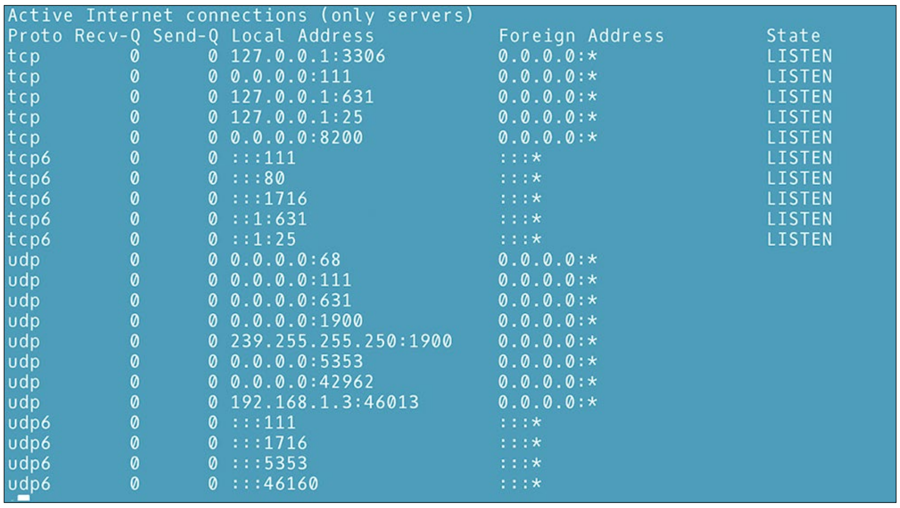
- System ports (numbers 0-1023): These are the most common ports and essential to external



```
#
# Note that it is presently the policy of IANA to assign a single well-known
# port number for both TCP and UDP; hence, officially ports have two entries
# even if the protocol doesn't support UDP operations.
#
# Updated from https://www.iana.org/assignments/service-names-port-numbers/service-names-port-nu
mbers.xhtml .
#
# New ports will be added on request if they have been officially assigned
# by IANA and used in the real-world or are needed by a debian package.
# If you need a huge list of used numbers please install the nmap package.

tcpmux          1/tcp                           # TCP port service multiplexer
echo            7/tcp
echo            7/udp
discard         9/tcp           sink null
discard         9/udp           sink null
systat          11/tcp          users
daytime         13/tcp
daytime         13/udp
netstat         15/tcp
qotd            17/tcp          quote
msp             18/tcp                          # message send protocol
msp             18/udp
```

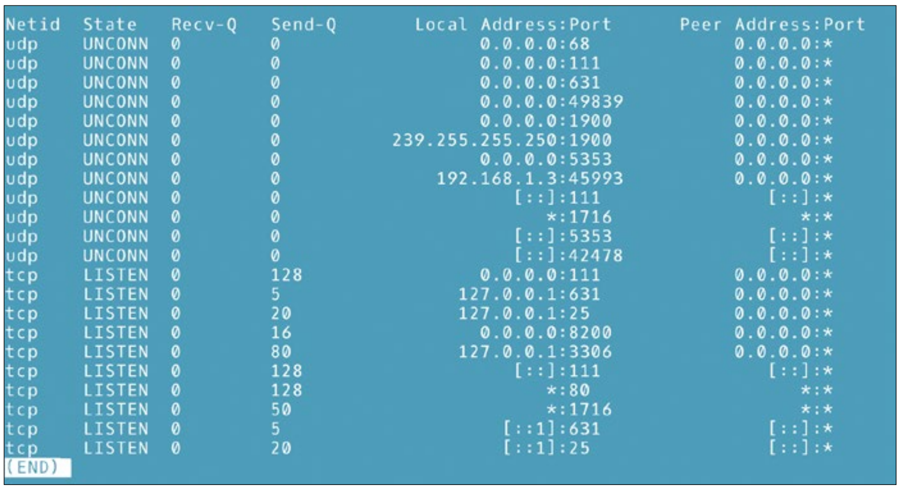
Figure 1: The start of a list of ports using `less`.

Photo by Jamie O'Sullivan on Unsplash



```
Active Internet connections (only servers)
Proto Recv-Q Send-Q Local Address           Foreign Address         State
tcp        0      0 127.0.0.1:3306          0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:111             0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:631           0.0.0.0:*               LISTEN
tcp        0      0 127.0.0.1:25            0.0.0.0:*               LISTEN
tcp        0      0 0.0.0.0:8200            0.0.0.0:*               LISTEN
tcp6       0      0 :::111                  :::*                    LISTEN
tcp6       0      0 :::80                   :::*                    LISTEN
tcp6       0      0 :::1716                 :::*                    LISTEN
tcp6       0      0 ::1:631                 :::*                    LISTEN
tcp6       0      0 ::1:25                  :::*                    LISTEN
udp        0      0 0.0.0.0:68              0.0.0.0:*
udp        0      0 0.0.0.0:111             0.0.0.0:*
udp        0      0 0.0.0.0:631             0.0.0.0:*
udp        0      0 0.0.0.0:1900            0.0.0.0:*
udp        0      0 239.255.255.250:1900    0.0.0.0:*
udp        0      0 0.0.0.0:5353            0.0.0.0:*
udp        0      0 0.0.0.0:42962           0.0.0.0:*
udp        0      0 192.168.1.3:46013       0.0.0.0:*
udp6       0      0 :::111                  :::*
udp6       0      0 :::1716                 :::*
udp6       0      0 :::5353                 :::*
udp6       0      0 :::46160                :::*
```

Figure 2: Open ports displayed using `netstat`.



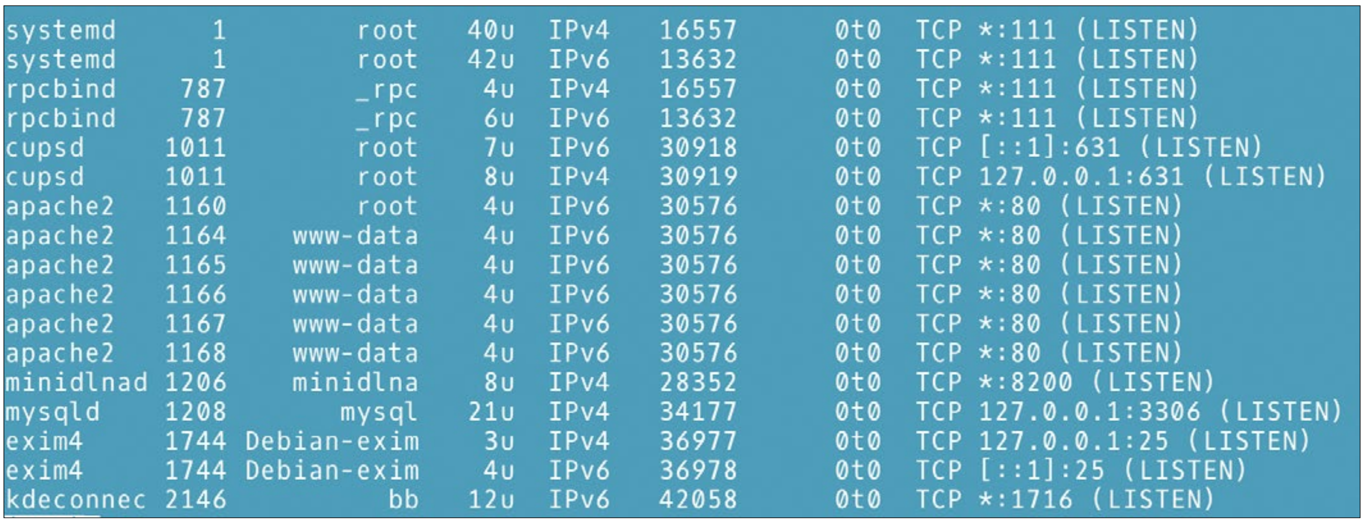
```
Netid  State   Recv-Q   Send-Q       Local Address:Port       Peer Address:Port
udp    UNCONN  0        0                0.0.0.0:68                0.0.0.0:*
udp    UNCONN  0        0                0.0.0.0:111               0.0.0.0:*
udp    UNCONN  0        0                0.0.0.0:631               0.0.0.0:*
udp    UNCONN  0        0                0.0.0.0:49839             0.0.0.0:*
udp    UNCONN  0        0                0.0.0.0:1900              0.0.0.0:*
udp    UNCONN  0        0        239.255.255.250:1900              0.0.0.0:*
udp    UNCONN  0        0                0.0.0.0:5353              0.0.0.0:*
udp    UNCONN  0        0            192.168.1.3:45993             0.0.0.0:*
udp    UNCONN  0        0                   [::]:111                  [::]:*
udp    UNCONN  0        0                      *:1716                    *:*
udp    UNCONN  0        0                   [::]:5353                 [::]:*
udp    UNCONN  0        0                   [::]:42478                [::]:*
tcp    LISTEN  0        128              0.0.0.0:111               0.0.0.0:*
tcp    LISTEN  0        5              127.0.0.1:631               0.0.0.0:*
tcp    LISTEN  0        20             127.0.0.1:25                0.0.0.0:*
tcp    LISTEN  0        16               0.0.0.0:8200              0.0.0.0:*
tcp    LISTEN  0        80             127.0.0.1:3306              0.0.0.0:*
tcp    LISTEN  0        128                 [::]:111                  [::]:*
tcp    LISTEN  0        128                    *:80                      *:*
tcp    LISTEN  0        50                     *:1716                    *:*
tcp    LISTEN  0        5                  [::1]:631                  [::]:*
tcp    LISTEN  0        20                 [::1]:25                   [::]:*
(END)
```

Figure 3: Querying sockets to find open ports.

communication. Sometimes called well-known ports, system ports can only be used by root or privileged users. For example, `ssh` is port 22, and the Network Time Protocol (NTP) is port 123. While these ports can be changed, the changed ports can only communicate with other machines that have also reassigned their ports accordingly.

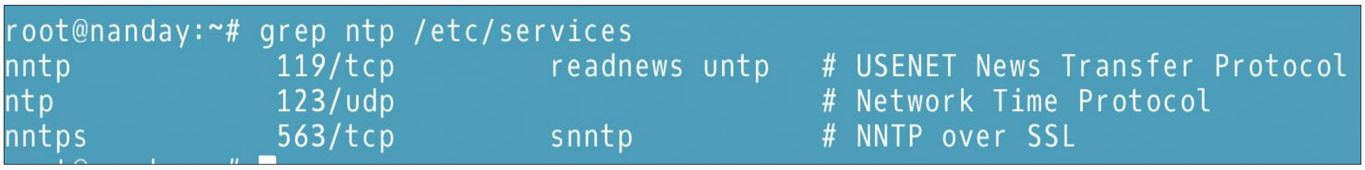
- Registered ports (numbers 1024-49151): These ports are available for processes and applications run by ordinary users. Registered ports are sometimes called user ports.
- Dynamic ports (numbers 49152-65535): These ports can be used by any processes or applications as needed, and they are assigned on the fly. Dynamic ports are also called private or ephemeral ports.

For the sake of thoroughness, a port may have both a TCP and a UDP address assigned by the Internet Assigned Numbers Authority (IANA). As you might expect, each open port is potentially a vulnerability that in theory can be maliciously attacked. That possibility is why security-conscious distributions install a minimal amount of software. It is also why checking for open ports is a basic security precaution. If an application or service is not necessary or currently in use, closing that port reduces the security risk.



```
systemd       1        root   40u  IPv4  16557      0t0  TCP *:111 (LISTEN)
systemd       1        root   42u  IPv6  13632      0t0  TCP *:111 (LISTEN)
rpcbind     787        _rpc    4u  IPv4  16557      0t0  TCP *:111 (LISTEN)
rpcbind     787        _rpc    6u  IPv6  13632      0t0  TCP *:111 (LISTEN)
cupsd      1011        root    7u  IPv6  30918      0t0  TCP [::1]:631 (LISTEN)
cupsd      1011        root    8u  IPv4  30919      0t0  TCP 127.0.0.1:631 (LISTEN)
apache2    1160        root    4u  IPv6  30576      0t0  TCP *:80 (LISTEN)
apache2    1164    www-data    4u  IPv6  30576      0t0  TCP *:80 (LISTEN)
apache2    1165    www-data    4u  IPv6  30576      0t0  TCP *:80 (LISTEN)
apache2    1166    www-data    4u  IPv6  30576      0t0  TCP *:80 (LISTEN)
apache2    1167    www-data    4u  IPv6  30576      0t0  TCP *:80 (LISTEN)
apache2    1168    www-data    4u  IPv6  30576      0t0  TCP *:80 (LISTEN)
minidlnad  1206    minidlna    8u  IPv4  28352      0t0  TCP *:8200 (LISTEN)
mysqld     1208       mysql   21u  IPv4  34177      0t0  TCP 127.0.0.1:3306 (LISTEN)
exim4      1744 Debian-exim    3u  IPv4  36977      0t0  TCP 127.0.0.1:25 (LISTEN)
exim4      1744 Debian-exim    4u  IPv6  36978      0t0  TCP [::1]:25 (LISTEN)
kdeconnec  2146          bb   12u  IPv6  42058      0t0  TCP *:1716 (LISTEN)
```

Figure 4: Using `lsof` to find open ports.



```
root@nanday:~# grep ntp /etc/services
nntp            119/tcp         readnews untp   # USENET News Transfer Protocol
ntp             123/udp                         # Network Time Protocol
nntps           563/tcp         snntp           # NNTP over SSL
```

Figure 5: Finding port numbers by grepping `/etc/service`.

Getting a List of Ports

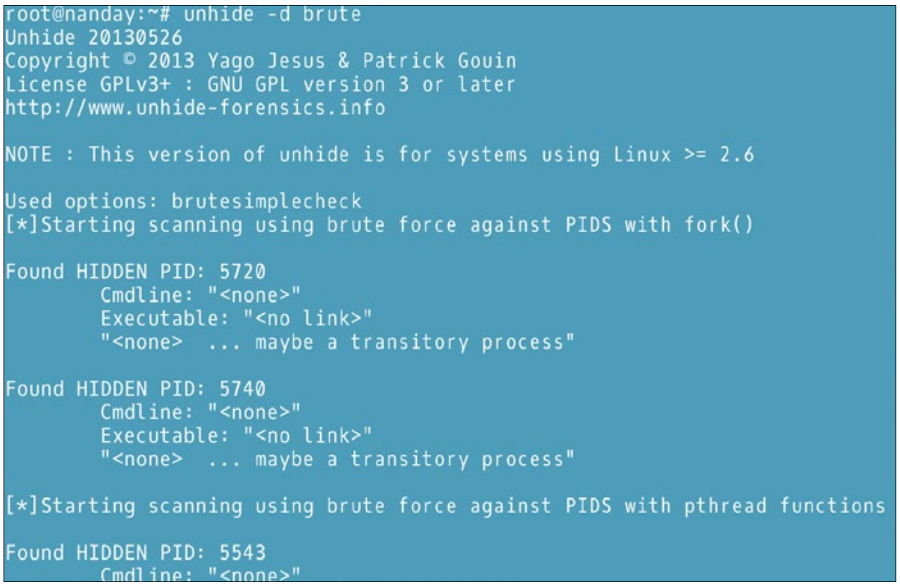
Ports are usually assigned when a package is installed. The quickest way to get a complete list of ports is to log in as root and enter:

```
less /etc/services
```

You will need the `less` command because there are hundreds of ports on the typical computer. In the output from `less`, the left-hand column shows the name of the service, and the second column from the left shows the port number and whether it uses the TCP or UDP protocol (Figure 1). Services may be abbreviated, so the third column may give the full name. The fourth column gives any explanatory notes. If you look at the output for another Linux computer, you will find most – perhaps all – the system ports are the same. As the name implies, many of the registered ports will also be the same on different computers.

Checking for Open Ports

Security is always balanced against user convenience. However, each distribution sets that balance differently. For this reason, immediately after installation, users may want to check which ports are open and consider which ones they might want to close. Users should also check the open ports periodically and investigate any ports



```
root@nanday:~# unhide -d brute
Unhide 20130526
Copyright © 2013 Yago Jesus & Patrick Gouin
License GPLv3+ : GNU GPL version 3 or later
http://www.unhide-forensics.info

NOTE : This version of unhide is for systems using Linux >= 2.6

Used options: brutesimplecheck
[*]Starting scanning using brute force against PIDS with fork()

Found HIDDEN PID: 5720
        Cmdline: "<none>"
        Executable: "<no link>"
        "<none>  ... maybe a transitory process"

Found HIDDEN PID: 5740
        Cmdline: "<none>"
        Executable: "<no link>"
        "<none>  ... maybe a transitory process"

[*]Starting scanning using brute force against PIDS with pthread functions

Found HIDDEN PID: 5543
        Cmdline: "<none>"
```

Figure 6: Using the `brute` technique with `unhide`.

that they do not recognize or are uncertain about. Several tools can be used, including `netstat` as follows:

```
netstat -lntu | less
```

The `-l` option displays only listening or open sockets. With the `-n` option, the port number is shown, while `-t` includes TCP ports and `-u` includes UDP ports (Figure 2).

An alternative is to query the sockets using:

```
ss -lntu | less
```

The options are the same as for `netstat`. The relevant information in the output is the `State` column second to the left, and the `Local Address:Port` in the fifth column from the left (Figure 3).

Perhaps the best choice is to pipe `lsof` through `grep`:

```
lsof -i -P -n | grep
```

In this command, *-i* lists matching files with the current Internet address. For network files, option `-P` suppresses the conversion of port numbers to port names, and `-n` suppresses the conversion of network numbers to hostnames. The result is a concise presentation of relevant information. Depending on your needs, you might want to omit some of these options. When managing a network, you can also use `-i` to specify an Internet address (Figure 4).

Finding a Port Number

Computer ports are too numerous for anyone to remember them all. At best, an administrator might memorize a few system and registered ports. Yet troubleshooting and security often require you to know a specific port. I know of at least one online crib sheet [1], but it may be quicker to look up ports on your own system. You can use `grep` to search `/etc/services`. You can even search for a specific service (Figure 5) using:

```
grep SERVICE /etc/service
```

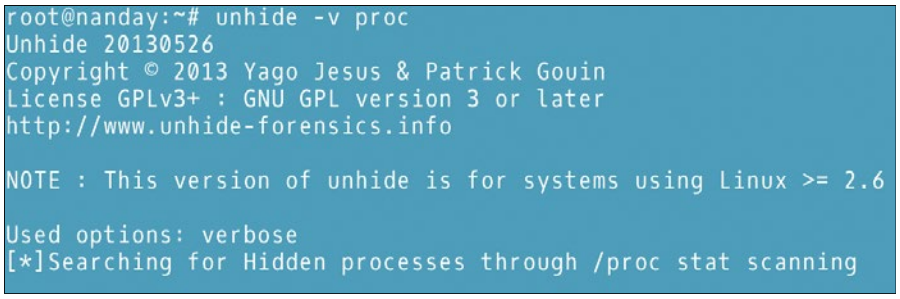
Several related results may display, one for each protocol in use.

Discovering Hidden Ports

Security breaches often go unnoticed because they use hidden ports. You can discover any hidden ports using the `unhide` utility. Two versions are available online, the general `unhide` command and the more specific `unhide-tcp`. Both have similar functions, but it appears that `unhide-tcp` may be obsolete. Usually, it is wise to run `unhide` with the `-f` option, which generates a text log called `unhide-linux.log` in the current directory so you can refer back to the output.

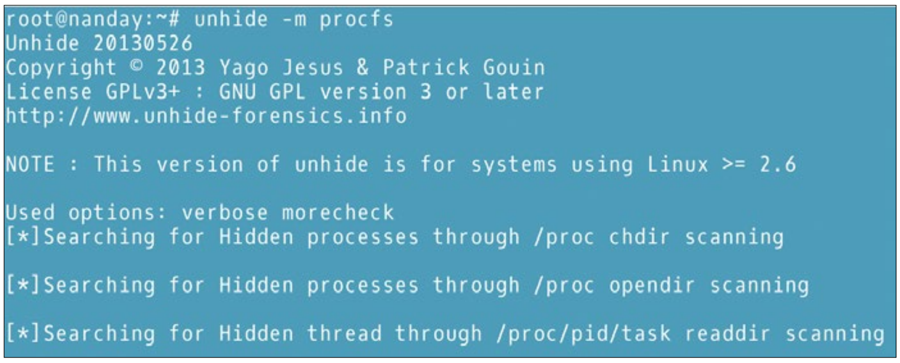
The `unhide` utility uses several techniques, which are entered as sub-commands after the main command, without any hyphens. Some techniques may take several minutes to run. The techniques include:

- The `brute` technique: Checks every process ID. It should be run twice



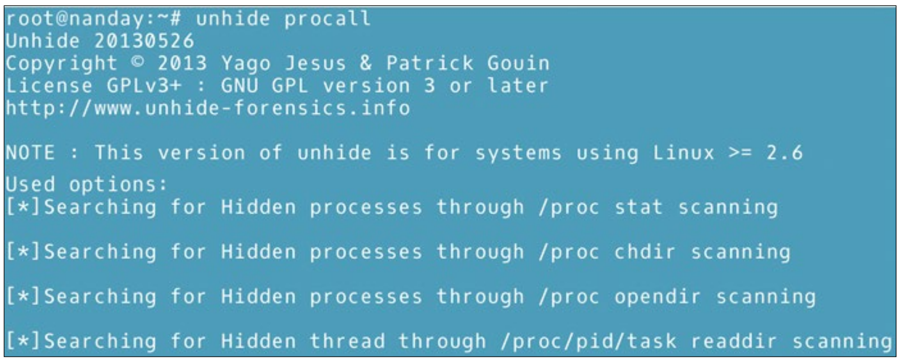
```
root@nanday:~# unhide -v proc
Unhide 20130526
Copyright © 2013 Yago Jesus & Patrick Gouin
License GPLv3+ : GNU GPL version 3 or later
http://www.unhide-forensics.info

NOTE : This version of unhide is for systems using Linux >= 2.6

Used options: verbose
[*]Searching for Hidden processes through /proc stat scanning
```

Figure 7: Running `unhide` with the `proc` technique in verbose mode.



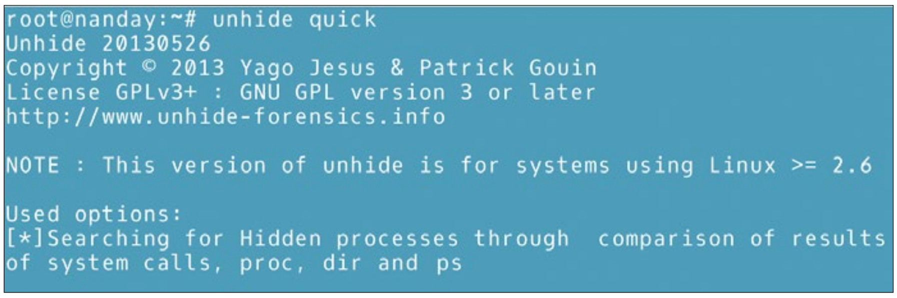
```
root@nanday:~# unhide -m procfs
Unhide 20130526
Copyright © 2013 Yago Jesus & Patrick Gouin
License GPLv3+ : GNU GPL version 3 or later
http://www.unhide-forensics.info

NOTE : This version of unhide is for systems using Linux >= 2.6

Used options: verbose morecheck
[*]Searching for Hidden processes through /proc chdir scanning

[*]Searching for Hidden processes through /proc opendir scanning

[*]Searching for Hidden thread through /proc/pid/task readdir scanning
```

Figure 8: Using the `procfs` technique with `unhide`.



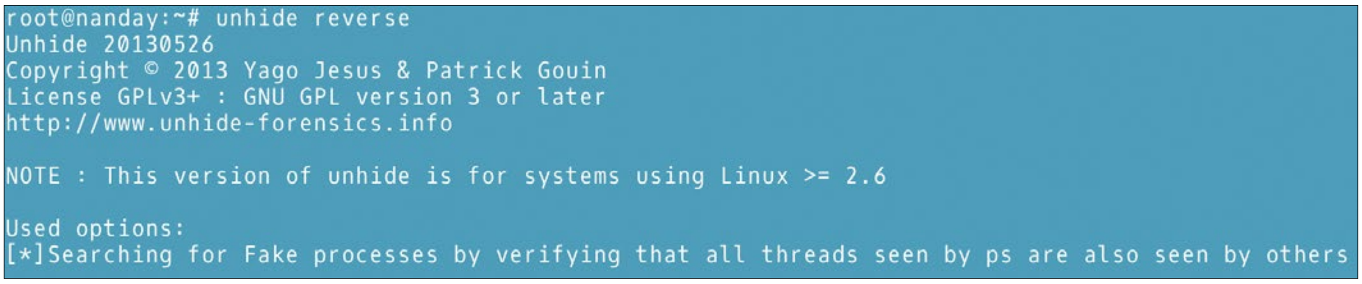
```
root@nanday:~# unhide procall
Unhide 20130526
Copyright © 2013 Yago Jesus & Patrick Gouin
License GPLv3+ : GNU GPL version 3 or later
http://www.unhide-forensics.info

NOTE : This version of unhide is for systems using Linux >= 2.6
Used options:
[*]Searching for Hidden processes through /proc stat scanning

[*]Searching for Hidden processes through /proc chdir scanning

[*]Searching for Hidden processes through /proc opendir scanning

[*]Searching for Hidden thread through /proc/pid/task readdir scanning
```

Figure 9: The `procall` technique running in `unhide`.



```
root@nanday:~# unhide quick
Unhide 20130526
Copyright © 2013 Yago Jesus & Patrick Gouin
License GPLv3+ : GNU GPL version 3 or later
http://www.unhide-forensics.info

NOTE : This version of unhide is for systems using Linux >= 2.6

Used options:
[*]Searching for Hidden processes through  comparison of results
of system calls, proc, dir and ps
```

Figure 10: Use the `quick` technique in `unhide` to start searching for hidden ports.



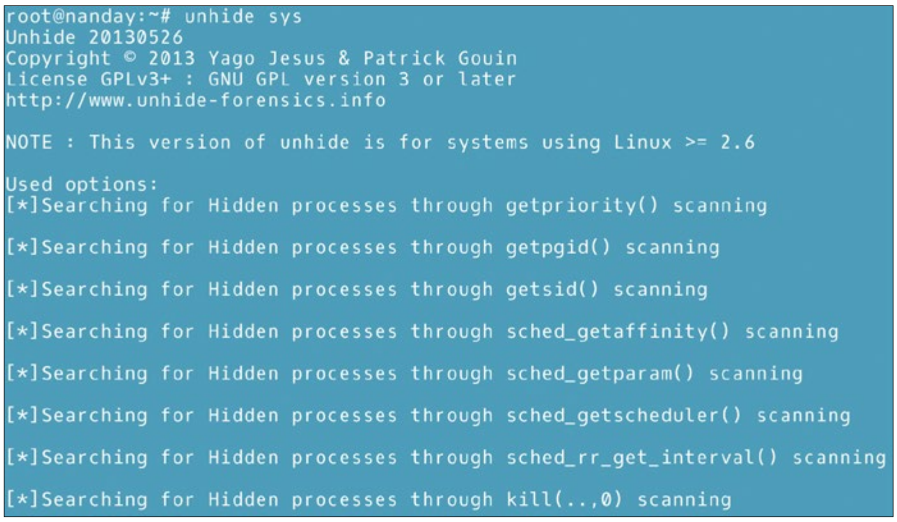
```
root@nanday:~# unhide reverse
Unhide 20130526
Copyright © 2013 Yago Jesus & Patrick Gouin
License GPLv3+ : GNU GPL version 3 or later
http://www.unhide-forensics.info

NOTE : This version of unhide is for systems using Linux >= 2.6

Used options:
[*]Searching for Fake processes by verifying that all threads seen by ps are also seen by others
```

Figure 11: Using `unhide`'s `reverse` technique is a quick way to search for hidden ports.

using the `-d` option to avoid false positives, such as an application running at the command line. A false positive may occur when a temporary process uses a port, and it should disappear when the process finishes. You may want to repeat this command several times to ensure that the temporary process has finished. In the documentation, this choice is called the brute force option (Figure 6).

- The `proc` technique: Compares `/proc` with `/bin/ps`. On a clean system, the output of the two directories should match. It may be helpful to add the `-v` option for verbose output (Figure 7).
- The `procfs` technique: Compares information from `/procfs` and `/bin/ps`. Add the `-m` option to run additional tests (Figure 8).
- The `procall` technique: Combines the `proc` and `procfs` tests (Figure 9).
- The `quick` technique: Combines the `proc`, `procfs`, and `sys` techniques and is about 20 times faster. The speed may result in more false positives, so the techniques might need to be run separately afterwards (Figure 10).
- The `reverse` technique: Checks that all threads in `/bin/ps` are also in `/procfs` and seen by system calls. This technique is specifically designed to see if a rootkit is making `/bin/ps` show a fake process (Figure 11).
- The `sys` technique: Compares information gathered from `/bin/ps` with information gathered from system calls (Figure 12).

When using `unhide`, be prepared to run several tests. Run the `quick` technique first and then other techniques to eliminate any false positives. The `reverse` technique also returns speedy results. As an extra precaution, run other tests to confirm the results.



```
root@nanday:~# unhide sys
Unhide 20130526
Copyright © 2013 Yago Jesus & Patrick Gouin
License GPLv3+ : GNU GPL version 3 or later
http://www.unhide-forensics.info

NOTE : This version of unhide is for systems using Linux >= 2.6

Used options:
[*]Searching for Hidden processes through getpriority() scanning

[*]Searching for Hidden processes through getpgid() scanning

[*]Searching for Hidden processes through getsid() scanning

[*]Searching for Hidden processes through sched_getaffinity() scanning

[*]Searching for Hidden processes through sched_getparam() scanning

[*]Searching for Hidden processes through sched_getscheduler() scanning

[*]Searching for Hidden processes through sched_rr_get_interval() scanning

[*]Searching for Hidden processes through kill(..,0) scanning
```

Figure 12: The `sys` technique used in `unhide` compares information gathered from `/bin/ps` with system calls.

Levelling Up

I've covered the basic commands for working with ports. You'll find many of these commands installed with sets of utilities, and all commands should be found in the repositories of most distributions. For large networks, you also might want to install `nmap`. `nmap` does numerous tests, but when checking ports it provides an interesting ports table, which lists the port number and protocol, service name, and the port's state (open, closed, filtered, or unfiltered – filtered meaning that a firewall or other network element blocks the port). However, `nmap` is often overkill. For small networks or standalone systems, the commands described in this article should be more than adequate for controlling ports. ■■■

Info

[1] Default port numbers: *https://geekflare.com/default-port-numbers/*

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/*.

Create cartoons and anime with OpenToonz

Toon Time

OpenToonz is a professional animation tool for comic and manga artists. *By Ralf Kirschner*

Many readers associate the terms “manga” and “anime” respectively with Japanese comics and animated series, such as *Dragon Ball* or *Pokemon*, but today, manga and anime are global phenomena, and artists all over the world devote their attention and effort to the forms.

If you want to experiment with animation yourself, there is no need to stick to pen and paper. OpenToonz [1] is an open source tool that meets the highest animation standards. The program was partly developed in cooperation with famous animation companies such as Studio Ghibli [2], the creative minds behind classic anime films such as *Princess Mononoke*, *Chihiro’s Spirited Away*, and *How the Wind Rises*.

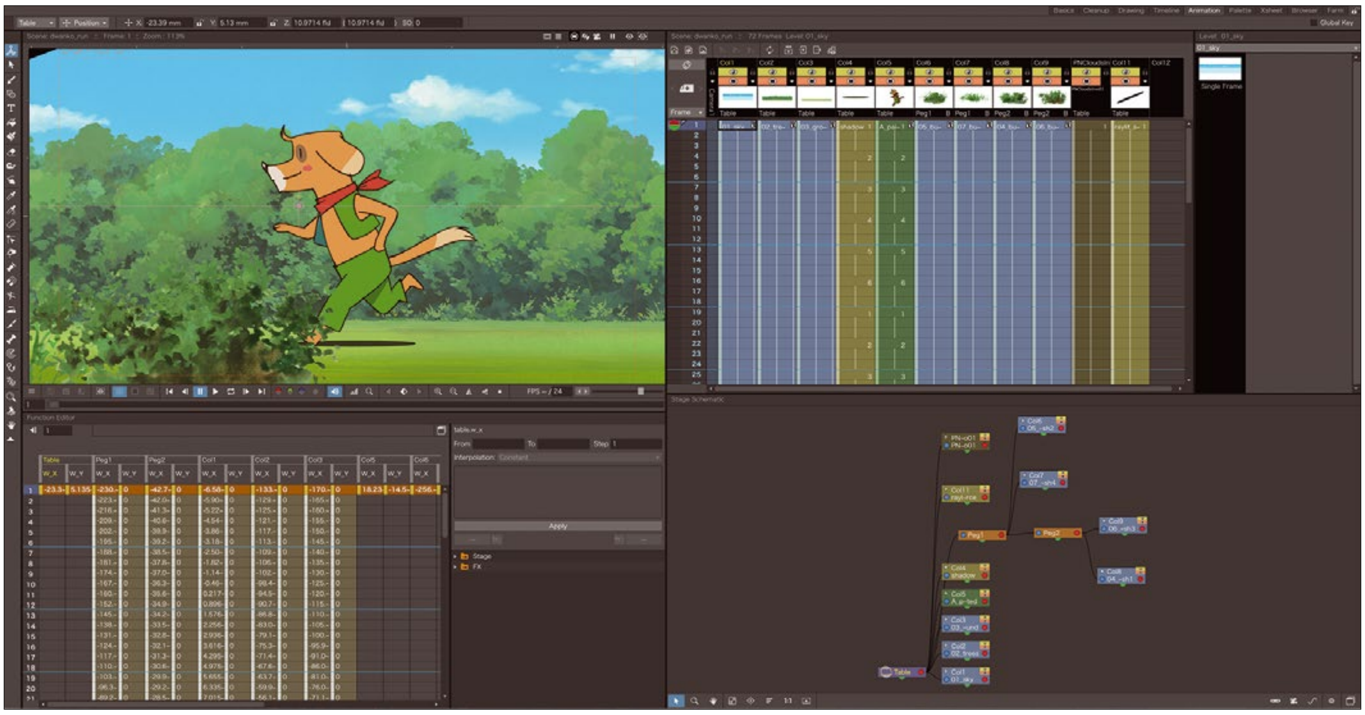


Figure 1: **The example animation provided for OpenToonz demonstrates the most important features.**

Photo by Dex Ezekiel on Unsplash

OpenToonz is available in the package sources of most popular Linux distributions, so installing should be easy. To test the software with a sample scene, download the sample animation as a ZIP archive from the project page [3] (`OpenToonz_sample.zip`). You can view this sample on YouTube without installing OpenToonz [4]. Figure 1 shows OpenToonz with the official sample, which stars a dog named Dwanko.

Basic Terms

The classic animation workflow starts with a script and the design of the acting characters. The storyboard is then created from the script. The storyboard contains at least one drawing for each shot that shows the camera position, the movement of the characters, and the type of background. There is also an *exposure sheet*, called an X-sheet or dope sheet for short. The X-sheet is a frame-by-frame script for each individual shot. You can see an example in the upper right area of Figure 1.

Animators often avoid the need for intermediate drawings by combining smart framing with camera motion over still images. In dialog scenes, for instance, the graphic artist only needs to animate the mouth movement. Drawings are divided into different layers: These individual layers are visible as columns in the X-Sheet.

Although OpenToonz is a 2D animation tool, you don't have to sacrifice a three-dimensional feel. Using motion parallax, you can easily add depth to animations. The parallax effect occurs when objects in the foreground move faster than objects in the background. Figure 2 and Figure 3 illustrate this effect by showing the first and last frames of the animation sequence in the OpenToonz 3D view.

Normally, at least three planes are used to create this 3D effect: a cloud plane at the back, one or more vegetation plane(s) in the middle, and a ground plane at the front. In Figure 3, however, in the area of the vegetation plane(s), there is also the plane with Dwanko the dog walking on his hind legs and a projected shadow moving slightly back and forth, which reinforces the spatial impression.

As Figure 4 shows, using only six different drawings can define Dwanko's motion. Each drawing is maintained for three frames before the next pose follows. This process runs in a loop. The illusion of a running dog is also created by the moving floor plane. It is also possible to change the Z coordinates for the individual planes and for the camera, thus adding to the motion effect.

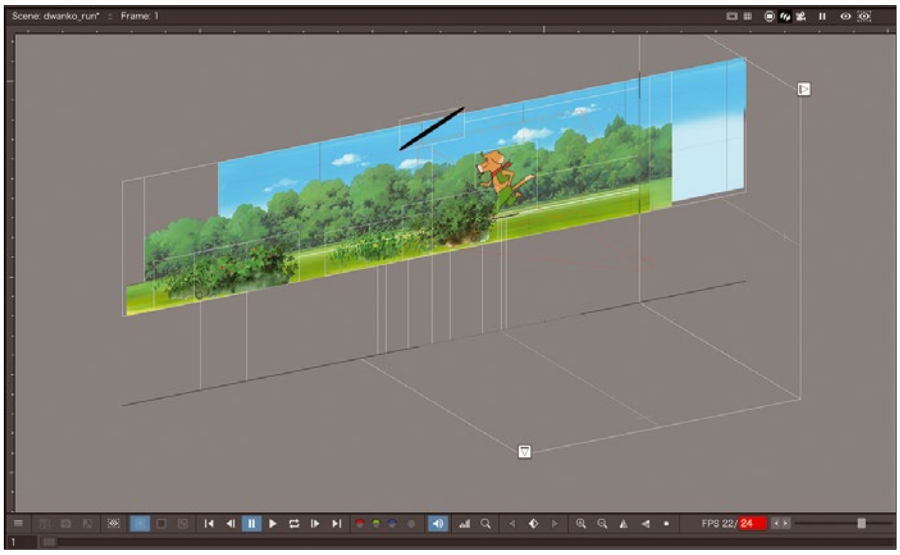


Figure 2: The starting point of the animation: Moving the backgrounds at different speeds creates a parallax effect.

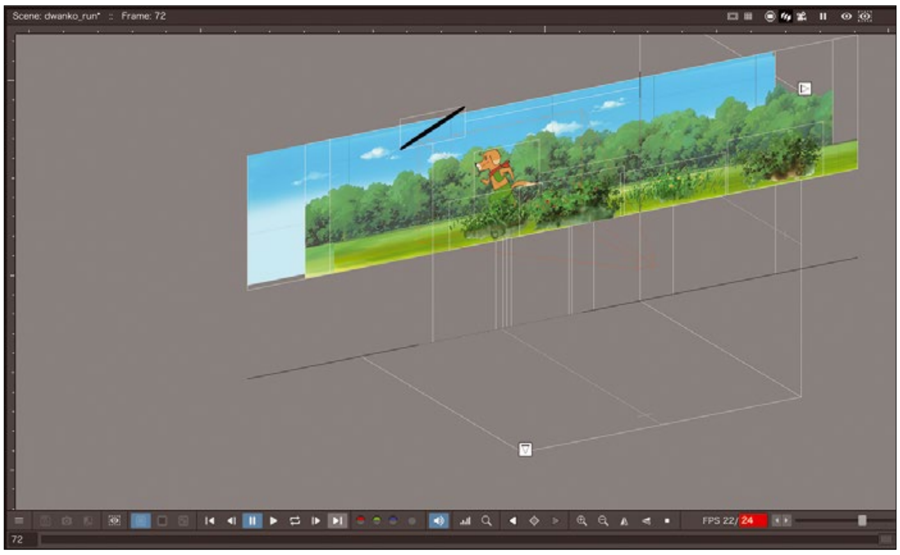


Figure 3: End point of the animation: Animating the dog enhances the motion effect.

Desktop

OpenToonz is divided into rooms, also known as *workspaces*. Each room contains a different collection of windows, which the program displays at special positions on the screen. A list of the rooms appears at the far right edge of the main window, below the menu bar. Some predefined rooms include the *Drawing*, *Palette*, *Animation*, *Xsheet*, and *Browser* rooms. You can also define your own rooms that combine the windows you need for a specific task. Almost all rooms contain the viewer window with the flipbook bar below it, playback and frame rate controls, a toolbox with drawing elements, and an X-Sheet window.

Another important window is the *Function Editor*, which is shown in Figure 1, bottom left. You can use the Function Editor to relate objects and effect transformations to key values and matching interpolations using a table or a graphical editor. The left side of the Function Editor shows a table of transformation values. On the right is the interpolation of the current transformation segment, along with a hierarchical representation of the available objects and effects.

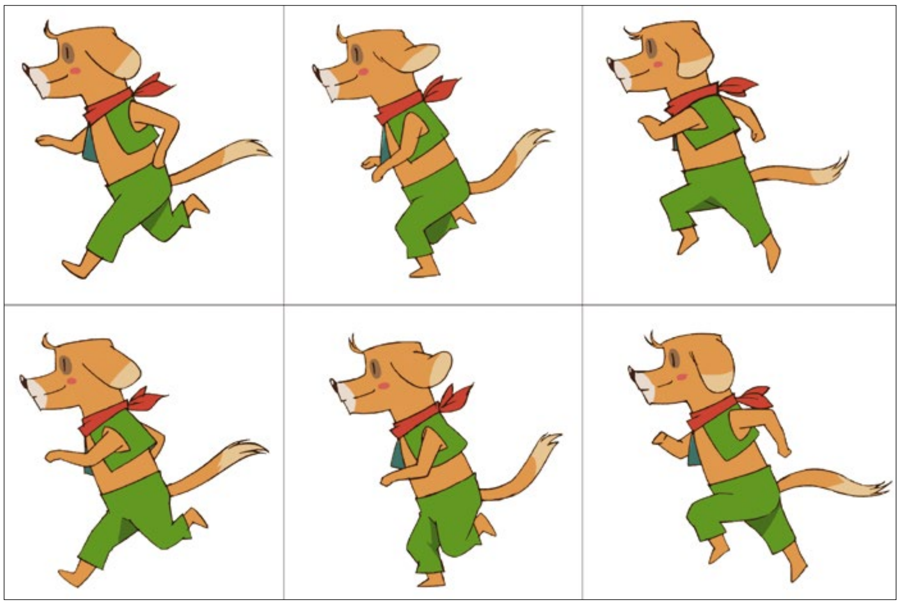


Figure 4: Just six poses are all it takes to make Dwanko run through the landscape.

The values defined in the keyframes are highlighted in yellow. Between them are the values calculated by reference to the interpolation method. Each column represents a single animation-capable parameter, and each row shows the value of the parameter for each frame. Empty cells represent constant values without change in the absence of corresponding keyframe definitions.

The *Stage* folder contains subfolders for the defined cameras, the table, the *pegbars*, and the layers used. A pegbar is a kind of ruler that groups several superimposed layers and ensures that they are transmuted together.

The *FX* folder contains subfolders for each effect. Each folder contains transformation parameters associated with the keys. The object and effect tree uses different icons to indicate whether the object or effect is animated and which parameter changes. An icon with an arrow indicates an animated object or effect. A closed folder indicates a non-animated object or effect. An icon with a dotted straight line symbolizes non-animated parameters, and an icon with a continuous curve indicates animated parameters.

You can save the transformation data and load it as a CURVE file, which means it can be exported to other scenes. You also have the option to export the data in the form of a DAT file for use in other programs.

Animations

Other important windows are the X-Sheet and Level Strip windows, shown in Figure 5. The X-Sheet, which lets you control the timing of all elements in a scene, is organized in the form of a table. Each column contains a level of animation and each row represents the set of all levels that will be displayed in the frame. Each cell represents the contents of the column in a specific frame.

You can load a wide variety of things into the scene, including animation levels, background images and overlays, and video clips. The viewer lets you view and check the contents. You can play back the animation and modify scene content as needed.

Because complex animations are not created by moving individual objects and layers in the Function Editor alone, you will also find the *Level Strip* window, which you can see on the far right in Figure 5. Images that you select in the Level Strip appear in the viewer window. Clicking on a row in the X-Sheet brings up the frame in the viewer window and updates the display of the Playback and Frame Rate controls in the Flipbook bar. In the column header of the X-Sheet window, you can hide individual levels and control their opacity via color filters.

In Figure 1, you can see the *Stage Schematic* window bottom right. The stage schematic contains nodes for all the objects used in the scene, letting you change the way the objects are connected. The schematic also shows the camera and lets you define more cameras.

Defining Levels

By default, the stage schematic window contains the table node and a camera node. Column or layer nodes appear automatically as soon as the content is loaded or created. Pegbars, cameras, and object nodes can be added at any time, and the program automatically links pegbars to the animation table. After selecting an object, you can arrange the object nodes or apply operations, such as cut, copy, and delete. It is also possible to group multiple nodes to create a single node.

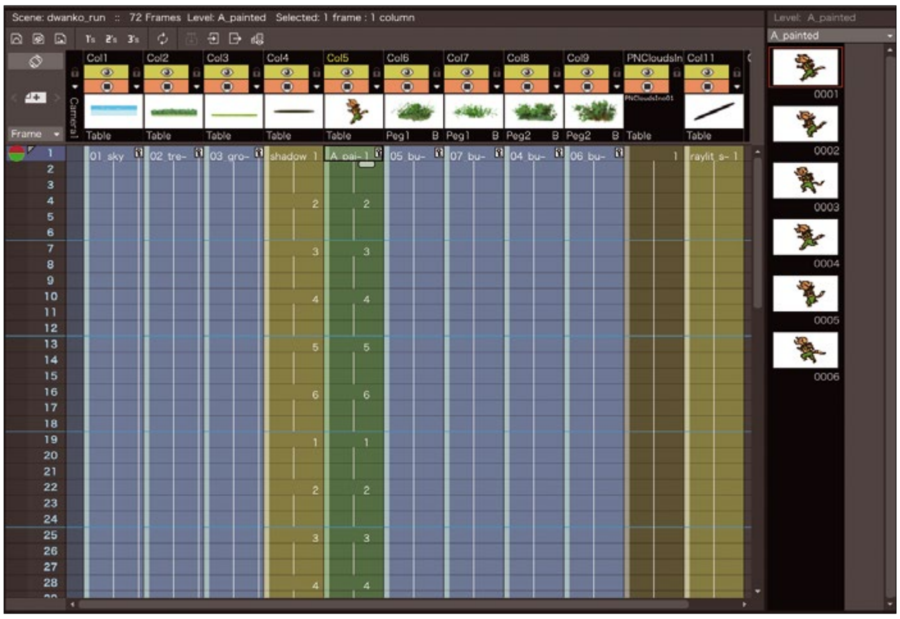

Scene: dwanko_run :: 72 Frames Level: A_painted Selected: 1 frame : 1 column

Figure 5: The X-Sheet controls the timing of all elements in a scene.

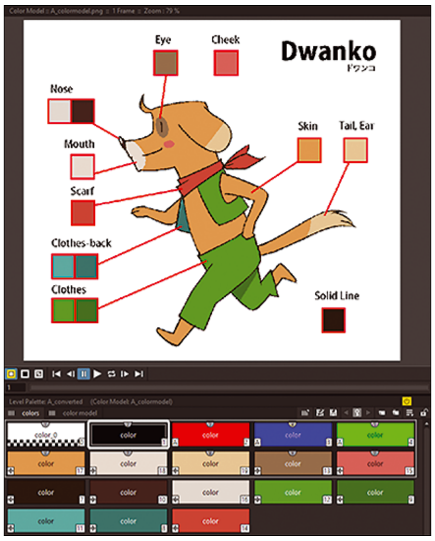


Figure 6: Dwanko's color scheme. A clearly defined scheme makes it easier to draw the different poses.

OpenToonz offers three types of levels, each appearing in a different color on the X-Sheet. *Toonz Vector Levels*, which appear in a yellowish tone, are the best option if you wish to draw directly in OpenToonz. *Toonz Raster Levels*, marked in green, are usually used when drawings have been scanned and there is a need to fill in areas with a predetermined color scheme. *Raster Levels*, which appear in the cells of the X-Sheet in a bluish tone, are used, for example, for colored backgrounds.

The color scheme of an animated film production is usually defined at the beginning of the production and is mandatory in order to have an exact reference for all character drawings. Figure 6 shows the color scheme defined for Dwanko the dog.

The OpenToonz toolbar appears on the left edge of the screen in Figure 7. Some of the tools can only be applied to Toonz vector levels: in particular, the *Control Point Editor*, *Pinch Tool*, *Pump Tool*, *Magnet Tool*, *Bender Tool*, *Iron Tool*, and *Cutter Tool*.

Start Your Own Project

Now that you are familiar with the OpenToonz interface, it is time to create your first animation. If you haven't already done so, install and start OpenToonz; a welcome dialog appears first (Figure 7). OpenToonz manages the scenes of your movie productions and the files used in them as projects. By default, the project name is `sandbox`. See the OpenToonz manual for more information on starting a project [5].

The first step is to create a scene in the Sandbox project. If you want to share your creations with other collaborators later, the browser room in OpenToonz is very useful, regardless of the operating system. In the *scene* subfolder, click on a scene file (`.tnz` suffix), and then export it via the *Export Scene* context menu.

On the start screen, in the *Create a New Scene* frame, enter a scene name of your choice in the *Scene Name* field and confirm it by pressing the *Create Scene* button. For your first custom animation, first create three layers: a Toonz Vector level for the horizon and the sky, which needs to contain a gradient, a Toonz Raster level for the clouds, and a Toonz Vector level for the buildings and streets.

For the horizon and sky layers, select the *Basics* room and then *Level* | *New* | *New Vector Level*. You will want to give the levels meaningful names, such as `sky`. In the *Level Palette* window, click on the plus sign to add a sky color, and set the color picker to a bluish color for the new *color_2*.

Drawing Backgrounds

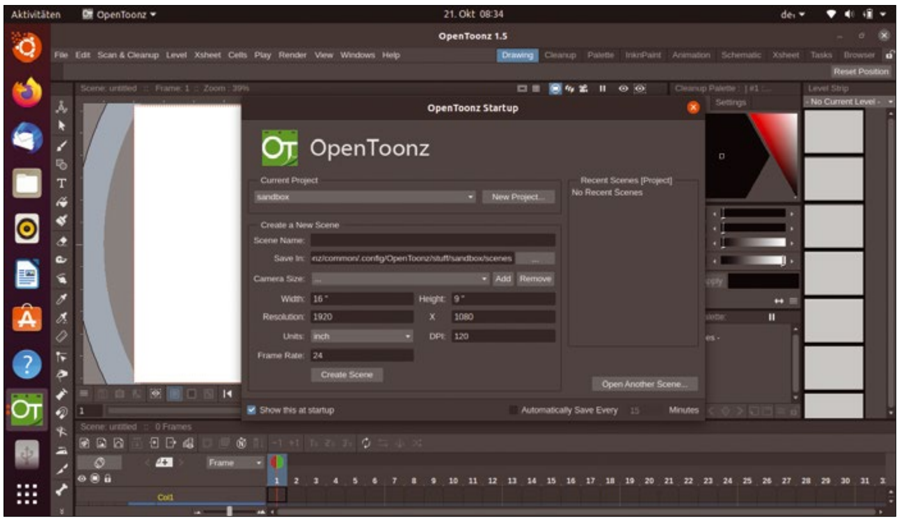
To draw a background, select the *Geometric Tool* (or press *G*) on the far left. Make sure that you select *Rectangle* in the upper section of *Shape*. Then use the mouse pointer to draw a rectangle that covers the entire viewer window. Activate the *Fill Tool* (*F*) in the toolbox. In the *Level Palette* window, select *Vector*, and then *Linear Gradient* (to the right of the green stripe pattern). The rectangle you have just drawn will then become invisible, but this will change as soon as you click on its center, when it will be filled with a simple gradient from left to right.

The green-blue gradient is not yet fit for the purpose of displaying the horizon. In the *Level Palette* window, you will see the two colors at the bottom. Use the color picker to change them to your liking. You can correct the direction of the gradient using the *Angle* parameter; in the example, I needed a 90-degree rotation. The *Y Position* parameter shifts the position of the horizon.

For the cloud layer, select *Level* | *New* | *New Toonz Raster Level* and enter `cloud` as the name of the level. Then, in the *Level palette*, use the color picker to change *color_1* to white. Then activate the *Brush Tool* (*B*), and in the *Level Palette*, open the *Raster* tab. In the upper area of the *Raster* tab, you will find a *Clouds* brush type, which you can use to paint some beautiful clouds in the sky with the mouse.

Finally, create a third layer of the *Vector* level type named `street`. Draw the skyline of a city made of rectangles and triangles here (*Shape* type *Polygon* with three *Polygon Sides*). Now select the *Camera Stand View* (white rectangle in a circle) in the upper part of the viewer window and zoom out a bit with the mouse wheel. The viewer window should look something like Figure 8.

At the moment – to compare it with the analog world, all I have is several


Aktivitäten
OpenToonz
21. Okt 08:34
OpenToonz 1.5
File Edit Scan & Cleanup Level Xsheet Cells Play Render View Windows Help
Drawing Cleanup Palette InknPaint Animation Schematic Xsheet Tasks Browser
Reset Position
Level Strip
Cleanup Palette
Settings

OpenToonz Startup

OpenToonz

Current Project: sandbox — New Project...

Recent Scenes [Project]: No Recent Scenes

Create a New Scene

Scene Name:

Save In: nz/common/.config/OpenToonz/stuff/sandbox/scenes

Camera Size: ... — Add — Remove

Width: 16 " Height: 9 "

Resolution: 1920 X 1080

Units: inch DPI: 120

Frame Rate: 24

Create Scene

Open Another Scene...

[✓] Show this at startup

Automatically Save Every 15 Minutes

Figure 7: Use the start screen to set up your first project. The toolbar appears at the left edge of the window.

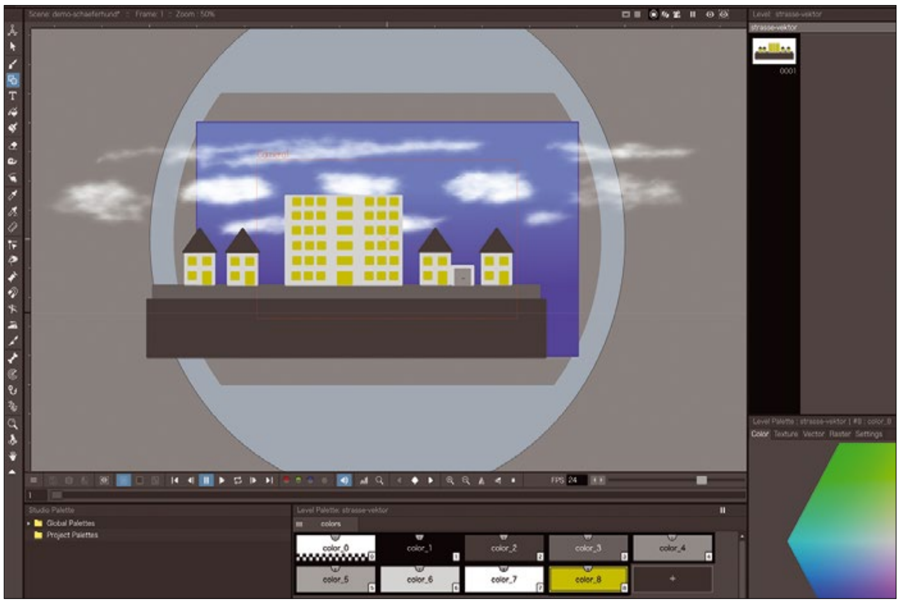


Figure 8: The viewer window and the adjacent areas with sky, clouds, street, and skyline.

slides sitting on top of each other on a table under a camera. What is still missing is the actual animation. To see the animation, switch to the *Animation* room, where you will see a picture like Figure 9.

An animation usually contains several frames. Whereas Figure 9 only defines a single frame, six frames are combined in Figure 10. However, combining frames alone does not create an animation because the individual levels/slides do not change their content or their positions on the table. It makes sense to start by increasing the number of frames with the *Smart Fill Handle*.

Clouds and Landscapes

The *Smart Fill Handle* appears when you click in one of the table cells in Figure 10. Select the first cell in each column and drag the *Smart Fill Handle* down a few rows.

In practice, it would make little sense to create an animation with just six frames. We chose this restriction to keep things simple and to illustrate the controls in Figure 10. What you need now are at least horizontal shifts of the individual slides. Click in the X-Sheet window on the cell in the first frame of the *Col2* column; as you will recall, its name is `clouds`.

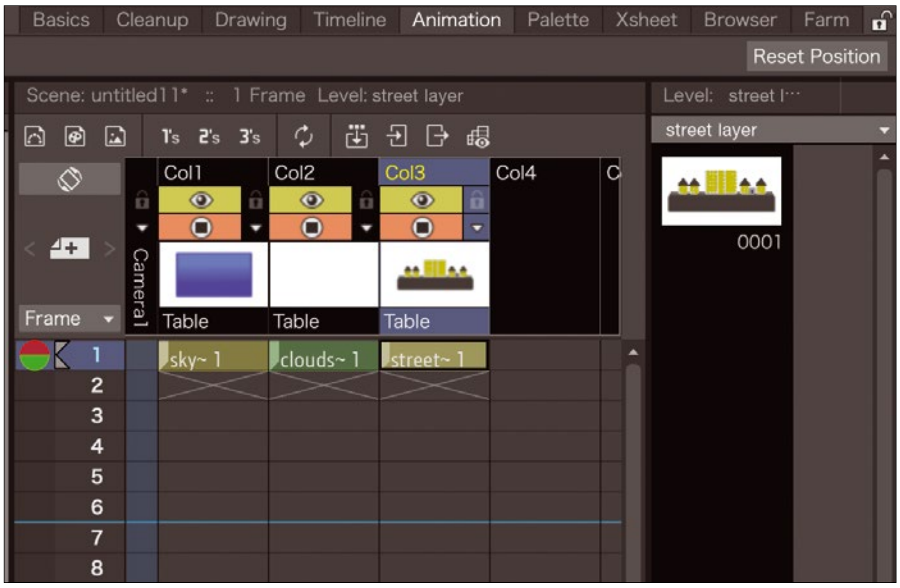


Figure 9: The X-Sheet window with the Level Strip window after completing the sky, clouds, street, and skyline.

Then select the *Animate Tool* (*A*) in the toolbar at the top left. Make sure that *Col2* and *Position* are selected in the *Tool Option Bar* above the viewer window. Now move the cloud layer inside the viewer window a little to the left with the mouse pointer. In the X-Sheet window in the first row, a small white key will then indicate on the right that you successfully defined a keyframe.

Now select the cell in row six in the same column and move the cloud layer inside the viewer window with the mouse pointer – this time to the right. A small white key now appears in row six. Do the same with the street layer in the *Col3* column. Now click on the play icon in the playback controls in the flipbook bar below the viewer window to enjoy your first simple animation.

People and Actors

To complete the animation, you still need one or more levels with people, animals, or machines – actors, to use the industry term – in the foreground. OpenToonz offers several ways to add actors.

If you want to use a character from a real movie as source material, export the desired scene as a GIF file. You can then open the file as layers in Gimp, and then crop the actor frame by frame. Save the result as a transparent PNG file with consecutive numbering.

To import into OpenToonz, select *Level* | *Load Level...* and open the directory in which the individual images are located. OpenToonz then only offers you one file for all of the PNG files you just saved and numbered consecutively; in addition, the dialog tells you the number of individual graphics and the total size of all files.

After you click *Load*, the software loads the individual files and inserts them as a new column into the X-Sheet, like the one shown in Figure 5. However, in Figure 10, notice that the view only displays the file name for the first image in each frame – with a tilde followed by a 1.

In the case of the sky, cloud, and road layers, displaying the first image in each frame is fine because the animation is created by moving the layers. In Figure 5, the numbers change in some lines and are again followed by vertical strokes. Vertical strokes mean: Repeat the content of the previous frame in this frame. The numbers let you specifically display the numbered frames of the image sequence.

Skilled animators sometimes draw directly in the viewer. To draw in the viewer, select the empty sheet in the *Level Strip* and open the last level there (*Col3* in our example). Then use *Level* | *New Level…* to create a new level and draw your actor directly using the toolbox. If you need more frames in the Level Strip, create them using *Level* | *Add Frames…* or directly in the Level Strip via the context menu using *Insert*.

Onion Skin Mode is an interesting aid that fades in green and red overlays in the viewer while drawing. This makes it easier to trace congruent elements between two frames and to make image changes more harmonious because they stand out quickly [5].

Defining Joints

Perhaps you have created a single image of your actor and want to change a part of it without having to redraw the figure completely. In such a case, the *Plastic Tool* will help you; you can press *X* to enable it. Then select the *Create Mesh* option in the *Tool Option Bar* above the viewer window.

A dialog with the same name appears. You can use this dialog to set the granularity of the mesh grid. Confirm by pressing *Apply*. A new column appears in the X-Sheet window with the same name and a `_mesh` suffix. This column represents a mesh layer with a wireframe model connected to the corresponding texture layer.

This connection is key to how the plastic tool works. Each column or layer connected to a mesh layer is deformed by the mesh and displayed within the mesh boundaries during rendering.

To animate a mesh plane, define a specific skeleton structure from a set of interconnected vertices. For each of the vertices, you can set various parameters, such as position, angular boundaries, and stiffness. To create the skeleton structure, select the mesh layer and enable *Build Skeleton* mode in the *Tool Option* bar.

To animate a mesh plane, first click on a point within the mesh layer that will not change under any circumstances during the subsequent animation – for example, the hip area if you want to animate the representation of a human being. Starting from this fixed point, define the directly connected joints by clicking on them. In this way, you can create a kind of tree structure. When you animate the subordinate nodes later, the underlying nodes will move along with them.

If you want to define the lower body joints, first define the position of the knee and the ankle of the left leg. Then click the hip again first, followed by the knee and ankle of the right leg. To define the joints of the upper body, click the hip again and then create the nodes for the shoulder, elbow, and wrist of the left arm. Once you're done with the left side, repeat the procedure for the right.

Once you have defined all the joints, change the mode in the *Tool Option* bar to *Animate*. After that, you can move the figure by clicking and dragging on the nodes indicated by squares – except for the hip joint, which is shown as a filled square.

If you drag the *Smart Fill Handle* of the mesh layer down in the X-Sheet, you can define the actor's poses for each frame because OpenToonz stores a keyframe with the parameters of its vertices inside the mesh layer. All you need is a drawing of your actor. However, you will also want to pull down the *Smart Fill Handle* of the connected layer beforehand.

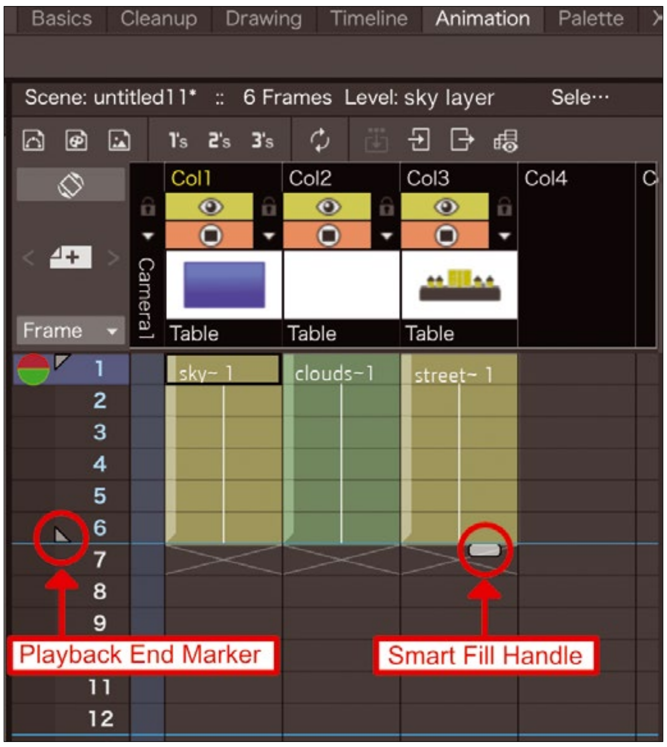


Figure 10: The X-Sheet with the repetition of frames after using the Smart Fill Handle.

Conclusions

OpenToonz is a sophisticated free animation software tool that is also used in professional projects. Accordingly, the learning curve is steep at the beginning. But OpenToonz imposes virtually no limits on your creativity. If you invest enough time and effort, you can achieve quite impressive results. ■■■

Info

[1] OpenToonz: *https://opentoonz.github.io/e/*

[2] Studio Ghibli: *https://en.wikipedia.org/wiki/Studio_Ghibli*

[3] Sample animation: *https://opentoonz.github.io/e/download/sample.html*

[4] "Dwanko Running Scene from OpenToonz Sample Data": *https://www.youtube.com/watch?v=hCui51kNvcQ*

[5] OpenToonz Manual: *https://opentoonz.readthedocs.io/en/latest/*

Author

Ralf Kirschner worked as a Visual Basic programmer in a software and system house. As a freelance system administrator, he offers computer training.

Avoiding data corruption in backups

Integrity Check

A backup policy can protect your data from malware attacks and system crashes, but first you need to ensure that you are backing up uncorrupted data. *By Rubén Llorente*

Most home users, and I dare say some system administrators, lack a backup policy. Their family pictures, music collections, and customer data files live on their hard drives and are never backed up to offline storage, only to be lost when the hard drive eventually crashes. A few users know to keep backups and regularly copy their files over to a safe storage medium. But even these conscientious people may find their strategy lacking when the time comes to recover from a system crash and they discover corrupted backup data. A successful backup strategy must involve checking for corrupted data.

Silent Data Corruption

The small number of users who keep copies of their important files only keep a single backup. Often, they use an external storage system, such as Tarsnap or a Nextcloud instance, periodically or continuously synchronizing the important files on their computers with the cloud. While a comfortable approach for end users, a single backup suffers a number of problems. Most importantly, single backups are vulnerable to silent data corruption.

Take for example a folder called `Foals`, which is full of pictures of happy young horses. My backup strategy consists of weekly copying the entire folder over to USB mass storage with a tool such as `rsync` [1]:

```
$ rsync -a --delete --checksum Foals/ ⮐
  /path/to/usb/
```

The `rsync` tool synchronizes the contents of `/path/to/usb` with the contents of `Foals`.

This strategy works until one of the pictures in `Foals` gets corrupted. Files get damaged for a number of reasons, such as a filesystem failing to recover properly after an unclean shutdown. Files also may be lost because of human error – you intend to delete `Foals/10_foal.jpg` but end up removing `Foals/01_foal.jpg` instead without realizing the mistake. If a file gets corrupted or lost and you don't detect the issue before the next backup cycle, `rsync` will overwrite the good copy in USB storage with bad data. At this point, all the good copies of the data cease to exist, destroyed by the very backup system intended to protect them.

To mitigate this threat, you can establish a long term storage policy for backups which involves saving your backup to a different folder each week within the USB mass storage. I could therefore keep a current backup of `Foals` in a folder called `Foals_2022-01-30`, an older backup in `Foals_2022-01-23`, and so on. When the backup storage becomes full, I could just delete the older folders to make room for the newer ones. With this strategy, if data corruption happens and it takes me a week to discover it, I may be able to dig up good copies of the files from an older snapshot (Figure 1). See the boxout "The rsync Time Machine" for instructions on how to set up this multi-week backup system.

Unfortunately, long term storage only works if the data corruption is discovered in time. If your storage medium only has room for storing four snapshots, a particular version of a file will only exist in the backup for four weeks. On the fifth week, the oldest snapshot will be deleted in order to make room for new copies. If the data corruption is not detected within this time window, the good copies of the data will be gone and you will no longer be able to retrieve them from a backup.

Solving for Silent Data Corruption

The first step in guaranteeing a good backup is to verify that you are backing up *only* uncorrupted data, which is easier said than done. Fortunately, a number of tools exist to help you preserve your data integrity.

Filesystems with checksum support (such as ZFS) offer a reasonable degree of protection against corruption derived from hardware errors. A checksum function takes data, such as a message or a file, and generates a string of text from it. As long as the function is passed the same data, it will generate the same string. If the data gets corrupted in the slightest, the generated string will be different.

ZFS [2], in particular, can verify if a data block is correct upon reading it. If it is not (e.g., as a result of a hard drive defect), ZFS either repairs the data block or throws an error for the user to see.

However, ZFS cannot protect data against human error: If you delete a file by accident with

```
rm Foals/01_foal.jpg
```

ZFS has no way of knowing this is a mistake instead of a legitimate operation. If a bogus image editor accidentally damages the picture using valid system calls, ZFS can not differentiate changes caused by software bugs from changes intended by the user. While ZFS is often praised as the ultimate guarantee for data integrity, its impressive capabilities fall short in my opinion.

Protection from Userspace

To verify that the data being backed up is correct, I suggest relying on userspace utilities. While many userspace programs are superb at locating damaged files, they are not easily executable from an arbitrary recovery environment. In a system crash scenario, you may find yourself using something like an obsolete SystemRescue DVD (perhaps from an old *Linux Magazine*) instead of your normal platform. In keeping with the KISS principle, you should choose userspace tools that are portable and easy to use from any platform.

If your distribution includes the GNU *coreutils* package (which the vast majority do), you need no fancy tooling.

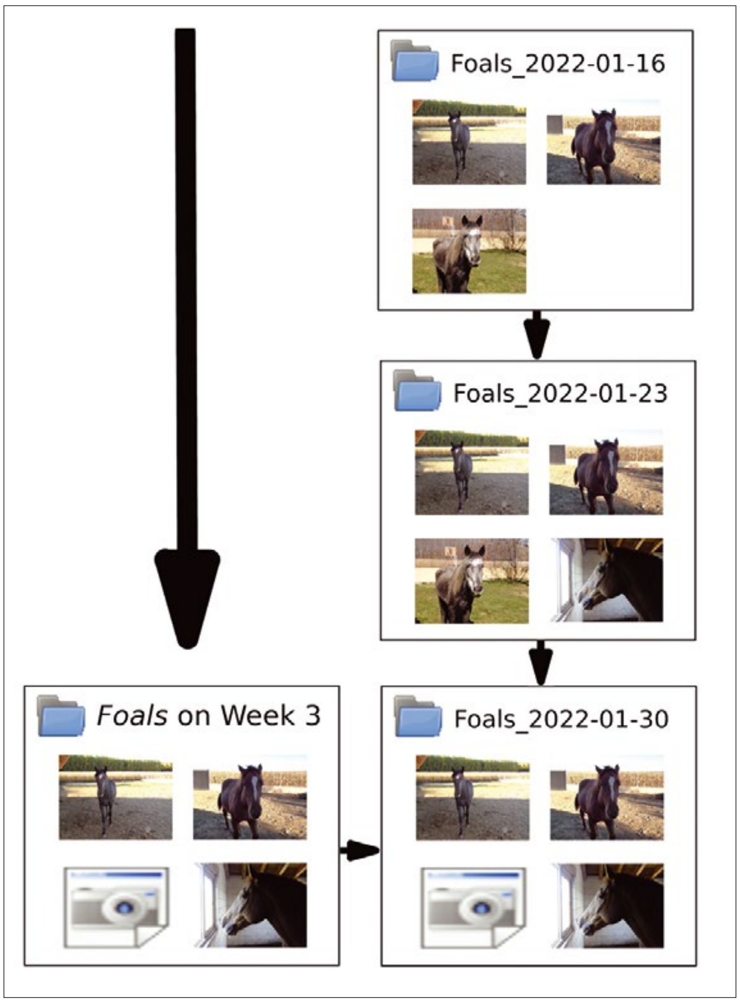


Figure 1: A damaged image in the `Foals` folder results in corrupted data in the current backup directory (`Foals_2022-01-30`). Luckily, an undamaged version can be retrieved from an earlier backup (`Foals_2022-01-23`).

The rsync Time Machine

With `rsync`, you can save backups to a directly attached drive or over a network. As an added convenience, the snapshot of the folder that `rsync` takes does not take much space on your storage device.

Suppose I have an external drive mounted under `/mnt`. The first snapshot would be saved with a regular invocation of `rsync`:

```
$ mkdir /mnt/Foals_2022-01-23
$ rsync -a Foals/ /mnt/Foals_2022-01-23
```

The first command creates a directory with a name reflecting the date. The second command copies `Foals` to the newly created directory. The `-a` switch instructs `rsync` to work in "archival" mode, recursively descending into subdirectories, preserving symlinks, time metadata, file permissions, and file ownership data.

When the time comes to make another weekly backup, I create a different backup folder (which references the new current date) and copy `Foals` to it. However, `rsync` has a trick up its sleeve: The `--link-dest` switch tells `rsync` to transfer *only* the changes since the last backup:

```
$ mkdir /mnt/Foals_2022-01-30
$ rsync -a --link-dest ⮐
  /mnt/Foals_2022-01-23 ⮐
  Foals/ /mnt/Foals_2022-01-30
```

As a result, `rsync` copies any new file to the new backup directory, alongside any file that has been modified since the last backup. Files that have been deleted from the source directory are not copied. For files that exist in the source directory but have not been modified since the last backup, `rsync` creates a hard link to the unmodified files' respective copies in the old backup directory rather than copying them to the new backup directory.

The end result is that `Foals_2022-01-23` contains a copy of `Foals` as it was on that date, while `Foals_2022-01-30` contains a current snapshot of `Foals`. Because only modified or new files are added to the storage medium, they barely take up any extra space. Everything else is included in the new backup folder via hard links.

Ideally, you should verify the files' integrity immediately before the backup is performed. The simplest way of ensuring a given file has not been modified, accidentally or otherwise, is to calculate its checksum and compare the result with the checksum it threw from a known good state (Figure 2). Thus, the first step towards protecting a given folder against corruption is by calculating the checksum of every file in the folder:

```
$ cd Foals
$ find . -type f ! -name ⮐
  '*.md5' -print0 | xargs -0 md5sum | ⮐
  sort -k 2 > md5sums_`date -I`.md5
```

(See the "Creating a Checksum" box for a more detailed explanation.)

If a week later I want to verify that the files are fine, I can issue the same command to generate a new list. Then, it would be easy to check the differences between the state of the `Foals` folder on the previous date and the state of the `Foals` folder on the current date with the following command:

```
$ diff md5sums_2022-01-23.md5 ⮐
  md5sums_2022-01-30.md5
```

The `diff` command generates a list of differences between the two files, which will make it easy to spot which files have been changed, added, or removed from `Foals` (Figure 3). If a file has been damaged, this command will expose the difference.

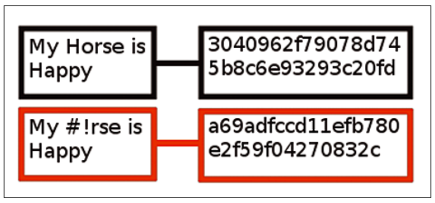

My Horse is Happy
3040962f79078d74 5b8c6e93293c20fd
My #!rse is Happy
a69adfccd11efb780 e2f59f04270832c

Figure 2: Checksums can be used to locate files that have been modified, intentionally or otherwise.

Creating a Checksum

Calculating a checksum is not intuitive, so I will break down the command and explain how it works its magic.

The `find` command locates any file (but not directories) in the current folder, excluding files with the `.md5` extension. It prints a list of the found files to the standard output. The path of each file is null terminated in order to avoid security issues (which could be derived from piping paths with special characters into the next command):

```
find . -type f ! -name '*.md5' -print0
```

Then `xargs` just accepts the list provided by the `find` command and passes it to the `md5sum` program, which generates a checksum for every entry in the list. The `-0` switch tells `xargs` that `find` is passing null-terminated paths to it:

```
xargs -0 md5sum
```

The `sort` command orders the list (because `find` is not guaranteed to deliver sorted results). The output of `md5sums` has two columns: The second column contains the path of each file; the first contains its corresponding checksum. Therefore, I pass the `-k 2` switch to `sort` in order to sort the list using the path names as a criteria:

```
sort -k 2
```

These commands create a list of all the files in the `Foals` directory, alongside its `md5` checksums, and places it under `Foals`. The file will have a name dependent on the current date (such as `md5sums_2022-01-23.md5`).

Listing 1: Newly Added Files

```
awk '{print $2}' < md5sums_2022-01-30.md5 | while read -r file; do
    if (! grep $file md5sums_2022-01-23.md5 > /dev/null); then
      echo "$file is new.";
    fi;
  done
```

Using `diff` is only practical if the dataset is small. If you are backing up several files, there are better ways to check that your data is not corrupted. For instance, you can use `grep` to list the entries that exist in the old checksum file but not in the new one. In other words: `grep` will list the files that have been modified or removed since the last time you performed a check:

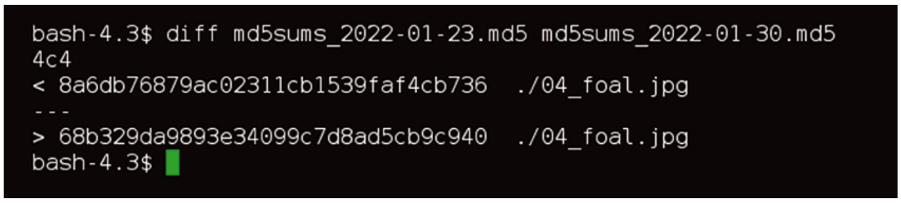
```
$ grep -Fvf md5sums_2022-01-30.md5 ⮐
  md5sums_2022-01-23.md5
```

The `-f md5sums_2022-01-30.md5` option instructs `grep` to treat every line of `md5sums_2022-01-30.md5` as a target pattern. Any line in `md5sums_2022-01-23.md5` that coincides with any of these patterns will be regarded as a match. The `-F` option forces `grep` to consider patterns as fixed, instead of as regular expressions. Therefore, for a match to be registered, it must be exact. Finally, `-v` inverts the matching: Only lines from `md5sums_2022-01-23.md5` that match no pattern will be printed.

You can also list the files that have been added since the check was last run with the shell magic in Listing 1.

With these tools, an integrity verification policy falls into place. In order to ensure you don't populate your backups with corrupted files, you must do the following:

- Generate a list of the files in the dataset and its checksums before initiating the backup.
- Verify this list against the list you generated at the last known good state.
- Identify which changes have happened between the last known good state and the current state, and check if they suggest data corruption.



```
bash-4.3$ diff md5sums_2022-01-23.md5 md5sums_2022-01-30.md5
4c4
< 8a6db76879ac02311cb1539faf4cb736  ./04_foal.jpg
---
> 68b329da9893e34099c7d8ad5cb9c940  ./04_foal.jpg
bash-4.3$
```

Figure 3: The `diff` utility will compare the checksum files, but the output will be messy if there have been multiple changes between the current and the last backups.

- If the data is good, back up your files.

A great advantage of this method is that the checksum files can be used to verify the integrity of the backups themselves. For example, if you dumped the backup to `/mnt/Foals_2022-01-23`, you could just use a command such as:

```
$ cd /mnt/Foals_2022-01-23
$ md5sum --quiet -c ⮐
  md5sums_2022-01-23.md5
```

If any file was missing from the backup or had been modified, this command would reveal the issue right away.

Alternatives

There are a number of alternatives with a long, successful history in the field of integrity verification.

AIDE, a host-based intrusion detection system, can be repurposed for verifying the integrity of the files in a given folder [3].

You can use `mtree`, a popular program from the BSD world, to verify that the contents of a directory tree match a specification. For example, `mtree` could be used to create a specification file from a folder containing known good data:

```
$ mtree -c -k md5digest ⮐
  -p Foals > /var/specification
```

Then, you can verify the contents of `Foals` against the specification file with:

```
$ mtree -f /var/specification -p Foals
```

`mtree` isn't popular in the Linux ecosystem, but you can find it in some repositories [4].

Finally, `bitrot`, a python script, can locate files damaged because of hardware defects. While it does not locate files lost because of human error or certain sorts of software error, it is very easy to set up and run despite its limited nature. If you are interested in using `bitrot`, I recommend reading Solène Rapenne's tutorial [5].

Limitations

Creating a checksum of every file within the folder you intend to backup is time consuming. For datasets that are larger than a couple of terabytes, the process may take more than half an hour.

Generating a checkum file and verifying it against the previous checksum file before each backup is only practical if the data being backed up doesn't change often. If you try this approach with a busy folder, comparing both checksum files will throw more differences than it would be reasonable to verify manually. For this reason, I recommend this method for folders which don't change often, such as directories full of family pictures or ebooks, in which files are usually added but rarely modified or removed.

Should it be necessary to verify the integrity of a folder whose contents change frequently, then I recommend using `bitrot`, because this tool only throws warnings for files whose checksum have changed but which have not suffered any changes to their modification times.

Conclusions

Having multiple backups of your data and keeping old versions of your files are great measures for preventing data loss, but they are not enough. In order for a backup strategy to work, you must be able to verify that your backup files are uncorrupted.

While many tools exist for verifying the integrity of your data, you don't need a complex solution. The *coreutils* package lets you manage moderate amounts of data.

Ultimately, discipline is the most important factor when it comes to data integrity. You need to define a routine and stick to it. This, in my experience, is where most users fail. ■■■

Info

[1] rsync: *https://rsync.samba.org*

[2] ZFS: *https://docs.freebsd.org/en/books/handbook/zfs/*

[3] "Detect evidence of break-in attempts with host-based intrusion detection systems" by Tobias Eggendorfer, *Linux Magazine*, issue 183, February 2016: *https://www.linux-magazine.com/Issues/2016/183/Host-Based-IDS*

[4] NetBSD's version of mtree: *https://repology.org/project/mtree-netbsd/versions*

[5] Solène Rapenne's bitrot tutorial: *https://dataswamp.org/~solene/2017-03-17-integrity.html*

Author

Rubén Llorente is a mechanical engineer who ensures that the IT security measures for a small clinic are both legally compliant and safe. In addition, he is an OpenBSD enthusiast and a weapons collector.

MakerSpace

Use a general purpose input/output interface on Linux computers and laptops.

Pinned

The general purpose input/output interface is not just for small-board computers anymore: You can use GPIO on your Linux desktop or laptop, too, through the USB port.

By John Schwartzman

I am impressed and intrigued by the GPIO (general purpose input/output) capability of the Raspberry Pi. The GPIO makes it easy for a Raspberry Pi user to control real-world electronic gadgets like lights and servomotors.

Wouldn't it be great if you could do the same thing with an everyday Linux PC? You can. The Future Technology Devices International Ltd. (FTDI) FT232H device [1] provides GPIO capabilities (as well as various serial protocols) to regular, non-Raspberry Pi Linux desktops and laptops through a USB port. You can buy the FT232H device from Adafruit for $15.

In a previous issue of this magazine, I developed a simple application for the Pi that used the GPIO [2]. In this article, I rewrite my Raspberry Pi application to run on Intel desktop and laptop systems with the help of the FTDI FT232H device. This simple example will give you a taste for how you can use the FT232H to bring GPIO capabilities to a standard, Intel-based Linux computer.

The Application

The application, which I wrote in C++ for portability and in the Intel x86_64 assembly language for fun [3], counts up and counts down in various number

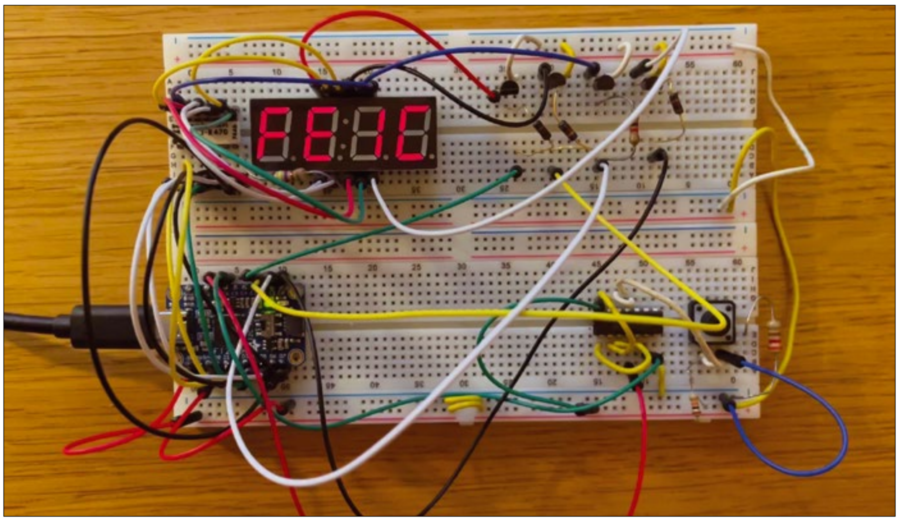


Figure 1: The breadboard and FT232H. The GPIO program is counting down in hexadecimal.

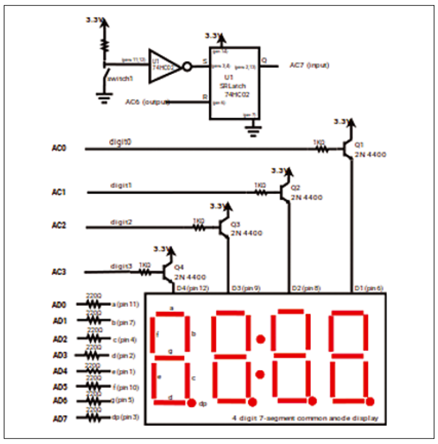


Figure 2: The wiring diagram for the breadboard and the FT232H.

systems (binary, octal, decimal, hexadecimal, and any other number system up to base 16). It also has 12- and 24-hour clocks. The output of the counts and the clocks is displayed on a four-digit seven-segment common anode display. A mode switch (switch1) allows you to change from one count or clock to another. The hardware is shown in Figure 1, and the pin-out of the hardware is shown in Figure 2. A description of the files comprising this project is provided in Table 1. See the box "Installing the Source Code and Library" for installation instructions.

Table 1: Files Used in This Project

File	Function
`gpio.asm`	x86_64 assembly language front end
`ft232h.asm`	x86_64 assembly language for interface to `libftd2xx.<version>.so`
`GPIO.hpp`	C++ front-end interface
`GPIO.cpp`	C++ front-end implementation
`FT232H.hpp`	C++ interface for interface to `libftd2xx.<version>.so`
`FT232H.cpp`	C++ implemention for interface to `libftd2xx.<version>.so`
`msl.sh`	Utility script that creates a symbolic link to a shared library
`Makefile`	Builds the source code in release or debug mode
`prepareFtdi.cpp`	C++ utility removes FTDI communication Linux-loadable kernal modules and changes ownership of latest `/dev/bus/usb/00x/00y` file to `root:dialout`
`gpio`	Executable built from `GPIO.cpp` and `FT232H.cpp`
`gpioasm`	Executable built from `gpio.asm` and `ft232h.asm`
`prepareFtdi`	Executable utility for preparing to run FT232H chip
`ftd2xx.h`	FTDI-supplied header file for `libftd2xx.<version>.so`, included in `FT232H.hpp`
`WinTypes.h`	FTDI-supplied header included in `ftd2xx.h`
`libftd2xx.<version>.so`	FTDI-supplied shared library driver for FT232H chip

The FT232H device contains two byte-sized ports, the AD bus and the AC bus, from and to which you can read and write. The program uses the AD bus to write the individual segments (*a*, *b*, *c*, *d*, *e*, *f*, *g*, and decimal point) and uses the AC bus to write the individual digits (`digit0`, `digit1`, `digit2`, and `digit3`). The segments for the four digits are tied together inside the display device. Each number of the 16-number symbols (0, 1, 2, 3, 4, 5, 6, 7, 8, 9, A, b,

Installing the Source Code and Library

To begin, open a browser and navigate to *https://ftdichip.com/drivers/d2xx-drivers/*, click on *1.4.24* (select a different architecture if you don't have an Intel processor), and extract `libftd2xx-x86_64_1.4.24` (or whatever your version is named) into `~/Downloads`. (I used 1.4.24 ARMv8 hard-float for building the Raspberry Pi 4 version.)

Open a terminal on your computer and enter:

```
$ sudo apt update
$ sudo apt install libboost-all-dev
$ sudo apt install make
$ sudo apt install gcc
$ sudo apt install mlocate
```

You need `libboost-all-dev` for the `prepareFtdi` executable. Now, find the path where the `boost_filesystem` library was installed. You should see something like the output below:

```
$ locate libboost_filesystem.so
/usr/lib/x86_64-linux-gnu/libboost_filesystem.so
/usr/lib/x86_64-linux-gnu/libboost_filesystem.so.1.71.0
```

In this case, you know to put the FTDI library in `/usr/lib/x86_64-linux-gnu/`.

Next, copy files, change permissions, and copy `ft232h.zip` into `~/Development/`

```
$ sudo cp ~/Downloads/release/build/libftd2xx.so.1.4.24 /
usr/lib/x86_64-linux-gnu/
$ sudo chmod 0775 /usr/lib/x86_64-linux-gnu/libftd2xx.
  so.1.4.24
$ cd
$ mkdir Development
$ cd Development
$ unzip ft232h.zip
$ cd FT232H
$ cp ~/Downloads/release/ftd2xx.h .
$ cp ~/Downloads/release/WinTypes.h .
```

Now plug in the FT232H's USB cable. Then, create a symbolic link to the FTDI library and make the executable files once before running the executable utility that prepares the FT232H chip:

```
$ sudo ./msl.sh /usr/lib/x86_64-linux-gnu/libftd2xx.so.1.4.24
$ cd ~/Development/FT232H
$ make
$ sudo ./prepareFtdi
```

(See the "Running on a VM" box if you are using a virtual machine.) Now you can run `./gpio` (C++ version) or `./gpioasm` (ASM version).

Pressing switch1 cycles through the operating modes. If you want to exit before getting through all of the modes, press Ctrl+C.

Running on a VM

If you are working on a virtual guest machine rather than a single computer, the FT232H may show up on the guest computer or on the host, in which case you need to move it to the guest.

To determine where the FT232H is, plug it in and type `lsusb` in a terminal to see all of the USB devices. If you do not see a line containing *Future Technology Devices International, LTD FT232H Single HS USB-UART/FIFO IC*, then your FT232H is currently on the host.

On VMware, open *VM | Removable Devices | Future Devices USB Serial Converter | Connect (Disconnect from Host)*. If you are not using VMware, do the equivalent on your virtualization software. The `lsusb` command should now show the FT232H line:

```
$ lsusb
...
Bus 00x Device 00x: ID xxxx:xxxx
Future Technology Devices ...
...
```

After preparing the FT232H chip with `sudo ./prepareFtdi`, you can now run `./gpio` (C++ version) or `./gpioasm` (ASM version).

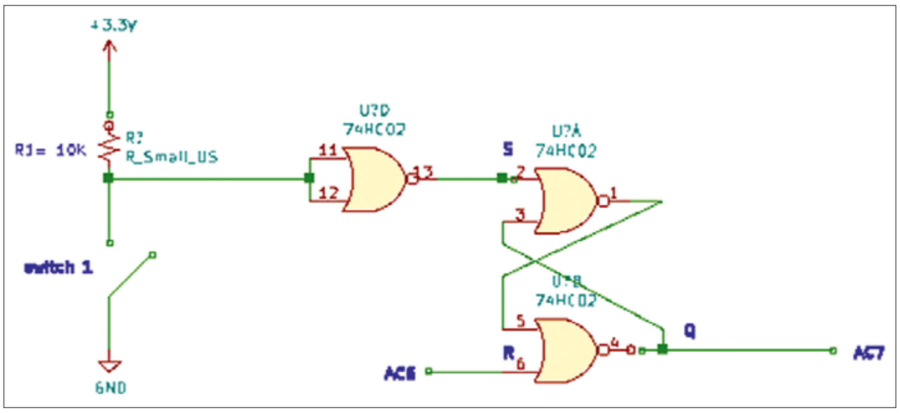


Figure 3: A schematic of U1, the 74HC02, switch1, and the connections to FT232H.

C, d, E, F) comprises a combination of segments. For example, to write the number 1, you must turn on segments *b* and *c* and turn off the other six segments. The number 7 is like the number 1, except the *a* segment is also lit. Notice that the symbol A corresponds to the decimal value 10, and the symbol b corresponds to the decimal value 11 (C = 12, d = 13, E = 14, F = 15).

In a common anode display, you write a 0 bit (0V) to turn a segment on and a 1 bit (3.3V) to turn a segment off. I know that sounds backwards, but because devices can usually sink more current than they can source, the common anode display is a good choice for this application. For the digit drivers, you need more current than can be supplied by a single device pin, so you use a transistor to turn the digits on and off. The output pins can supply about 16mA maximum. To drive a digit with eight lit segments takes about 96mA. The transistors being used here take about 5mA of base current to supply up to 100mA of collector current.

To display numbers, you write each segment in turn and briefly turn on the driver bit for each digit, switching from digit to digit so quickly that the eye can't detect that the digits are turning on and off. Displaying the digits takes 12 output bits from the GPIO. The eight segments take all 8 bits of the AD bus, and the four digits take 4 bits of the AC bus. That leaves 4 bits on the AC bus to use as needed. AC7 is designated an input and AC6 an output. AC7 handles detecting a press of switch1 to interrupt a count or clock method. AC6 is used to reset an interrupt request by bringing AC7 low.

Momentary contact switches like switch1, as well as being noisy, produce pulses of uncertain duration. You need to get rid of the noise and make the pulse width wide enough for the application to detect, which can be accomplished with an S-R latch (a sort of flip-flop), shown as U1 in Figure 2. The 74HC02 integrated circuit contains four individual two-input NOR gates in a 14-pin dual inline package (DIP) (Figure 3.)

Connecting two of the NOR gates together acts as an S-R (set-reset) latch, and the third NOR gate has its inputs tied together so that it acts as an inverter (a NOT gate). They are connected to switch1 and to pins 6 and 7 of the AC bus (Figures 2 and 3). Why 74HC02 instead of 7402? Because you want to be able to use either 3.3V or 5V. The FT232H gets 5V from the USB port and provides a 3.3V output. Unlike the Raspberry Pi, the FT232H is 5V tolerant: You can use 5V instead of 3.3V if you prefer.

When the user presses switch1, the switch signal is inverted and sent to the

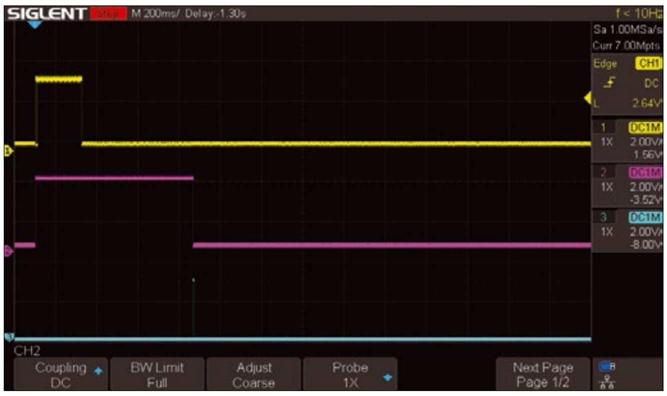


Figure 4: The reset waveforms for the device with a single Set pulse.

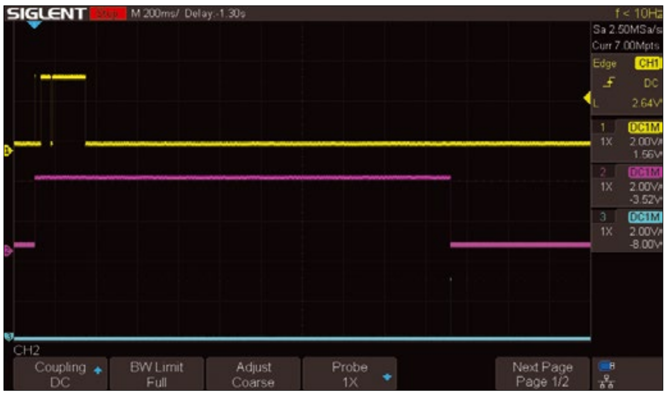


Figure 5: The reset waveforms for the device with a noisy Set pulse.

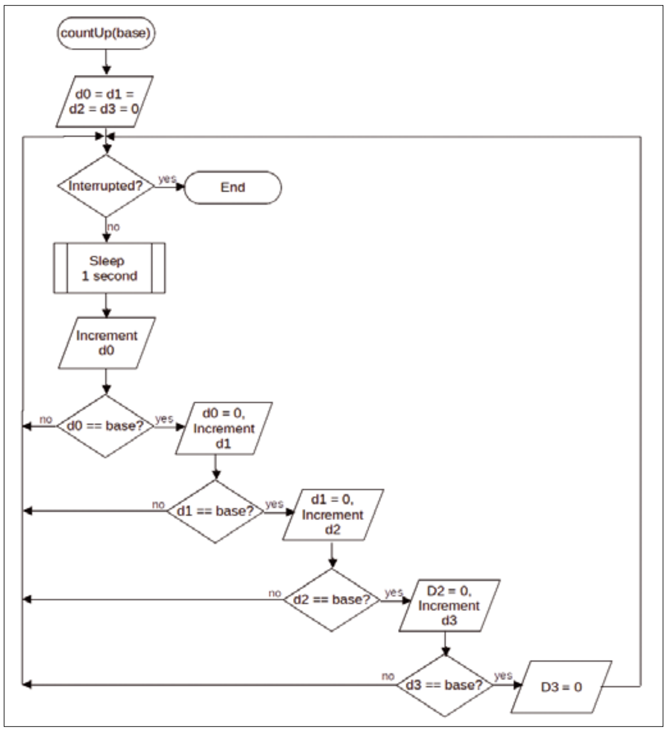


Figure 6: A flowchart for the countUp(base) subroutine (GPIO.cpp line 69).

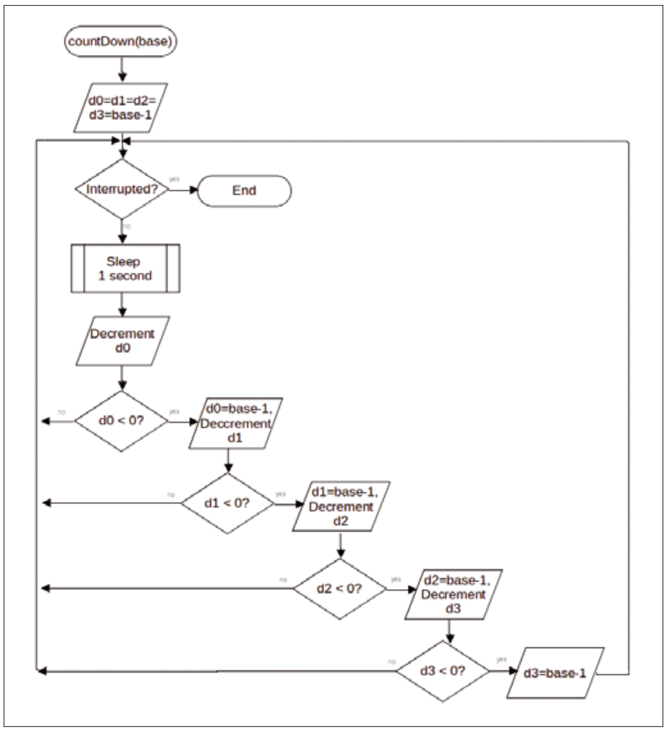


Figure 7: A flowchart for the countDown(base) subroutine (GPIO.cpp line 97).

S input of the latch, which causes the S input to momentarily go high and, in turn, causes the Q output to go high and remain high until it is reset by a high-low pulse to the R latch input. GPIO pin AC7 serves as a GPIO input to the program. The program's `readWrite()` thread method periodically reads the AC byte of the FT232H. If it finds AC7 high, it knows that an interrupt has occurred and sets `bInterruptFlag` high to tell the currently running clock or count method to exit and return to `main()`.

The program then sends a high-low pulse to the AC6 GPIO output pin, which is connected to the R input of the S-R latch. This pulse causes the latch to reset and bring the Q output (AC7 input) low. Note that the switch can be noisy. When it is pressed, it can produce a single pulse (Figure 4) or multiple pulses (Figure 5). The S-R latch doesn't care. Q goes high on the first falling edge of the switch pulse. It remains high, despite what the S input is doing, until the R input goes high.

The main program has subroutines that count or tell time, and they don't know about or care how numbers are displayed. Their purpose is simply to fill `digit0`, `digit1`, `digit2`, and `digit3` with numbers. The `countUp(base)` method (Figure 6) starts by setting all four digits to `0` and then incrementing the digits every second according to the rules of the number system (base) that it is told to use. The `countDown(base)` method (Figure 7) starts by setting all four digits to base 1 and then decrements the digits every second according to the rules of the number system it is told to use.

The `runClock(hours)` method simply sets the four digits to the hours and minutes of the current time, which it gets from Linux. If the hours parameter is `24`, `runClock` displays the hours exactly as returned by Linux. If the hours parameter is 12, `runClock` subtracts 12 from `hours` that are greater than 12.

readWrite Subroutine

The other main subroutine is called `readWrite()`, and it runs in a separate thread. It doesn't care how the four digits were obtained. It simply gets the contents of each of the digits and writes them to the four digits of the seven-segment display. The `runClock()` method tells `readWrite()` to display a blinking colon between the hours and the minutes of the clock.

The `readWrite()` thread method handles all matters involving input and output to the hardware, which means it must write to port AD the segments necessary to display a number and use port AC to power on each digit in turn. (See Figure 8 for the waveforms associated with the `readWrite` method.) It must strobe (write high-low pulses to) each of the digit lines in turn, fast enough to appear to be one four-digit number instead of four separate one-digit numbers that appear on the display at different times.

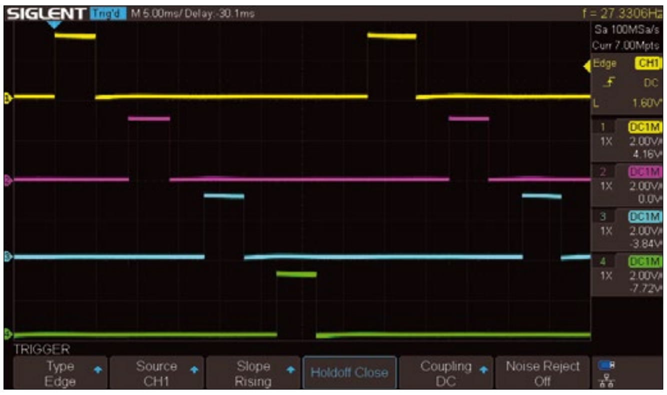


Figure 8: The oscilloscope trace indicates the waveforms produced by gpio.cpp. Yellow = AC0, violet = AC1, blue = AC2 and green = AC3.

You can see that the program turns off digit3, writes the segment data to digit3, and then turns on digit3. After waiting a small amount of time, it then repeats this process for digit2, digit1, and digit0. User input is a slow process, so you don't check for an interrupt request on each iteration of the readWrite loop. Instead, you check for an interrupt request every 16 iterations. In this way, you reduce flicker in the display. (You can try varying the DELAY_COUNT constant to see what value works best on your hardware. Six iterations seemed to work well on the Raspberry Pi 4.)

The Raspberry Pi allows you to use designated GPIO pins as Linux kernel interrupts. All interrupt activity can be coordinated by a loadable kernel module (LKM). The FT232H does not have this capability. The readWrite() thread is responsible for polling for an interrupt. It must handle the interrupt, reset the S-R latch, and notify the main thread of execution that the currently running method should end and return to main(). The main program simply launches a series of subroutines in an order determined by the programmer. It doesn't know or care that its subroutines must poll bInterruptFlag approximately once a second to decide whether to keep running or to return to main().

The main() routine also creates a signal handler, so it can handle a Ctrl+C interrupt. The signal handler subroutine simply writes a 1 to bExitFlag. While the main() subroutines are monitoring for bInterruptFlag, they are also monitoring for bExitFlag and will return if they receive either one. The main() method decides whether to start the next subroutine (bInterruptFlag == true) or return to Linux (bExitFlag == true).

The FC232H is not exactly plug and play. First, it requires you to solder two 11-pin headers to the device before the first use. Second, when it is plugged in to a USB port, it starts a built-in LKM, ftdi_sio, which it uses for its sundry communications functions. The ftdi_sio chipset serial I/O application uses a library that is not compatible with the library needed for GPIO. The proprietary library that is used (libftd2xx.so.1.4.24) requires you to remove the LKMs ftdi_sio and usbserial from the kernel and to open a character device named /dev/bus/usb/<xxx>/<yyy>, which is created and named at USB plug-in time.

If you're prepared to always run as root, you don't need to know the name of this character device. If you're not, then run the utility program prepareFtdi as root after plugging in the FT232H. It will find the latest of the /dev/bus/usb/<qrs>/<xyz> character devices and change its ownership from *root:root* to *root:dialout*. All you need to do is to add yourself to the *dialout* group

```
sudo usermod -a -G dialout <userID>
```

before you first use the FT232H.

FT232H Class

The source code in FT232H.hpp and FT232H.cpp define a class, FT232H, that is a slim wrapper around the *libftd2xx* library's GPIO functions. It has a constructor FT232H::FT232H() that initializes the FT232H and puts it into MPSSE mode, which is needed for access to the *libftd2xx*'s GPIO functions. FT232H has a destructor ~FT232H::FT232H() that cleans up and closes the FT232H device.

The other functions primarily involve writing to and reading from the two GPIO ports AC and AD. The line with FT232H::writeACByte(byte byteToWrite) writes a single byte to the AC bus, and FT232H::readACByte(byte* byteToRead) reads a single byte from the AC bus and places it in byteToRead. The writeACByte() class method requires a single call to the FT_Write() library function, and readACByte() calls both FT_Write() and FT_Read(), in that order.

The FT232H class encapsulates items like FT_HANDLE, FT_STATUS, ACBusDirection, and ADBusDirection, all private to and hidden away inside the class. You can ignore them in calling your class methods, knowing that the FT232H class will take care of them internally.

A class is an abstraction mechanism that presents you with a menu of functions over its interface, FT232H.hpp. All of the ugly details, like FT_HANDLE, which must be created when initializing the class and must be sent to every single library call, are hidden away inside the implementation file of the class, FT232H.cpp. This is called object-oriented programming, where FT232H is a first-class object with public methods and functions. Class data and private methods very specific to the class are inaccessible from the other parts of the program.

Every FT_ library operation returns either success, FT_OK == 0, or some other enumerated code that describes the error. Common practice is to follow class construction with a request for that enumerated code so that you know whether to continue or to exit:

```
FT232H*   pFT    = new FT232H();
FT_STATUS status = pFT->getStatus();
if (status != 0)
   {
    return status;
   }
```

The file ft232h.asm is the assembly language equivalent of the C++ FT232H class. It provides the same functionality but has no neat mechanism for hiding things from public view. All data is global in an assembly language program. C++ is portable, so you can take your program from one Linux desktop with an Intel processor and run it on another Linux laptop with an ARM processor because both machines have a C++ compiler that turns C++ into the correct machine language for its processor. If you want to run the assembly language program on another processor, you're out of luck. You would have to port the program into an entirely different assembly language.

C++ is easy to understand, by humans and by machines, and is portable from machine to machine, whereas assembly language is confusing to humans at first sight and is not portable, so why do I like it? When you have a program that talks to hardware, like this one, the assembly language shows you exactly what's happening step-by-step at the hardware level. You can also shave a few cycles from an assembly language program and make it run slightly faster than the C++ program in the places where you need speed. (This program doesn't need speed; it spends most of the time sleeping.)

You can compare the C++ and assembly language code side by side and be sure that you really understand exactly what the code is accomplishing. Your C++ compiler translates C++ into assembly language and then the assembly language into machine code in an object

Listing 1: Write to GPIO

```
void* readWrite(void* pArg) // thread function - writes digits to GPIO
{
  GPIO*   pGPIO = GPIO::getGPIO();
  FT232H* pFT   = FT232H::getFT232H();
  int     nCnt  = DELAY_COUNT;

  while (!pGPIO->getExitFlag())
  {
        if (--nCnt == 0)    // is it time to check for an interrupt?
        {
            if (pGPIO->interrupted()) // yes - check for interrupt
                {
                pGPIO->setInterruptFlag(true);  // respond to interrupt
                pGPIO->resetInterrupt();
                }
            nCnt = DELAY_COUNT;    // repopulate nCnt
        }

    pFT->writeACByte(0); // clear digits - turn off display
    pFT->writeADByte(pGPIO->getPattern(3)); // write digit3 segments
    pFT->writeACByte(DRIVE_DIGIT_3);        // turn on digit3
    sleep2_5ms();
    ...
  }
...
}
```

Listing 2: GPIO.cpp Snippet

```
bool GPIO::interrupted()  // check for interrupt (GPIO.cpp line 55)
{
    byte byteToRead
    FT232H::getFT232H()->readACByte(&byteToRead);    // readACByte into byteToRead
    return (byteToRead & INTERRUPT_MASK) != 0;
}
```

Listing 3: gpio.asm Snippet

```
interrupted:    ; check for interrupt (gpio.asm line 343)
  prologue
  sub   rsp, VAR_SIZE * NUM_VAR ; make space for local var on stack

  lea rdi, byte nParam          ; pass addr of nParam to readACByte
  call  readACByte              ; read a byte from AC into nParam
  mov al, byte nParam           ; get the byte we just read
  test  al, INTERRUPT_MASK      ; is the INTERRUPT_MASK bit set?
  setnz al                      ; al = 1 if yes, al = 0 if no
.fin:
    epilogue
```

file. Although you usually wouldn't look at the assembly code produced by the compiler, you can if you add the `-S` flag to the call to the compiler.

GPIO Class

The `GPIO` class (`GPIO.cpp` and `GPIO.hpp`) does the work of the program. Its member functions count up in a variety of number systems, count down in a variety of number systems, and display 12- and 24-hour clocks. The `main()` function, which is not a member of the `GPIO` class, instantiates `FT232H` and `GPIO` objects and starts a separate thread of execution to run the `readWrite()` method.

As stated before, `GPIO` counts by using its member methods: `countUp(base)`, `countDown(base)`, and `runClock(hours)`. The `GPIO` non-member thread method `readWrite()` employs the `FT232H` object to write the counts and time to the FT232H GPIO ports. `GPIO` also contains some member helper methods for `readWrite()`: `bool GPIO::interrupted()`, `void GPIO::resetInterrupt()`, and `byte GPIO::getPattern(int digit)`.

Non-member functions implement time delays. The `FT232H` and `GPIO` classes have static helper functions, `FT232H::getFT232H()` and `GPIO::getGPIO()` that return pointers to each class. Wherever you are in the program, you can call these functions and access each classes' members. For example, `GPIO.cpp` has the code shown in Listing 1. Note that the static functions `FT232H::getFT232H()` and `GPIO::GetGPIO()` are called by a fully declared name and not by a preexisting object. You use the pointers returned by these static functions to access the member methods of the class. The file `gpio.asm` is the assembly language equivalent of `GPIO.hpp` and `GPIO.cpp`.

Now look at a C++ member function (Listing 2) next to its assembly language equivalent (Listing 3). The C++ version has a member function of the `GPIO` class that takes no arguments and returns a boolean value. It declares a local (or automatic) variable, `byte byteToRead`, that lives on the stack.

The static member of the `FT232H` class, `FT232H::getFT232H()`, gets a pointer that you can use to call `FT232H::readACByte()` and then pass it the address of `byteToRead`. The question is: When you AND `byteToRead` with `INTERRUPT_MASK` (0x80), is the result not equal to zero? Remember that

AC7 is an input pin that gets the Q output pin of the S-R latch. The answer to this question is returned to the calling method.

gpio.asm

Now, it's a little hard to believe, but the assembly language function does exactly the same thing, starting with the macro definition, `prologue`. All Intel assembly language functions that call other subroutines, need

```
%macro prologue 0
  push  rbp
  mov   rbp, rsp
%endmacro
```

to prepare the stack. (The `epilogue` macro at the end of the function,

```
%macro epilogue 0
  leave
  ret
%endmacro
```

restores the stack to the way it was before `prologue` was invoked.) After invoking `prologue`, the line

```
  sub   rsp, VAR_SIZE * NUM_VAR
```

subtracts the product (8x2 = 16 because `NUM_VAR` is rounded up to an even number) from the stack pointer to make room on the stack for the local 8-bit variable `nParam`. Next, the code

```
  lea   rdi, byte nParam
readACByte
  call  readACByte nParam
  mov   al, byte nParam
  test  al, INTERRUPT_MASK ⮐
    ; is the INTERRUPT_MASK bit set?
  setnz al  ⮐
    ;   al = 1 if yes, al = 0 if no
  epilogue
```

loads the effective address `lea` of `byte nParam` into the `rdi` register. In the C calling convention, the first argument to a function is always passed in the `rdi` register.

The lines that follow call `readACByte`, get the byte just read into the 1-byte `al` register, and ANDs `al` with `INTERRUPT_MASK` to clear the processor's zero flag if bit 7 of the byte read is a <1>1<1>. The `setnz al` instruction sets the `al` register to 1 if the zero flag is cleared, or it sets `al` to `0` if the zero flag is set. By the C calling convention, you always return the result in the `rax` register (of which `al` is the least significant byte). After invoking the `ret` (in `epilogue`), the `al` register will contain 1 if there was an interrupt, or `0` otherwise.

The calling function (in this case, `readWrite`) can then invoke `test al, al` to find out whether the interrupted function found an interrupt. The `test` instruction does an AND operation, which affects the flags but doesn't save the result. The relevant piece of the calling function is:

```
readWrite:
    ...
  call  interrupted   ⮐
    ; check for interrupt
  test  al, al      ; interrupted?
  jz   .continue0   ; jump if no

  call resetInterrupt   ⮐
    ; handle interrupt
.continue0:
    ...
```

Why are label names preceded by a period? It's a YASM and NASM assembler trick that lets you reuse label names from method to method. The label `.continue0` is really known to the assembler as `readWrite.continue0`. Assembly language contains lots and lots of jumps (equivalent to the dreaded `goto` statement). To avoid calling the `resetInterrupt` method, `gpio.asm` performs `jz .continue0` to jump around the call to `resetInterrupt` if `al == 0`. Every conditional (`jz .next`, i.e., jump if zero flag is *1*) or unconditional (`jmp .next`) jump needs a target label. Reuse of label names is a real time and energy saver. ■■■

Info

[1] FT232H : *https://www.adafruit.com/product/2264*

[2] "Access Raspberry Pi GPIO with ARM64 assembly" by John Schwartzman, *Linux Magazine*, issue 247, June 2021, pg. 56, *https://www.linuxpromagazine.com/Issues/2021/247/ARM64-Assembly-and-GPIO*

[3] Code for this article: *ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/258/*

Author

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

MakerSpace

Industrial network communications

OPC UA on a Raspberry Pi

Use Python and Node-RED to create an industrial client-server test system. By Pete Metcalfe

Industrial operations such as chemical refineries, power plants, and mineral processing operations have different network communications requirements than most IT installations. Some of the key industrial communication requirements include security, multivendor connectivity, time stamping of data, and alarm or quality indicators.

To meet industrial requirements, a communications standard called OLE for Process Control (OPC) [1] was created. The original OPC design was based on Microsoft's Object Linking and Embedding (OLE) technology, and it quickly became the standard for communications between control system consoles, historians (data storage devices), and third-party applications (Figure 1).

The original OPC standard worked well, but it had major limitations in the areas of Linux and embedded systems, routing across wide-area networks (WANs), and new security concerns. To better address new industrial requirements, the Open Platform Communications Unified Architecture (OPC UA) standard [2] was created.

In this article, I look at three small projects that introduce you to OPC UA. The first project uses Python to create an OPC UA server with a client and a graphical user interface (GUI). The second project builds a Node-RED OPC UA server and client app. The final project uses Node-RED web dashboards for read-write access to a Python OPC UA server.

Author

You can investigate more neat projects by Pete Metcalfe and his daughters at *https://funprojects.blog*.

Getting Started

For my test system, I used a Raspberry Pi 4 and an Ubuntu laptop, but the test could also be done on a single Raspberry Pi or laptop. You have a number of OPC UA open source servers from which to choose. For C development applications, the open62541 project [3] offers a C99 architecture that runs on Windows, Linux, VxWorks, QNX (BlackBerry), Android, and a number of embedded systems.

For lightweight, quick testing, OPC UA servers are available in Python and Node-RED. The Free OPC-UA Library project [4] has a great selection of open source tools for people who want to learn and play with OPC UA.

To keep things simple, I am using the *python-opcua* library, which is a pure Python OPC UA server and client. (Note that a more complete Python OPC UA library, *opcua-asyncio*, is available for more detailed work.)

The Free OPC UA project also includes a number of useful tools, such as an OPC UA modeler and client GUI. The commands

```
# Python OPC UA server/client install
sudo apt install python-opcua
# OPC UA client/QT dependencies
sudo apt install PyQT*
pip3 install opcua-client
# OPC UA modeler tool
$ pip3 install opcua-modeler
```

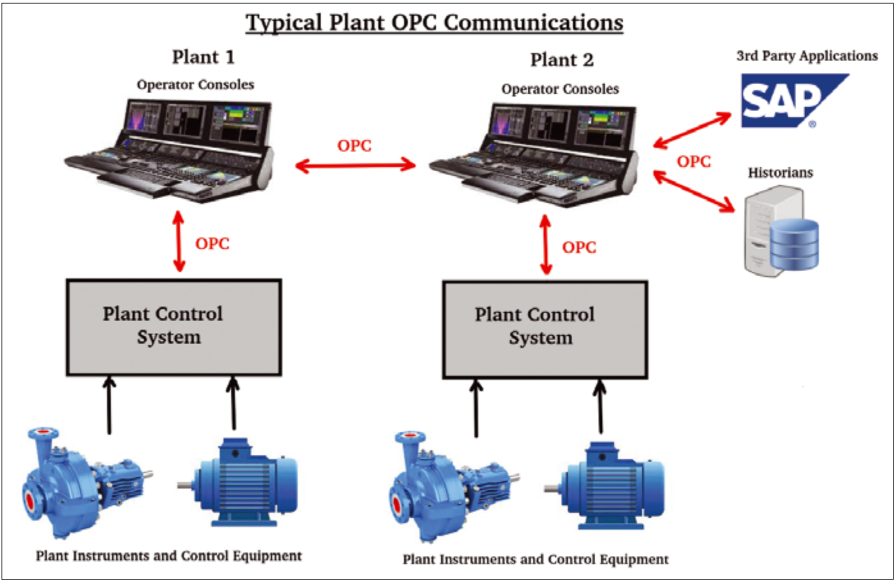
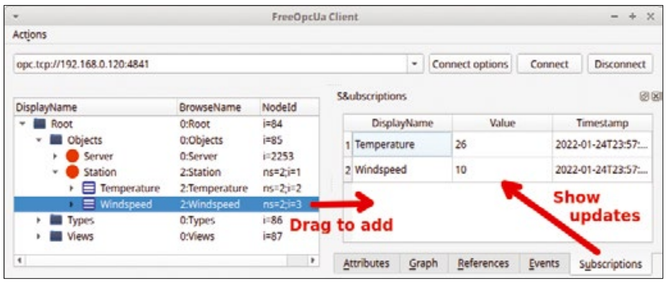


Figure 1: Typical plant OPC architecture.

Listing 1: Python OPC UA Simple Server

```
01 # opcua_server1.py - Create an OPC UA server and simulate
   2 tags
02 #
03 import opcua
04 import random
05 import time
06
07 s = opcua.Server()
08 s.set_server_name("OpcUa Test Server")
09 s.set_endpoint("opc.tcp://192.168.0.120:4841")
10
11 # Register the OPC UA namespace
12 idx = s.register_namespace("http://192.168.0.120:4841")
13 # start the OPC UA server (no tags at this point)
14 s.start()
15
16 objects = s.get_objects_node()
17 # Define a Weather Station object with some tags
18 myobject = objects.add_object(idx, "Station")
19
20 # Add a Temperature tag with a value and range
21 myvar1 = myobject.add_variable(idx, "Temperature", 25)
22 myvar1.set_writable(writable=True)
23
24 # Add a Windspeed tag with a value and range
25 myvar2 = myobject.add_variable(idx, "Windspeed", 11,4,4)
26 myvar2.set_writable(writable=True)
27
28 # Cycle every 5 seconds with simulated data
29 while True:
30     myvar1.set_value(random.randrange(25, 29))
31     myvar2.set_value(random.randrange(10, 20))
32     time.sleep(5)
```


FreeOpcUa Client

opc.tcp://192.168.0.120:4841

DisplayName	BrowseName	NodeId
Root	0:Root	i=84
Objects	0:Objects	i=85
Server	0:Server	i=2253
Station	2:Station	ns=2;i=1
Temperature	2:Temperature	ns=2;i=2
Windspeed	2:Windspeed	ns=2;i=3
Types	0:Types	i=86
Views	0:Views	i=87

S&ubscriptions

	DisplayName	Value	Timestamp
1	Temperature	26	2022-01-24T23:57:...
2	Windspeed	10	2022-01-24T23:57:...

Figure 2: Free OPC UA client tool.

Listing 2: Getting a Tag Value

```
>>> import opcua
>>>
>>> # Connect as an OPC UA client and get a tag value
>>> client = opcua.Client("opc.tcp://192.168.0.120:4841")
>>> client.connect()
>>> temptag = client.get_node("ns=2;i=2")
>>> temptag.get_value()
26
```

load these items in Ubuntu/Debian/Raspian environments.

Simple Python OPC UA Server

As a first project, I create a simple OPC UA server with some simulated OPC UA tags (Listing 1) [5]. The first steps in setting up an OPC UA server are to define a server name, set a network location endpoint, and register the namespace (lines 7-12). A little later you will see that the namespace index is very important in the definition of OPC UA tags. To simplify my code, I hard-coded the IP address (lines 9 and 12). For a more flexible setup, the Python socket library could be used to determine this value for the endpoint and namespace.

The OPC UA structure is based on objects and files, under which tags are configured. For this example a `Station` object is created (line 18), with `Temperature` and `Windspeed` tags (lines 21-26). The final step is to use the `set_value` method to update the tags with random values (lines 30-31).

The Free OPC UA project's *opcua-client* GUI tool lets you view the OPC UA server and its tags. Figure 2 shows the server with two simulated tags and their NodeIDs.

OPC UA items are accessed by their NodeIDs. The OPC UA server will autocreate a NodeID if one isn't defined when an object or tag is added. In this example, `Temperature` is given the NodeID `ns=2,i=2`, which means it has an index of 2 in node space 2. In the next sections, I show how NodeIDs are used and how they can be created.

In OPC UA, the terms "tags" and "variables" are often used interchangeably. In an industrial plant, hardware signals such as pumps and sensors are usually referred to as "tags" in the control systems; however, within the OPC UA server, the term "variable" is used. The key difference is that a variable can also be an internal or software-generated point, such as a counter.

Python OPC UA Client App

The *opcua* library is used for both server and client applications. An OPC UA client app only requires five lines to connect to a server and define a NodeID object and get its value. Listing 2 shows how to get the previously defined `Temperature` tag.

The `Client` method defines the OPC UA server's endpoint location, and the `connect` method enables the client connection. An OPC UA NodeID object is defined by a namespace number (`ns`) and an index (`i`), and the `get_value` method returns the current value. For my Python client application, I use this basic code to show tag values on a gauge chart (Figure 3).

The Python *tk_tools* [6] library contains some useful graphic widgets (e.g., charts, gauges, LEDs, seven-segment displays). To install, enter:

```
pip install tk_tools
```

The client app (Listing 3) creates a Tkinter object (line 11) and then adds two gauge components (lines 15-20). The

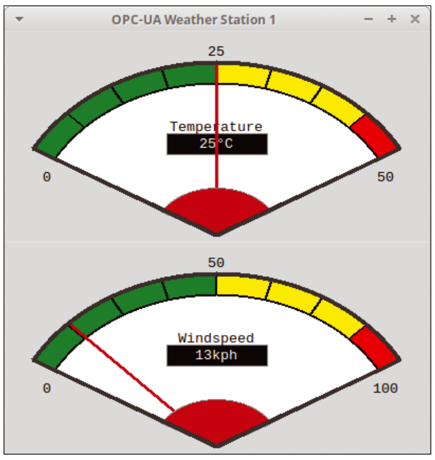


Figure 3: Python OPC UA client app.

Tkinter `after(<msec>,<function_name>)` method calls an update function (line 26) that refreshes the gauge components with OPC UA tag updates (lines 26 and 28).

XML Databases

In the earlier Python OPC UA server example, variables were added dynamically when the server was started. This method works fine for simple testing, but it can be awkward for larger tag databases.

All industrial control vendors will have proprietary solutions to create OPC UA XML configuration files from process control logic, piping and instrument drawings (P&IDs), or instrument lists.

OPC UA XML configuration files follow a defined standard, so an XML file exported from a Python server can be imported into a Node-RED server or another standard OPC UA server.

The Python *opcua* library syntax to import and export XML files is:

```
# to export from the online system ↩
  to an XML file:
# where: s = opcua.Server()
s.export_xml_by_ns("mytags.xml")
# to import an XML file:
s.import_xml("mytags2.xml","/mypath")
```

The XML files can be viewed in a web browser; unfortunately, the format is a little ugly (Figure 4). The file starts with a header area, followed by object (`<UAObject>`) and variable definitions (`<UAVariable>`). The variable section defines the variable's NodeID, name, and description.

The Free OPC UA Modeler tool can help with the creation and modification of XML databases. With `opcua-modeler`, you can insert new variables, objects, or properties into a new or existing XML structure. In Figure 5, I add a variable called *Waveheight*, which I save to the XML file. In the final project, I use this XML file with Node-RED dashboards.

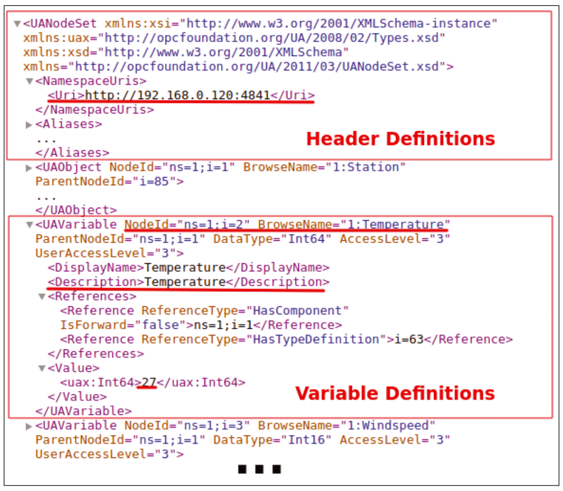
Although the modeler tool works great for manually adding or deleting a few variables (Figure 5), it can be awkward when a hundred variables need to be added.

CSV to XML

A CSV file is an easy format for defining tag databases. For example, a `mytags.csv` file could be defined with three fields; `tagname`, `description`, and `default value`:

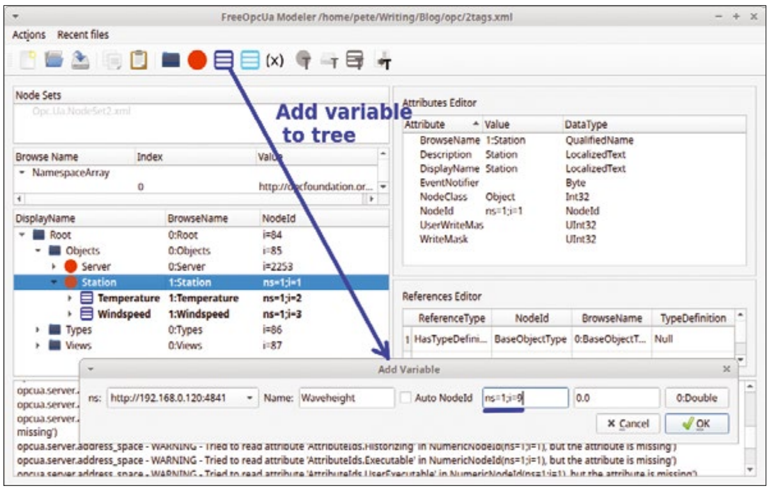
```
# mytags.csv - simple tag database
# format: tagname, description, ↩
  default value
#
TI-101,temperature at river, 25
PI-101,pressure at river, 14
```

A basic CSV to XML import tool can be created to meet your project requirements. A number of good programming options can help you with this migration. For my project, I created a small



```
▼<UANodeSet xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xmlns:uax="http://opcfoundation.org/UA/2008/02/Types.xsd"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns="http://opcfoundation.org/UA/2011/03/UANodeSet.xsd">
  ▼<NamespaceUris>
     <Uri>http://192.168.0.120:4841</Uri>
   </NamespaceUris>
  ▶<Aliases>
   ...
   </Aliases>
  ▶<UAObject NodeId="ns=1;i=1" BrowseName="1:Station"
   ParentNodeId="i=85">
   ...
   </UAObject>
  ▼<UAVariable NodeId="ns=1;i=2" BrowseName="1:Temperature"
   ParentNodeId="ns=1;i=1" DataType="Int64" AccessLevel="3"
   UserAccessLevel="3">
     <DisplayName>Temperature</DisplayName>
     <Description>Temperature</Description>
    ▼<References>
       <Reference ReferenceType="HasComponent"
       IsForward="false">ns=1;i=1</Reference>
       <Reference ReferenceType="HasTypeDefinition">i=63</Reference>
     </References>
    ▼<Value>
       <uax:Int64>27</uax:Int64>
     </Value>
   </UAVariable>
  ▶<UAVariable NodeId="ns=1;i=3" BrowseName="1:Windspeed"
   ParentNodeId="ns=1;i=1" DataType="Int16" AccessLevel="3"
   UserAccessLevel="3">
                         ■ ■ ■
```

Figure 4: OPC UA XML database.



Browse Name	Index	Value
NamespaceArray		
	0	http://opcfoundation.or...

DisplayName	BrowseName	NodeId
Root	0:Root	i=84
Objects	0:Objects	i=85
Server	0:Server	i=2253
Station	1:Station	ns=1;i=1
Temperature	1:Temperature	ns=1;i=2
Windspeed	1:Windspeed	ns=1;i=3
Types	0:Types	i=86
Views	0:Views	i=87

Attribute	Value	DataType
BrowseName	1:Station	QualifiedName
Description	Station	LocalizedText
DisplayName	Station	LocalizedText
EventNotifier		Byte
NodeClass	Object	Int32
NodeId	ns=1;i=1	NodeId
UserWriteMas		UInt32
WriteMask		UInt32

	ReferenceType	NodeId	BrowseName	TypeDefinition
1	HasTypeDefini...	BaseObjectType	0:BaseObjectT...	Null

ns: http://192.168.0.120:4841 Name: Waveheight Auto NodeId ns=1;i=9 0.0 0:Double

```
opcua.server.
opcua.server.
opcua.server.
missing')
opcua.server.address_space - WARNING - Tried to read attribute 'AttributeIds.Historizing' in NumericNodeId(ns=1;i=1), but the attribute is missing')
opcua.server.address_space - WARNING - Tried to read attribute 'AttributeIds.Executable' in NumericNodeId(ns=1;i=1), but the attribute is missing')
opcua.server.address_space - WARNING - Tried to read attribute 'AttributeIds.UserExecutable' in NumericNodeId(ns=1;i=1), but the attribute is missing')
```

Figure 5: Python OPC UA Modeler tool.

Listing 3: Python OPC UA Client App

```
01 # station1.py - Put OPC UA data into gauges
02 #
03 import tkinter as tk
04 import tk_tools
05 import opcua
06
07 # Connect to the OPC UA server as a client
08 client = opcua.Client("opc.tcp://192.168.0.120:4841")
09 client.connect()
10
11 root = tk.Tk()
12 root.title("OPC-UA Weather Station 1")
13
14 # Create 2 gauge objects
15 gtemp = tk_tools.Gauge(root, height = 200, width = 400,
16             max_value=50, label='Temperature', unit='°C')
17 gtemp.pack()
18 gwind = tk_tools.Gauge(root, height = 200, width = 400,
19             max_value=100, label='Windspeed', unit='kph')
20 gwind.pack()
21
22 def update_gauge():
23     # update the gauges with the OPC UA values every 1
         second
24     gtemp.set_value(client.get_node("ns=2;i=2").get_
         value())
25     gwind.set_value(client.get_node("ns=2;i=5").get_
         value())
26     root.after(1000, update_gauge)
27
28 root.after(500, update_gauge)
29
30 root.mainloop()
```

Bash/Awk program to translate the three-field CSV file to the required OPC UA XML format (Listing 4).

In the Bash script, the first `awk` section (lines 6-15) prints out the XML header information, and the second `awk` section (lines 18-36) reads the input (CSV) text line by line. A comma field separator (line 19) maps the three CSV fields to the variables `$1`, `$2`, and `$3`. The `UAVariable` definition is printed with `$1` as the variable name (lines 24-25), `$2` as the description (line 26), and `$3` as the default value (line 31). The `awk` variable `i` increments the NodeID index (line 23).

To run this script to read the CSV file `mytags.csv` and create the XML file `mytags.xml`, enter:

```
cat mytags.csv | /csv2xml.sh > mytags.xml
```

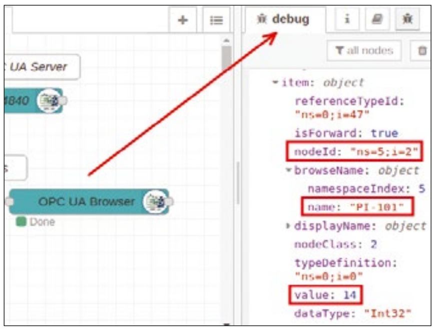
Now that I have a tool to create XML configuration files, I can devise small test systems that easily scale up to larger tag counts. Also, having a starting file in CSV lets me manage my tag changes and additions in Excel or LibreOffice.

Node-RED OPC UA Server

The Node-RED *node-red-contrib-opcua* node [7] includes an OPC UA server and most of the common OPC UA client functions. It can be installed within Node-RED from the *Manage Palette* option.

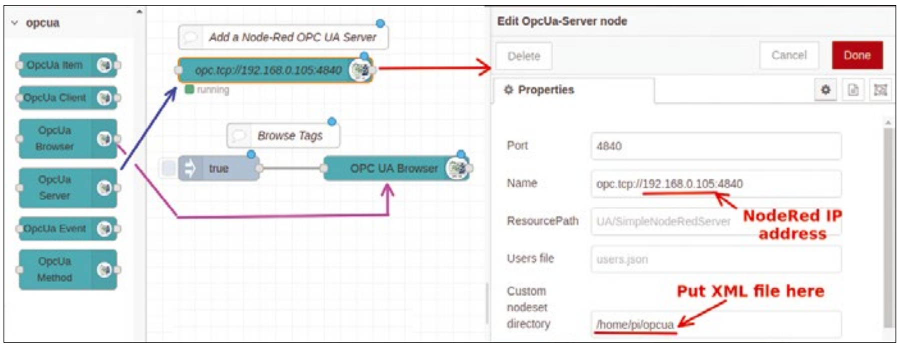
Figure 6 shows the setup of a Node-RED OPC UA server with a browser node that lists objects configured in the server. The *OpcUa server* node name is defined (`<IP-address>:4840` in this case), and the Custom nodeset directory is set to `/home/pi/opcua`, into which I copy the XML file that I created from the CSV file (`mytags.xml`).

The *OpcUa Browser* node sends messages directly to the debug pane



```
▾item: object
  referenceTypeId: "ns=0;i=47"
  isForward: true
  nodeId: "ns=5;i=2"
  ▾browseName: object
     namespaceIndex: 5
     name: "PI-101"
  ▸displayName: object
  nodeClass: 2
  typeDefinition: "ns=0;i=0"
  value: 14
  dataType: "Int32"
```

Figure 7: Browser node shows OPC UA pressure object PI-101 in the debug pane with NodeID ns=5;i=2.



Port: 4840
Name: opc.tcp://192.168.0.105:4840
ResourcePath: UA/SimpleNodeRedServer
Users file: users.json
Custom nodeset directory: /home/pi/opcua

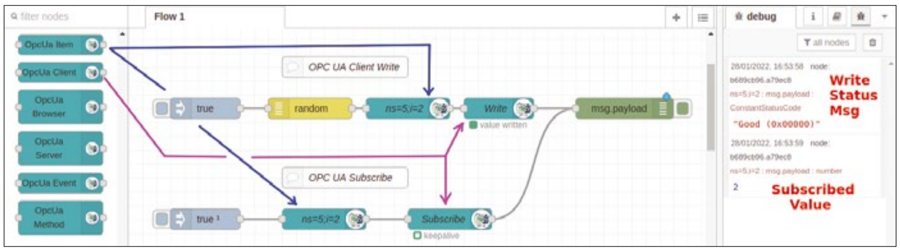
Figure 6: Node-RED OPC UA server and browser logic.

(Figure 7), which allows me to see the objects and variables I defined in my XML file.

The next step is to look at writing and reading values. Figure 8 shows the Node-RED logic to write a random number and then read the value back as a subscribed datapoint. The simplest way to communicate with an OPC UA server is to use an *OpcUa Item* node to define the NodeID and an *OpcUa Client* node to initiate an action.

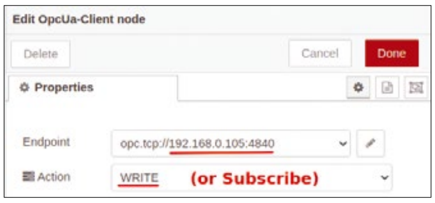
Listing 4: CSV to OPC UA XML

```
01 #!/usr/bin/bash
02 # csv2xml.sh - create an OPC UA XML file from CSV
03 #
04
05 # add the xml header info
06 awk ' BEGIN {
07   print "<?xml version=\"1.0\" encoding=\"utf-8\"?>"
08   print "<UANodeSet xmlns=\"http://opcfoundation.org/UA/2011/03/UANodeSet.xsd\""
09   print "           xmlns:uax=\"http://opcfoundation.org/UA/2008/02/Types.xsd\""
10   print "           xmlns:xsd=\"http://www.w3.org/2001/XMLSchema\""
11   print "           xmlns:xsi=\"http://www.w3.org/2001/XMLSchema-instance\">"
12   print "<NamespaceUris>"
13   print "   <Uri>http://192.168.0.120:4841</Uri>" ;
           # This address would be passed in
14   print "</NamespaceUris>"
15 }'
16
17 # Read the input CSV format and process to XML
18 awk ' {
19    FS="," ; # separate fields with a comma
20   # Skip any comment lines that start with a #
21   if ( substr($1,1,1) != "#" )
22   {
23     i = i+1 ; # increment the NodeID index
24     print "<UAVariable BrowseName=\"1:"$1"\" DataType=\"Int32\" NodeId=\"ns=1;i="i"\"
           ParentNodeId=\"i=85\">"
25     print "  <DisplayName>"$1"</DisplayName>" ;
           # set the display name to the 1st field
26     print "  <Description>"$2"</Description>" ;
           # set the description to the 2nd field
27     print "     <References>"
28     print "       <Reference IsForward=\"false\" Reference
           Type=\"HasComponent\">i=85</Reference>"
29     print "     </References>"
30     print "   <Value>"
31     print "     <uax:Int32>"$3"</uax:Int32>" ;
             # set the default value to the 3rd field
32     print "   </Value>"
33     print "</UAVariable>"
34   }
35 }
36 END{ print "</UANodeSet>"} '
```



```
28/01/2022, 16:53:58  node:
b689cb06.a79ec8
ns=5;i=2 : msg.payload :
ConstantStatusCode
"Good (0x00000)"
28/01/2022, 16:53:59  node:
b689cb06.a79ec8
ns=5;i=2 : msg.payload : number
2
```

Figure 8: Write and subscribe to an OPC UA value.



Endpoint: opc.tcp://192.168.0.105:4840

Action: WRITE (or Subscribe)

Figure 9: Define an Endpoint and Action for an OpcUa-Client node.

The pressure NodeID `ns=5;i=2` from Figure 7 is entered into the *OpcUA Item* node. The *OpcUa Client* node needs a defined Endpoint address and Action (Figure 9). When a `WRITE` action is issued, a *Good* or *Bad* status message is returned.

The *OpcUa Client* node supports a number of actions. Rather than a `READ` action, as used in the Python client app, a `SUBSCRIBE` action returns a value whenever the value changes.

Node-RED with OPC UA Server

For the final example, I use the Python OPC UA server from the first example. The `Temperature` and `Windspeed` tags from the Python example use the same simulation code, but an added `Wave Height` tag will be a manually entered value from Node-RED.

Figure 10 shows a Node-RED application that connects to the Python OPC UA server and presents the data in a Node-RED dashboard. This example subscribes to two real-time inputs (`Temperature` and `Wind Speed`) and presents the values in gauges. The *OpcUA Item* nodes define the OPC UA NodeIDs to be used.

All the *OpcUa Client* nodes need their endpoints defined to the Python OPC UA server's address.

The subscribed data values are returned as a two-item array (because the data type is an Int64). The *Gauge* node only reads the first payload array item (which is 0), so a small *Function* node copies the second payload item (`msg.payload[1]`) to the payload message:

```
// Copy second payload array item ↩
  to be the payload
// Note: msg.payload[0] = 0, the ↩
  value is in payload[1]
msg.payload = msg.payload[1]
return msg;
```

For this example, a manual input for the `Wave Height` tag is subscribed to like the other tags, and the slider position is updated to its value. The slider can also be used to set the value manually by having the slider output passed to an *OpcUa Client* node with a `WRITE` action.

After the logic is complete, the *Deploy* button makes the application live. The Node-RED dashboard can be viewed at *http://node-red-ip:1880/ui*.

Summary

Learning OPC UA can be quite intimidating because of all the features; however, if you start small with Python and Node-RED, it's possible to put together a usable test system with a minimal amount of code and setup.

In this article, I only touched on data access, with no security settings. The next steps would be to look at more advanced features like alarms and events, historical data access, and user security.

Also, you should note that most high-end OPC UA servers support access to OPC UA items with their browser names, so instead of accessing a point with the NodeID `ns=5;i=6`, a browser name string can be used (e.g., `ns=5;s=MYTAGNAME`). ■■■

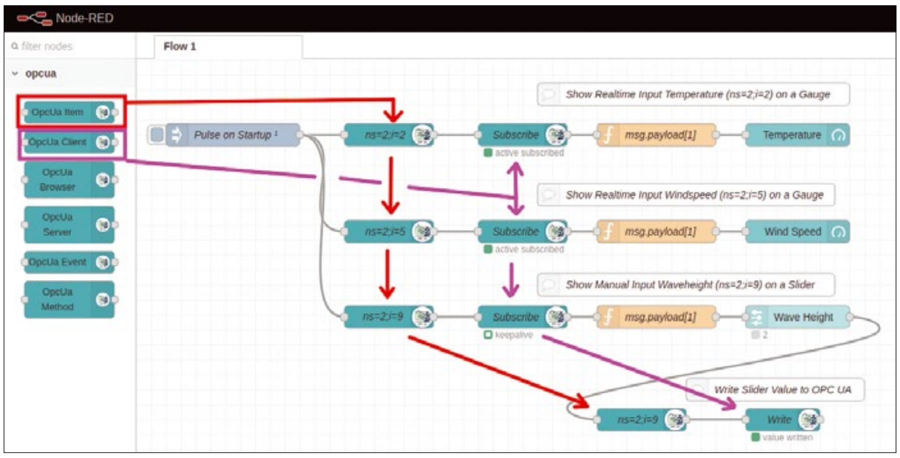


Figure 10: Node-RED logic to show OPC UA data on a dashboard.

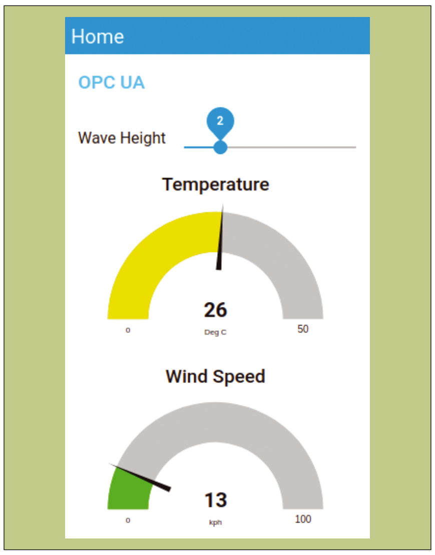


Figure 11: Node-Red OPC UA dashboard.

Info

[1] OPC Foundation: *https://opcfoundation.org/*

[2] OPC Unified Architecture: *https://opcfoundation.org/about/opc-technologies/opc-ua/*

[3] open62541 Project: *https://open62541.org/*

[4] Free OPC-UA Library project: *https://github.com/FreeOpcUa*

[5] Code for this article: *ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/258/*

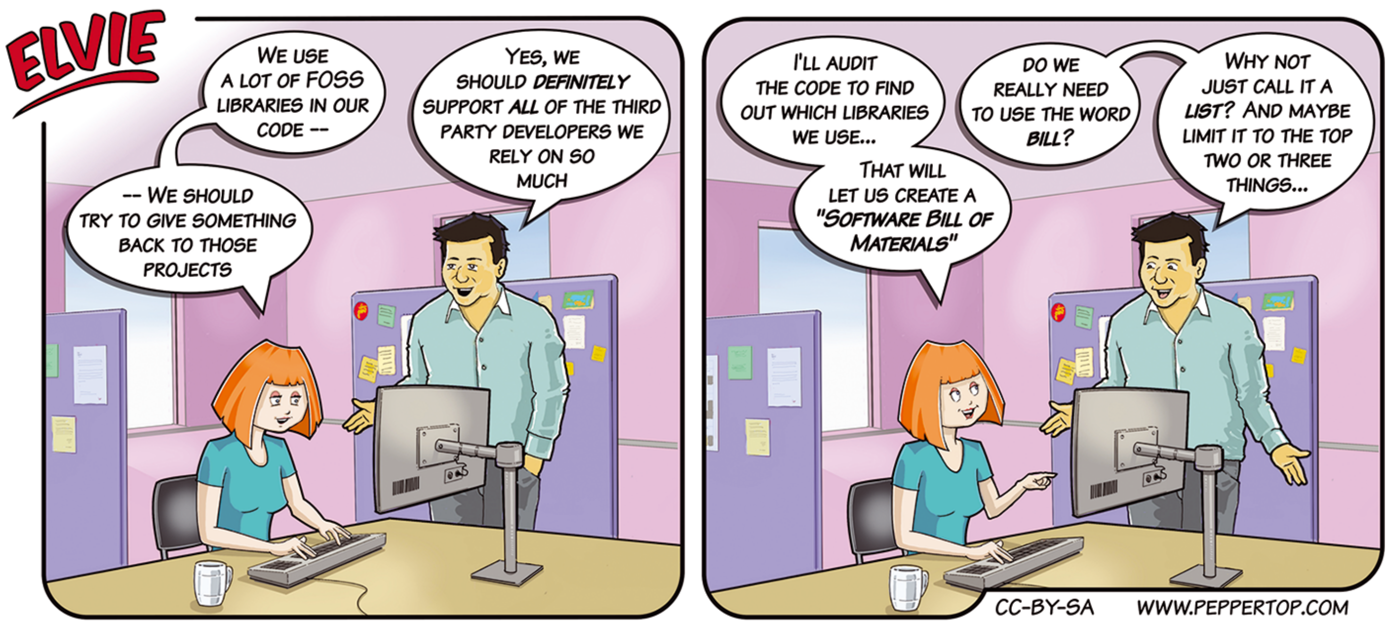
[6] Python tk_tools library: *https://tk-tools.readthedocs.io/*

[7] Node-RED OPC UA docs: *https://flows.nodered.org/node/node-red-contrib-opcua*

The open source ecosystem has a solution for every problem – in fact, it often has many solutions. You might need a powerful and full-featured tool to build a powerful and full-featured application, or you might just want something that is fast and easy to implement. When it comes to database systems, the options run the spectrum, from enterprise-ready tools like PostgreSQL to less intimidating alternatives like MariaDB. But then, if you *really* want to keep it simple, you could always just use LibreOffice. This month we fire up LibreOffice Base for a simple home database project: organizing a personal music collection. Also this month, we take a crack at eg, a tool that lets you call up command syntax examples at the command line, and we show you how to use Inkscape and a tool called Paperfold to create arty 3D objects with folded paper.

Image © Olexandr Moroz, 123RF.com

LINUXVOICE


ELVIE
We use a lot of FOSS libraries in our code --
-- We should try to give something back to those projects
Yes, we should definitely support all of the third party developers we rely on so much
I'll audit the code to find out which libraries we use...
That will let us create a "Software Bill of Materials"
Do we really need to use the word bill?
Why not just call it a list? And maybe limit it to the top two or three things...
CC-BY-SA
WWW.PEPPERTOP.COM

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

Bugs and security issues aren't limited to open source software, despite comments to the contrary. BY JON "MADDOG" HALL

Where do they come from?

Whether you are sailing a ship at sea or shooting a rocket into outer space, you often have course corrections. A couple of people in the press and in personal conversation have recently made statements which indicate that people need a "course correction."

One example was in a loud discussion about the amount of bugs or security issues that exist in "open source" (and I use that term specifically rather than "free software") and how this is somehow big news. This has led to additional discussion about the need for a Software Bill of Materials (SBOM) which indicates what pieces of open source appear in the products of companies. Don't get me wrong, I think an SBOM is a good idea, particularly if it could be done easily and almost automatically through the software build process. However, people talk about "the bug problem" as if it were solely the problem of open source and that it has not been a problem of all software throughout time.

When software was a lot simpler, people could wrap their minds around the code and (with a couple of good engineers) manage to keep a hand on the libraries and environments. However, I have worked for closed source companies where large portions of code were not reviewed by software engineers over large periods of time, because of a lack of resources. I have seen large amounts of code not covered by test scripts or regression tests. It happens in both free and open source software (FOSS) and closed source, but "FOSS" or "Linux" are the names that grab attention.

The difference with open source (and particularly free software) is that the code is exposed to many eyes and is available for patching when the problem is understood. Closed source code (or closed source that uses open source components) cannot be patched by the end user. The end user must rely on the company that produced the closed source product to get the patch, and that assumes the closed source company actually produces a patch for that software.

I do not mind that the press and pundits point out that there are bugs in FOSS. I do mind when they somehow imply that this is a problem with FOSS and not also problem with closed source.

Somewhat tied to this is a recent incident where an open source developer purposely put a bug into the code they were developing, ostensibly because they objected to companies making large amounts of money off the code that FOSS people generate. This created a call for companies to compensate FOSS people for their work in the FOSS community.

While I applaud developers being paid for their time and effort in creating FOSS projects, and while I share the frustration of many people with multi-billionaires receiving their riches, making tens of thousands of times the money that "their workers" make, the motivation behind FOSS is not being paid for software that others use but to make the software that we ourselves need.

In the early days of computers, there were few, if any, "professional programmers," people paid to write programs. People wrote programs to solve their own problems and then often gave them away to help other people. Perhaps those other people would help to make the programs better.

Eventually there were people who did not know how to program who could use these programs too … and that was great. Software that no one uses is useless.

Shortly after the Linux kernel project was started a number of developers brought forth the issue of "companies making money from the software I write for free," and ways of making these companies pay for the software were considered. It was observed, however, that if this path was followed, then FOSS would move forward slowly, like a glacier. Some people left the project, and some were hired by companies that saw the value of hiring them, and some continued to write the software for their own purposes.

When I talk to programmers about becoming involved with FOSS, I first suggest that they pick an area of software that they otherwise have a passion for: audio, video, games, databases and data mining, word processing tools … the list is endless. They will learn more about their passion and perhaps be hired by a company that needs their expertise delivering on that passion.

Or you may anticipate that the software you are working on facilitates an industry where you can make money, but this is a lot harder and you have to be careful not to create a project that is not sustainable, both for you and the other people using it. However, we should not purposely punish users if we no longer get satisfaction out of our software development work.

The press and pundits are insinuating these are only "open source" issues, and they are not.

Let's keep them honest. ■■■

Shell tool examples in the terminal

eg Explains

Need to know how to use a command-line tool? eg provides real life examples, and it is easier to access than the man pages. BY FERDINAND THOMMES

On forums and in discussion groups, you increasingly find people claiming that using the command line is an anachronism. They say that it should be possible to handle any task with graphical tools. This kind of statement shows that whoever posted the comments has never seriously tried to work with a terminal. Anyone who is aware of the speed and elegance of working at the command line would never agree that graphical tools alone would be preferable. I admit to using a hybrid of command line and graphical tools on the Plasma desktop. I have increasingly used both, equally, in combination for many years, and there is no way I would want to do without the command line given this very satisfying division of labor.

Complex Man Pages

There are several ways to approach working in a terminal as a newcomer with the willingness to learn. The first place to start for many users is the man pages, which are great as a reference because they list all the options, flags, and parameters for a tool. But if you actually want to learn how to use command-line interface (CLI) tools, the man pages offer very limited help – in fact, they are more likely to scare off most newcomers. Hands-on examples provide far more easily digestible information, and there are countless examples on the web. The trick is finding the right example from such a massive selection.

This is where the interactive tool `eg` enters the scene. It stands for *exempli gratia* (e.g. for short), which comes from Latin. The name makes sense, until you need to search the web or your hard drive for "eg" and are bombarded with matches. `eg`'s self-description states that it provides examples of common uses of command line tools. I checked how well this works.

Quickly Installed

To do so, I first visited the tool's GitHub page [1] to see what the install looks like. The tool is not available from the package archives of most of the popular Linux distributions. You can install it with `pip`, the package installer for Python packages from the Python Package Index. But to do this, you may first need to install the *python3-pip* package on your machine from the package manager, if it is not already in place. You then retrieve eg with the `pip install eg` command and store it on your hard disc.

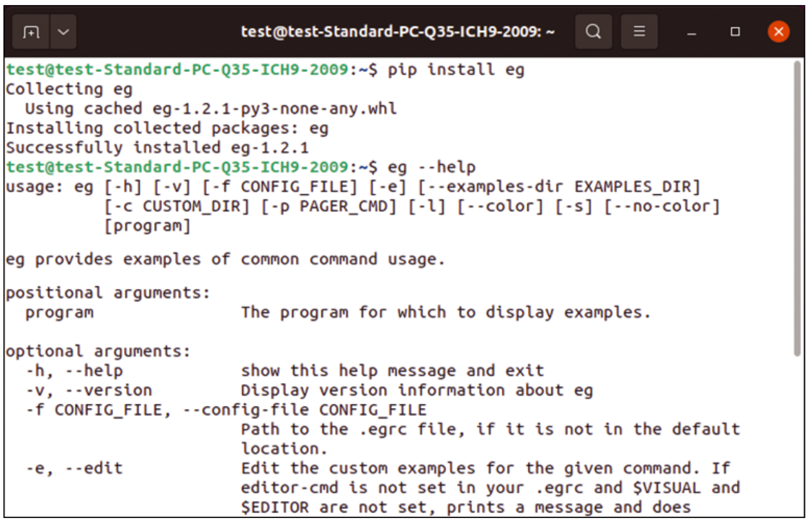
Once `eg` is installed and ready to use, there is no need for configuration, but it is possible. As your first official step, type the `eg --list` command, which shows you an alphabetically sorted list of the 80 or so tools and commands currently supported (Figure 1). If the system outputs a *command not found* message when you do this, you will probably have to log off the desktop and log back on again. The system will then include the installation directory in its path.

Examples of one of the listed commands are easy to access by calling `eg NAME`. You will notice that some entries have only a few lines (Figure 2), while others – such as the entry for Git – have several hundred lines. Once you have finished reading an entry, pressing *Q* will return you to the prompt. If you do not want to continue using `eg` after a test, simply remove it by typing `pip uninstall eg`.

Scope for Your Own Ideas

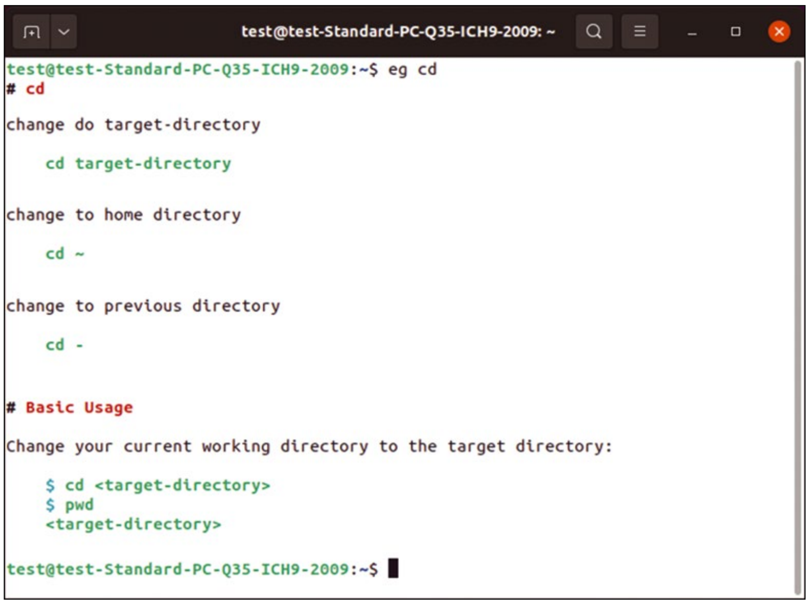
`eg` stores in `$HOME/.local/lib/python3.9/site-packages`. Check out the "Installation Path" box for more details. In the installation directory,

Figure 1: Installing `eg` is painless. If you have not already done so, you need to install the *python3-pip* package first.



test@test-Standard-PC-Q35-ICH9-2009: ~

```
test@test-Standard-PC-Q35-ICH9-2009:~$ pip install eg
Collecting eg
  Using cached eg-1.2.1-py3-none-any.whl
Installing collected packages: eg
Successfully installed eg-1.2.1
test@test-Standard-PC-Q35-ICH9-2009:~$ eg --help
usage: eg [-h] [-v] [-f CONFIG_FILE] [-e] [--examples-dir EXAMPLES_DIR]
          [-c CUSTOM_DIR] [-p PAGER_CMD] [-l] [--color] [-s] [--no-color]
          [program]

eg provides examples of common command usage.

positional arguments:
  program               The program for which to display examples.

optional arguments:
  -h, --help            show this help message and exit
  -v, --version         Display version information about eg
  -f CONFIG_FILE, --config-file CONFIG_FILE
                        Path to the .egrc file, if it is not in the default
                        location.
  -e, --edit            Edit the custom examples for the given command. If
                        editor-cmd is not set in your .egrc and $VISUAL and
                        $EDITOR are not set, prints a message and does
```



```
test@test-Standard-PC-Q35-ICH9-2009:~$ eg cd
# cd

change do target-directory

    cd target-directory

change to home directory

    cd ~

change to previous directory

    cd -


# Basic Usage

Change your current working directory to the target directory:

    $ cd <target-directory>
    $ pwd
    <target-directory>

test@test-Standard-PC-Q35-ICH9-2009:~$ 
```

Figure 2: Working with **eg** is very simple. You can see the output for the **cd** (change directory) command here.

you will find the supported commands and tools below the `eg/examples/` folder in Markdown format. If you want to add examples or parameters yourself, open the corresponding Markdown file and make your changes.

Also, `eg` lets you store your own entries in the data structure. To do this, create a file named `NAME.md` in `examples/`, and `eg` will load it automatically from there. Ultimately, the tool does nothing more than match a command you type, such as `eg git`, with the files in the default and custom directories (including subfolders). If `eg` finds a matching entry, it outputs the entry, piping it through the `less` pager (Figure 3) to do so.

If you want to organize your own creations in a separate directory, create a configuration file named `$HOME/.config/eg/egrc` or `$HOME/.egrc`. In the simplest case, this will look something like Listing 2. You can also define an alternate location for the default `examples/` directory. The output can also be extensively customized to suit your own needs, for example, by adding colors. Examples of this can be found on the project's GitHub page [1].

Conclusions

Newcomers and power users alike will find `eg` to be a small but useful tool – because, let's be honest, virtually nobody can remember all of those commands and options. With `eg`, all it takes is a short command to call up explanations for around 80 tools that are far easier to follow than the actual man pages. ■■■

Installation Path

You install `eg` in `$HOME/.local/lib/`, and the executable is located in `$HOME/.local/bin`. You need to make sure that this directory is in your path variable. You can determine whether this is the case by typing `echo $PATH`. If the directory is missing from the list, you can add it temporarily by typing `export PATH="$HOME/.local/bin:$PATH"`. But this configuration will not survive the next reboot. Use the entry shown in Listing 1 to store it permanently in your `$HOME/.profile`. Most desktop environments preconfigure the shell appropriately. But the `if` query in Listing 1 only integrates the path if the directory already exists at login time. You may need to log out and back on again after installing `eg`.

Listing 1: ~/.profile

```
[...]
# set PATH so it includes user's
# private bin if it exists
if [ -d "$HOME/.local/bin" ]; then
  PATH="$HOME/.local/bin:$PATH"
fi
[...]
```

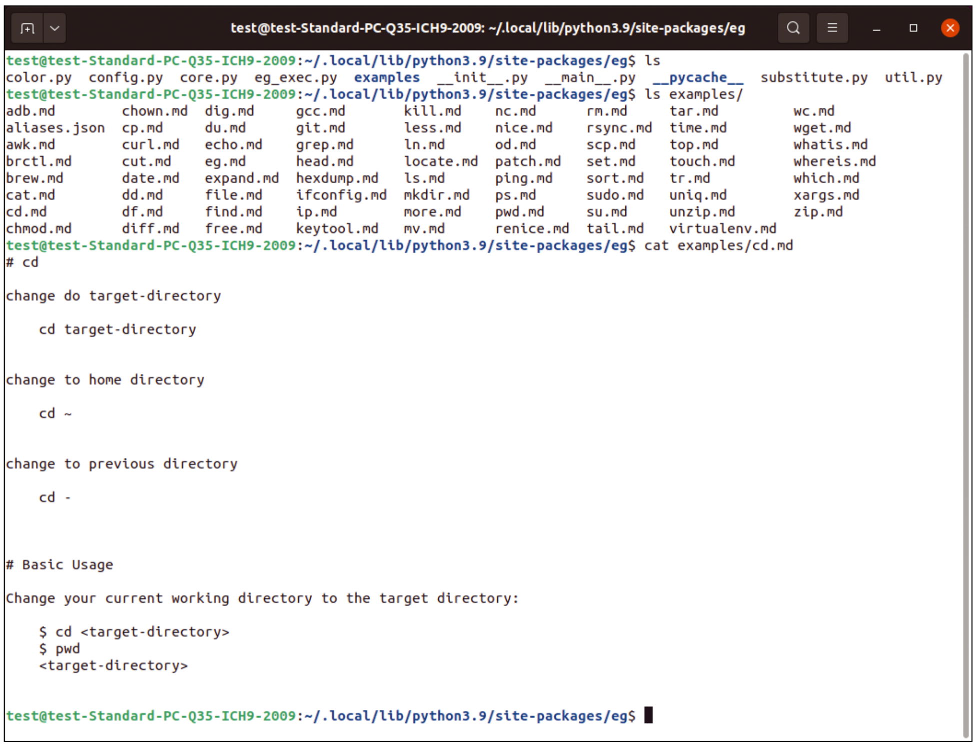
Listing 2: Configuration

```
[eg-config]
custom-dir = $HOME/my-eg
```

Info

[1] eg: *https://github.com/srsudar/eg*

Figure 3: In the installation directory below **examples/**, you will find the Markdown files that act as the basis for the output. You can customize these files or create your own examples.



test@test-Standard-PC-Q35-ICH9-2009: ~/.local/lib/python3.9/site-packages/eg

```
test@test-Standard-PC-Q35-ICH9-2009:~/.local/lib/python3.9/site-packages/eg$ ls
color.py  config.py  core.py  eg_exec.py  examples  __init__.py  __main__.py  __pycache__  substitute.py  util.py
test@test-Standard-PC-Q35-ICH9-2009:~/.local/lib/python3.9/site-packages/eg$ ls examples/
adb.md        chown.md  dig.md     gcc.md       kill.md    nc.md      rm.md     tar.md         wc.md
aliases.json  cp.md     du.md      git.md       less.md    nice.md    rsync.md  time.md        wget.md
awk.md        curl.md   echo.md    grep.md      ln.md      od.md      scp.md    top.md         whatis.md
brctl.md      cut.md    eg.md      head.md      locate.md  patch.md   set.md    touch.md       whereis.md
brew.md       date.md   expand.md  hexdump.md   ls.md      ping.md    sort.md   tr.md          which.md
cat.md        dd.md     file.md    ifconfig.md  mkdir.md   ps.md      sudo.md   uniq.md        xargs.md
cd.md         df.md     find.md    ip.md        more.md    pwd.md     su.md     unzip.md       zip.md
chmod.md      diff.md   free.md    keytool.md   mv.md      renice.md  tail.md   virtualenv.md
test@test-Standard-PC-Q35-ICH9-2009:~/.local/lib/python3.9/site-packages/eg$ cat examples/cd.md
# cd

change do target-directory

    cd target-directory

change to home directory

    cd ~

change to previous directory

    cd -


# Basic Usage

Change your current working directory to the target directory:

    $ cd <target-directory>
    $ pwd
    <target-directory>


test@test-Standard-PC-Q35-ICH9-2009:~/.local/lib/python3.9/site-packages/eg$ 
```

Creating a LibreOffice Music Database

Tune Finder

LibreOffice Calc and Base are all you need to create a simple database for organizing the songs in your music collection.

BY JOHN COFIELD

MySQL is the most commonly used open source database management system. Developers often use MySQL and its cousin MariaDB to build database applications for organizing office records, managing inventories, and other common tasks. However, MySQL is often too complex and too much trouble for personal, home-office uses. LibreOffice offers a simpler alternative for users who just want to create a small, simple database to address a specific need. This article describes how to create a quick and easy database solution using LibreOffice tools. In this case, I'll show you how to set up a music database from an iTunes library.

The Plan

I'll use two tools from the LibreOffice integrated suite to create my music database: LibreOffice Calc (spreadsheet) and LibreOffice Base (database management). With these two applications plus iTunes, the general process is as follows:

1. Export library from iTunes as a tab-separated text file.
2. Import library into LibreOffice Calc for minor edits.
3. Copy modified library data into LibreOffice Base.
4. Create SQL queries.
5. Run queries and filters to display results.

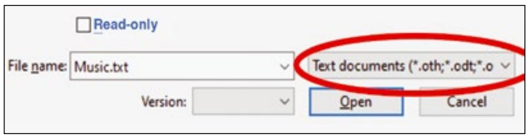
Of course, you can adapt this process for other types of data. For instance, you could organize a stamp collection or track incoming invoices for this year's taxes.

Export from iTunes

The first step is to export music information from iTunes to a text file in a format that can be imported into LibreOffice. Playlists can be exported from iTunes as a table in a tab-delimited plain text file format. The comma-separated values (CSV) format is unacceptable because some data fields (such as album titles, song titles, or artist names) may contain commas or other special characters. The exported playlist may consist of some or all of the songs in the library. I use the following steps to export my playlist:

1. Select all songs in the library
2. Go to *File | Library | Export Playlist*
3. In the *File name* field, enter *Music.txt*
4. Select *Text file (*.txt)*

These steps save the playlist as a tab-separated text file, `Music.txt` (but you choose any name you want), that can be directly imported as a plain text spreadsheet.

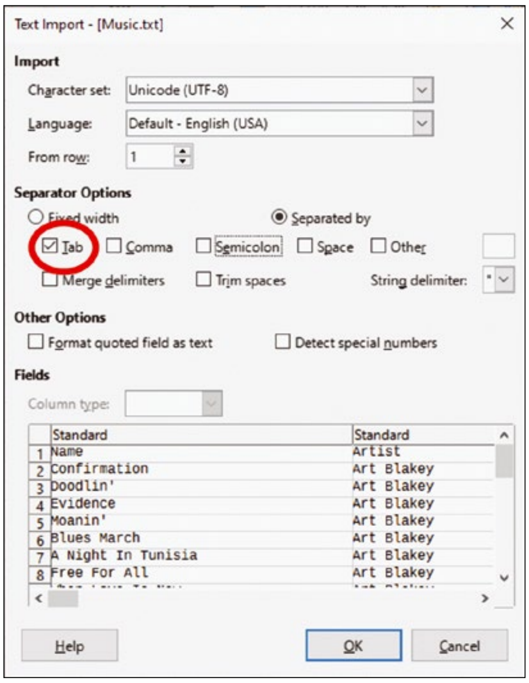


Read-only
File name: Music.txt
Text documents (*.oth;*.odt;*.o
Version:

Figure 1: Select ***Text documents*** as the file format.

Import Text File into LibreOffice Calc

Next, I import the tab-separated file `Music.txt` into LibreOffice Calc for minor editing. When I open the text file, LibreOffice Calc pops up a Text



Text Import - [Music.txt]

Import
Character set: Unicode (UTF-8)
Language: Default - English (USA)
From row: 1

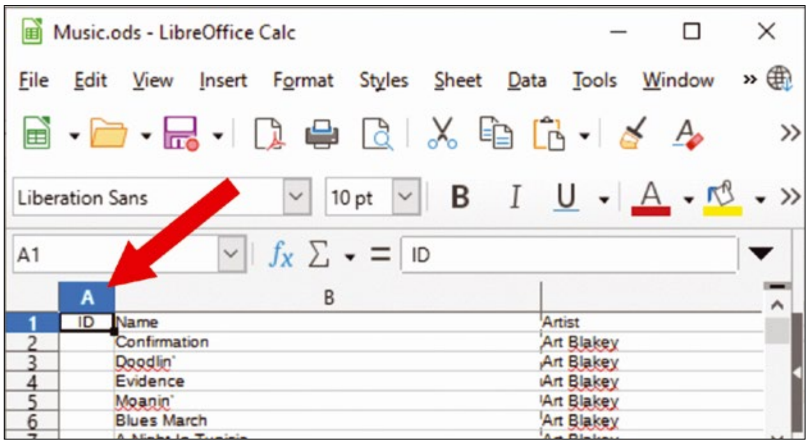
Separator Options
Fixed width [✓ Separated by]
[✓ Tab] Comma Semicolon Space Other
Merge delimiters Trim spaces String delimiter: "

Other Options
Format quoted field as text Detect special numbers

Fields
Column type:

	Standard	Standard
1	Name	Artist
2	Confirmation	Art Blakey
3	Doodlin'	Art Blakey
4	Evidence	Art Blakey
5	Moanin'	Art Blakey
6	Blues March	Art Blakey
7	A Night In Tunisia	Art Blakey
8	Free For All	Art Blakey

Figure 2: The Text Import dialog box.


Music.ods - LibreOffice Calc
File Edit View Insert Format Styles Sheet Data Tools Window
Liberation Sans
10 pt
A1
ID

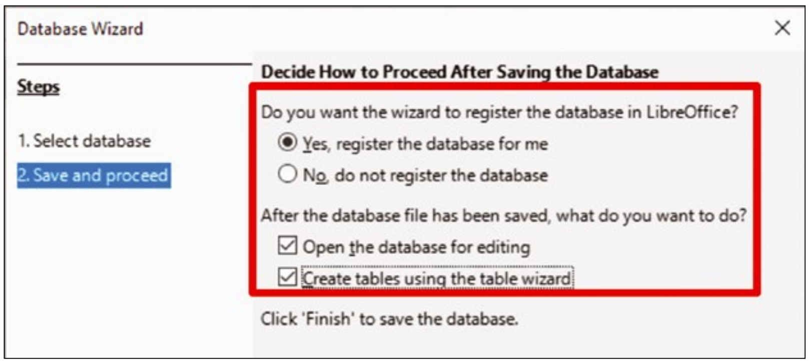
	A	B	
1	ID	Name	Artist
2		Confirmation	Art Blakey
3		Doodlin'	Art Blakey
4		Evidence	Art Blakey
5		Moanin'	Art Blakey
6		Blues March	Art Blakey

Figure 3: Inserting the *ID* column to be used as a primary key.

Import dialog box with the option to select or change the delimiter. To import the file into Calc:

1. Go to *File | Open*
2. Select *Text documents* from the file type drop-down menu (Figure 1)
3. Select *Music.txt*
4. Click *Open*
5. In the *Text Import* dialog box that opens, check the *Tab* option under *Separator Options* and uncheck the other options here (Figure 2)
6. Examine the preview in the *Fields* section
7. Click *OK*

Figure 4: Options for saving a database.


Database Wizard

Steps

1. Select database
2. Save and proceed

Decide How to Proceed After Saving the Database

Do you want the wizard to register the database in LibreOffice?

- (●) Yes, register the database for me
- ( ) No, do not register the database

After the database file has been saved, what do you want to do?

- ☑ Open the database for editing
- ☑ Create tables using the table wizard

Click 'Finish' to save the database.

Figure 5: The Table Wizard shows the steps for setting up a table.

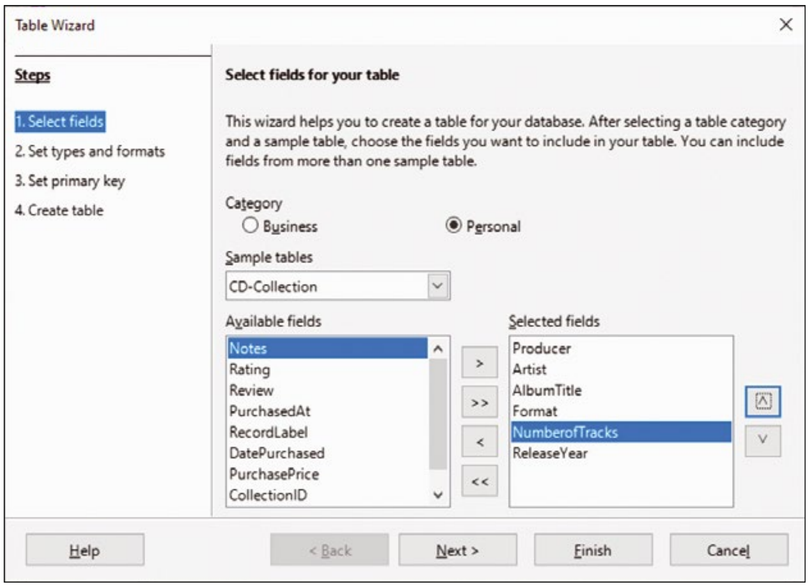


Table Wizard

Steps

1. Select fields
2. Set types and formats
3. Set primary key
4. Create table

Select fields for your table

This wizard helps you to create a table for your database. After selecting a table category and a sample table, choose the fields you want to include in your table. You can include fields from more than one sample table.

Category: ( ) Business [✓ (●) Personal]

Sample tables: CD-Collection

Available fields: Notes, Rating, Review, PurchasedAt, RecordLabel, DatePurchased, PurchasePrice, CollectionID

Selected fields: Producer, Artist, AlbumTitle, Format, NumberofTracks, ReleaseYear

iTunes exports many data fields as table columns, and some of them are not useful or wanted. Because I am only interested in *Name* (song title), *Artist*, *Album*, *Genre*, *Track Number*, and *Year*, I delete all columns except for these. I also insert an *ID* field to be used as a primary key (which I will discuss in the database section of this article.)

To insert the *ID* field (shown in Figure 3), follow these steps:

1. Insert a column in the first position, left of the *Name* column: *Sheet | Insert Columns | Columns Left*
2. Label the new column *ID*
3. Leave cells in the *ID* column blank (they will be automatically filled in LibreOffice Base)
4. Click *Save*
5. Optionally, leave LibreOffice Calc open to cut and paste data later.

At this point, I have the data that I need to populate the database. I need to get that spreadsheet data into a database. To create my music database, I will now use LibreOffice Base, which can create, manage, and edit flat and relational databases.

LibreOffice Base Database

LibreOffice makes use of wizards in most of its applications. I find that wizards are very helpful in LibreOffice Base, especially for users that are not expert in the SQL language. The first step is to create a new database. While LibreOffice Base is capable of importing data directly from a spreadsheet, I prefer to copy and paste the table data from Calc into a predefined template in Base using a wizard.

Create New Database

Next I create and register a new database as follows (Figure 4):

1. Open LibreOffice Base and the Database Wizard appears
2. Click *Create a new database*
3. Click *Next >*
4. Check the following options:
   - *Yes, register the database for me*
   - *Open the database for editing*
   - *Create tables using the table wizard*
5. Click *Finish*
6. Click *Save As MusicLibrary.odb*

Note: Registering the database allows the database to be used by other LibreOffice components such as Writer. It is not required. You can use it at your own discretion.

Next, the Table Wizard (Figure 5) opens, prompting you to do the following steps:

1. Click on *Select fields* under *Steps*. Select *Personal* under *Category*. Then select *CD-Collection* from *Sample tables* drop-down list. Next, select and move the following fields from *Available fields* to *Selected fields*: *Producer*,

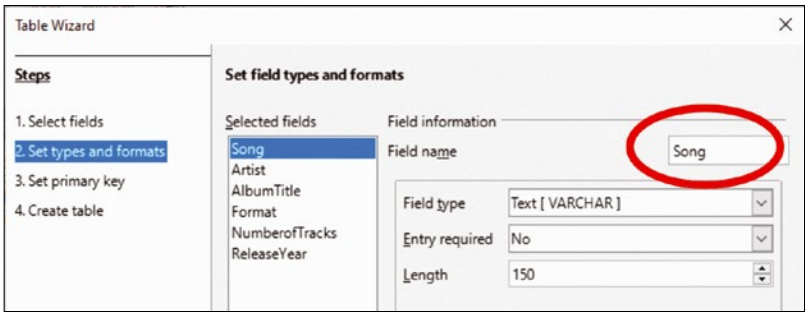


Table Wizard

Steps
1. Select fields
2. Set types and formats
3. Set primary key
4. Create table

Set field types and formats

Selected fields: Song, Artist, AlbumTitle, Format, NumberofTracks, ReleaseYear

Field information
Field name: Song
Field type: Text [VARCHAR]
Entry required: No
Length: 150

Figure 6: Setting field types and formats in the Table Wizard.

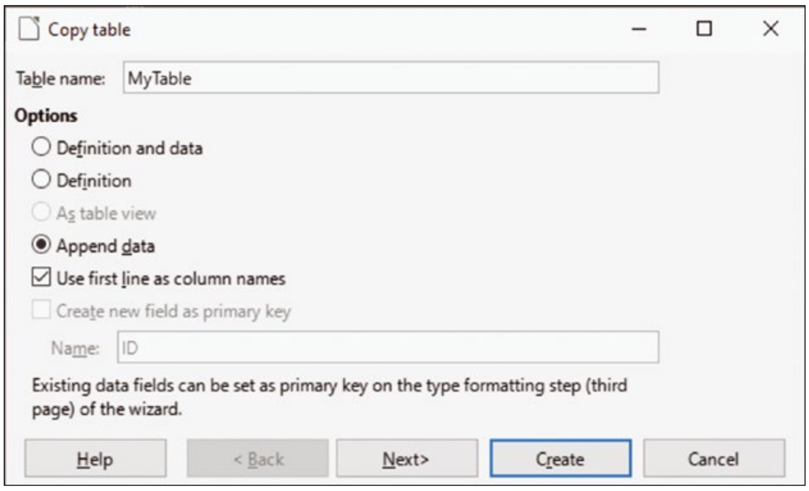

Copy table

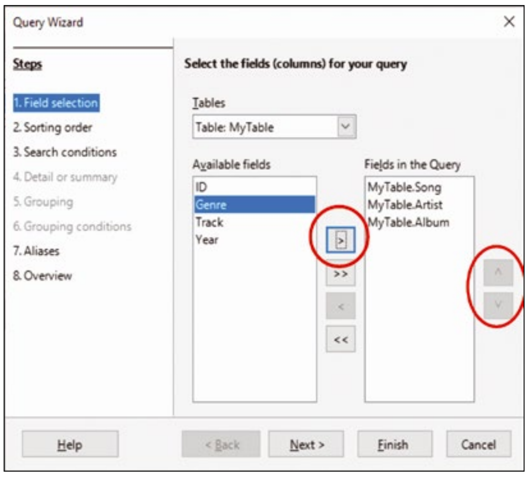
Table name: MyTable

Options
- ( ) Definition and data
- ( ) Definition
- ( ) As table view
- (●) Append data
- [✓] Use first line as column names
- [ ] Create new field as primary key

Name: ID

Existing data fields can be set as primary key on the type formatting step (third page) of the wizard.

Figure 7: Copying a table from Calc into Base.


Query Wizard

Steps
1. Field selection
2. Sorting order
3. Search conditions
4. Detail or summary
5. Grouping
6. Grouping conditions
7. Aliases
8. Overview

Select the fields (columns) for your query

Tables: Table: MyTable

Available fields: ID, Genre, Track, Year

Fields in the Query: MyTable.Song, MyTable.Artist, MyTable.Album

Figure 8: Setting up a query in the Query Wizard.

Artist, *AlbumTitle*, *Format*, *NumberofTracks*, *ReleaseYear*. (Note: Some of these field names will be modified.) Finally, click *Next >*.

2 Click on *Set types and formats*. First, change the *Field name* but keep the corresponding *Field type*. For the *Song*, *Artist*, and *Album* fields, set *Length* to *150* (Figure 6). Next change the *Field name* of the following fields: *Producer* to *Song*, *AlbumTitle* to *Album*, *Format* to *Genre*, *NumberofTracks* to *Track*, and *ReleaseYear* to *Year*. Then, click *Next >*.

3 Click on *Set the primary key*. Check the *Create a primary key* box if not marked already. Select options: *Create a primary key*, *Automatically add a primary key*, and *Auto value*. Click *Next >*.

4 Click on *Create a table*. First change the name to *MyTable*, and then select the option *Insert data immediately*. Click *Finish*.

(The Table Wizard now closes and Table Data View opens. Confirm column labels and close the window. The LibreOffice Base window appears with *Database*, *Tasks*, and *Tables* sections visible.)

Once the table is created in Base, open the `Music.txt` spreadsheet in LibreOffice Calc. Select and copy all data cells from *A1* through the last cell (that includes column label cells).

Now, switch back to Base to paste the copied table data into MyTable in Base.

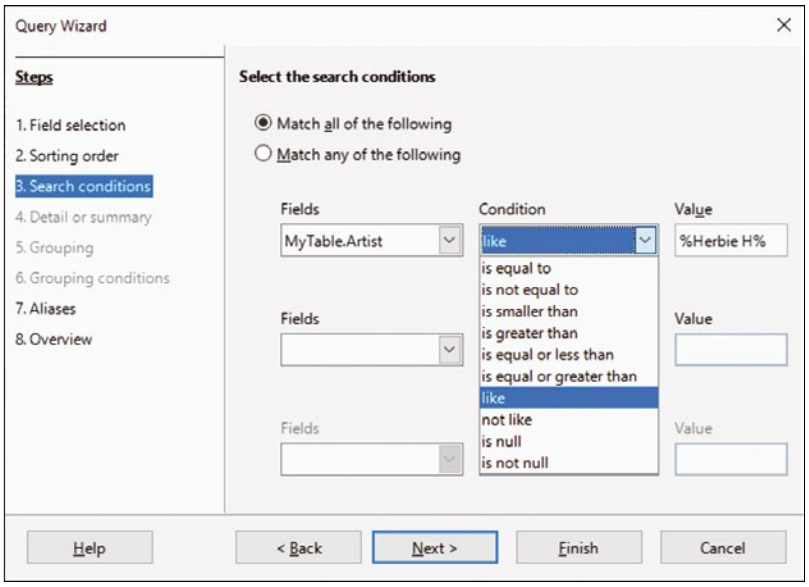
In the *Tables* section, select *MyTable*. Right-click and choose *paste* from the drop-down menu (the Copy table dialog opens, see Figure 7). Select *Append data*, check *Use first line as column names*, and click *Next >* (which will take you to the Assign columns dialog). Review to confirm that the Source table data aligns with the destination table labels, and then click *Create*.

Double-click *MyTable* to open the Table Data View dialog and confirm that the data copied correctly. Set the width of the columns (the width of the query result field is determined by the width of the table fields.) Uncheck *Automatic*.

Set Up and Run Query

I now have the database ready to run SQL queries to find songs, artists, and album information from my music library. The next step is to set up and run queries on the database.

Figure 9: Selecting the search conditions for a query.


Query Wizard

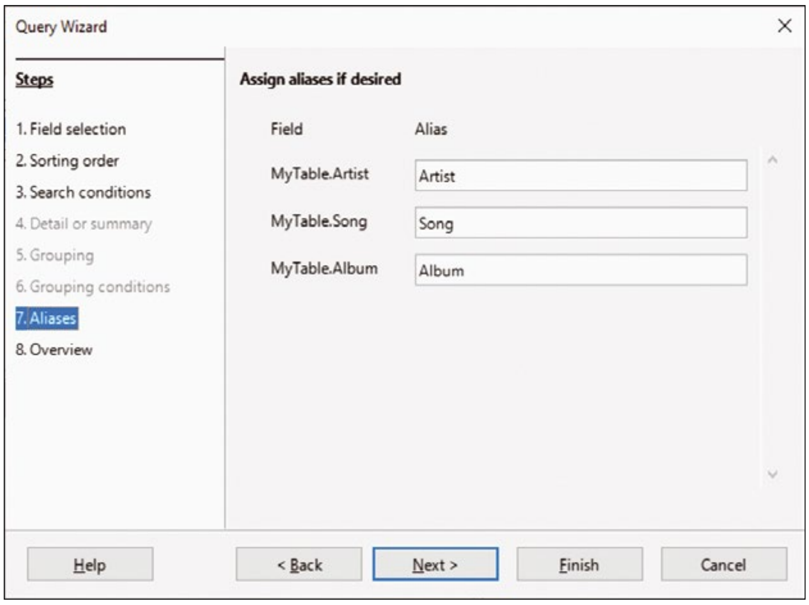
Steps
1. Field selection
2. Sorting order
3. Search conditions
4. Detail or summary
5. Grouping
6. Grouping conditions
7. Aliases
8. Overview

Select the search conditions

- (●) Match all of the following
- ( ) Match any of the following

Fields	Condition	Value
MyTable.Artist	like	%Herbie H%

Condition list: is equal to, is not equal to, is smaller than, is greater than, is equal or less than, is equal or greater than, like, not like, is null, is not null


Query Wizard

Steps

1. Field selection
2. Sorting order
3. Search conditions
4. Detail or summary
5. Grouping
6. Grouping conditions
7. Aliases
8. Overview

Assign aliases if desired

Field	Alias
MyTable.Artist	Artist
MyTable.Song	Song
MyTable.Album	Album

Figure 10: Assigning aliases.

LibreOffice Base provides three methods to create SQL queries: *Create Query in Design View*, *Use Wizard to Create Query*, or *Create Query in SQL View*. I use the *Use Wizard to Create Query* method to set up and run a query. As an example, I first set up a query search for all songs by a particular artist. Then, I use the *Standard Filter* on the query results to refine my searches.

To begin, click on the *Queries* icon in the Database section. Then click on *Use Wizard to Create Query*, and complete the following steps in the Query Wizard:

1. Click on *Field selection* in the Query Wizard: Select *Song*, *Artist*, and *Album* in the *Available fields* window, and then move them to the *Fields in the Query* window. Use the up and down arrows to change the order if necessary (Figure 8).
2. Click on *Sorting order*: Sort in ascending order by *Artist* first, then *Album*, and then *Song*.
3. Click on *Search conditions*: Select *MyTable.Artist* in the *Fields* drop-down list (Figure 9). Select *like* from the *Condition* drop-down list and enter *%Herbie H%* in the *Value* field.
   Note: In SQL, the % sign is used as a wildcard character. The *like* condition matches string patterns. The SQL view shows the SQL statement as:

```
SELECT "Artist" AS "Artist", "Song" ↩
FROM "MyTable" WHERE "Artist" ↩
LIKE '%Herbie H%'
```

   (Skip the *Grouping* and *Grouping conditions* steps in the Query Wizard because grouping will not be used for this project.)
4. Click on *Aliases* (listed at Step 7 in the Query Wizard). The alias for *MyTable.Artist* defaults to *Artist*, and the alias for *MyTable.Song* defaults to *Song*. I use the defaults (as shown in Figure 10).
5. Click on *Overview*. In the *Name of the query* field, I change the default name *Query_MyTable* to *Query_ArtistsAndSongs*. Next review the contents in the *Overview* window. Use the default *Display Query* option.

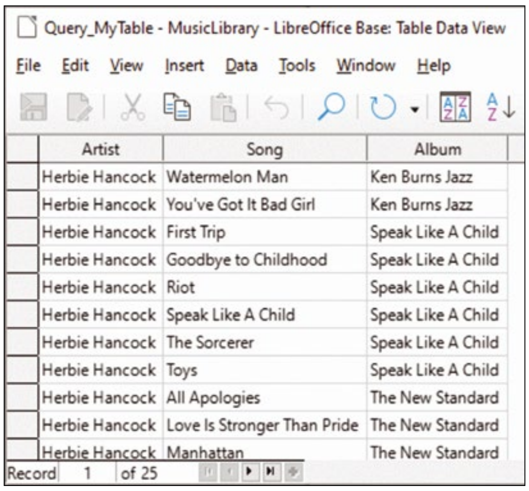
Set Up and Run the Filter

The results that appear in the Table Data View list all songs and albums by Herbie Hancock in my music library. To narrow my search down to just songs in one of the three albums, I could create another query by repeating the five steps above, but there is an easier way. I use the *Standard Filter* in the Table Data View from this query result.

If the Table Data View is not already open, I can click on the *Queries* icon in the *Database* section Next I double-click *Query_MyTable*, and the Table Data View opens (Figure 11).

From there, I click on the *Standard Filter icon* in toolbar (Figure 12). In the Standard Filter dialog

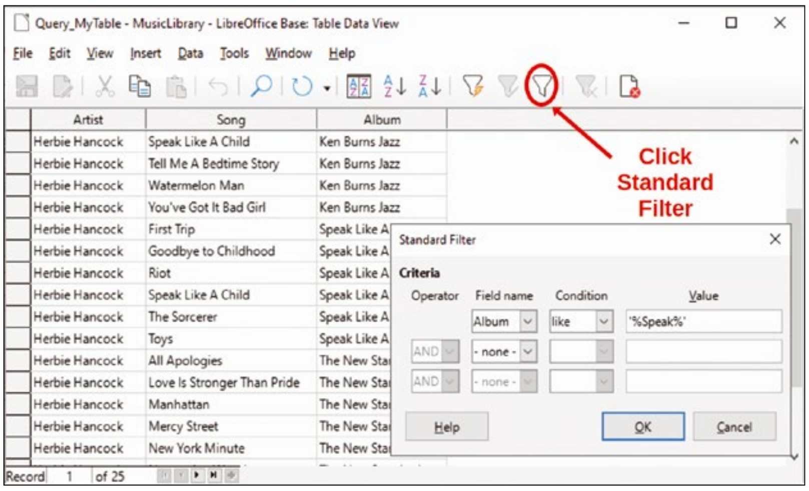
Figure 11: The Table Data View dialog.


Query_MyTable - MusicLibrary - LibreOffice Base: Table Data View

Artist	Song	Album
Herbie Hancock	Watermelon Man	Ken Burns Jazz
Herbie Hancock	You've Got It Bad Girl	Ken Burns Jazz
Herbie Hancock	First Trip	Speak Like A Child
Herbie Hancock	Goodbye to Childhood	Speak Like A Child
Herbie Hancock	Riot	Speak Like A Child
Herbie Hancock	Speak Like A Child	Speak Like A Child
Herbie Hancock	The Sorcerer	Speak Like A Child
Herbie Hancock	Toys	Speak Like A Child
Herbie Hancock	All Apologies	The New Standard
Herbie Hancock	Love Is Stronger Than Pride	The New Standard
Herbie Hancock	Manhattan	The New Standard

Record 1 of 25

Figure 12: Selecting filter criteria in the Standard Filter dialog.


Query_MyTable - MusicLibrary - LibreOffice Base: Table Data View

Artist	Song	Album
Herbie Hancock	Speak Like A Child	Ken Burns Jazz
Herbie Hancock	Tell Me A Bedtime Story	Ken Burns Jazz
Herbie Hancock	Watermelon Man	Ken Burns Jazz
Herbie Hancock	You've Got It Bad Girl	Ken Burns Jazz
Herbie Hancock	First Trip	Speak Like A
Herbie Hancock	Goodbye to Childhood	Speak Like A
Herbie Hancock	Riot	Speak Like A
Herbie Hancock	Speak Like A Child	Speak Like A
Herbie Hancock	The Sorcerer	Speak Like A
Herbie Hancock	Toys	Speak Like A
Herbie Hancock	All Apologies	The New Sta
Herbie Hancock	Love Is Stronger Than Pride	The New Sta
Herbie Hancock	Manhattan	The New Sta
Herbie Hancock	Mercy Street	The New Sta
Herbie Hancock	New York Minute	The New Sta

Record 1 of 25

Standard Filter

Criteria

Operator	Field name	Condition	Value
	Album	like	'%Speak%'
AND	- none -		
AND	- none -		

that opens, I select *Album* from the *Field name* drop-down list and *like* from the *Condition* drop-down list. Note: Using the *like* condition allows the use of wildcards to match partial strings in the *Value* field.

Next I enter the artist name in the *Value* text box. The *Value* string is case sensitive and requires exact spelling. Again I use the SQL wildcard percent (%) character (e.g., *%Speak%)*, in place of the full album name *Speak Like A Child*. Other filter conditions can be applied to either the *Artist*, *Song*, or *Album* fields.

Filter Result

As shown from the six song tracks in Figure 13, the standard filter narrowed the album information to one album, *Speak Like A Child*.

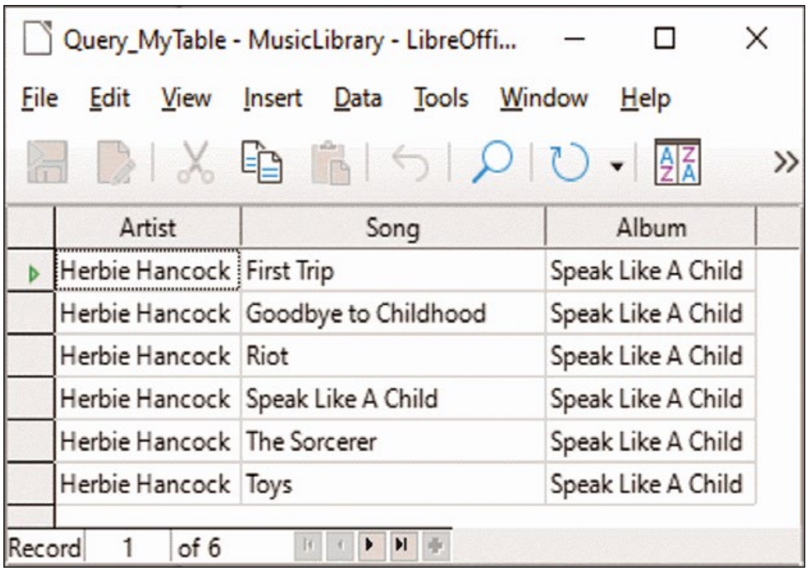
This was a simple example to illustrate that by following the process I've described, LibreOffice can easily be used to migrate music information from iTunes into a searchable database and perform simple to moderately complex database queries.

Summary

To facilitate the database creation and query process for users that may have minimal familiarity with SQL, I used wizards to create a database table and SQL query. More sophisticated SQL users could write more complex queries and table relationships. Those features are available in LibreOffice Base.

Although I used the LibreOffice Base embedded database engine HSQLDB, Base can also connect to other popular SQL databases such as MySQL, MariaDB, and PostgreSQL, to name a few.

For users who need to create and manage one or more databases for personal, educational, or home office use, LibreOffice is faster to implement, simpler, and has a built-in user interface. LibreOffice achieves the same goals as the more complicated MySQL for simple to moderately complex database queries. ∎∎∎


Query_MyTable - MusicLibrary - LibreOffi...
File Edit View Insert Data Tools Window Help

Artist	Song	Album
Herbie Hancock	First Trip	Speak Like A Child
Herbie Hancock	Goodbye to Childhood	Speak Like A Child
Herbie Hancock	Riot	Speak Like A Child
Herbie Hancock	Speak Like A Child	Speak Like A Child
Herbie Hancock	The Sorcerer	Speak Like A Child
Herbie Hancock	Toys	Speak Like A Child

Record 1 of 6

Figure 13: Using the standard filter to narrow results down to one album.

The Author

John Cofield is a retired software marketing manager in Northern California. His training is in electrical engineering, and he has worked at multiple Silicon Valley semiconductor and software companies. His non-technical interests include Jazz music, ranging from Modal to Fusion.

FOSSPicks

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

Graham is currently hooked on using tiling window managers on KDE Plasma, using ALT left-click and ALT right-click to move and resize windows without needing to remember keyboard shortcuts. BY GRAHAM MORRISON

Software synthesizer

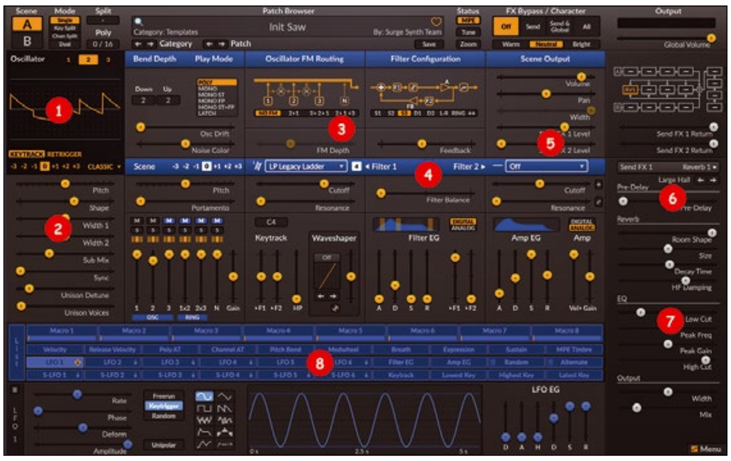
Surge XT

We've looked at previous versions of the Surge software synthesizer before, simply because it's been developing, changing, and improving at an incredible pace. This release is no different, with the project even adding the "XT" letters to its name to show the significance of the update. The synth itself is one of the best sounding pieces of software you can install, proprietary or otherwise, with only the equally amazing Vital able to challenge it for sound complexity and quality. But Vital, being a "spectral warping wavetable synthesizer," is quite esoteric both in the way it sounds and in the way you coax sounds out of it.

Surge is just as capable, and sometimes just as complex, but the audio path and signal flow are more traditional and easier to follow. Sound starts in the top left of the window, where a small panel lets you tab between three oscillators. The oscillators are based on the classic sawtooth, square, triangle, and sine shapes, but they can all be tweaked and morphed using the shape, width, and submix sliders in a lower panel. As soon as you move a slider, the waveform display updates to show the new shape and, of course, you can hear the results immediately if you play your computer keyboard or MIDI device. The amplitude of each of these oscillators is mixed into the patch using another panel of vertical sliders to the right of the oscillators, and this includes additional sliders for noise and two sets for "oscillator multiplication" to create robotic and metallic timbres.

To the right of these sections are the filters which are used to impart character on the raw oscillator sound, either by removing harmonics or by emphasizing them. Surge has two wonderful filters that can be configured to run in serial or parallel and in other configurations, according to a very neat adjustable flow chart. The same flow chart approach is also used for frequency modulation, and it's a brilliant way to visualize what is often difficult to understand and hidden from the user in other synths. The filters themselves sound amazing, and there's a large selection to choose between, including those that sound like the classics from Moog, ARP, and Oberheim, and modern hybrids too.

Modulation, where one signal changes the value of another, is another key element in synthesizer design because it's used to generate both sounds, such as with the ring modulation effect and frequency modulation, and to change the values of parameters in a sound over time. The middle and lower parts of the main window are dedicated to a matrix of modulation sources and destinations, plus a variety of LFOs, sequencers, and envelopes. Almost anything can be set to change anything else, letting you create hugely complex patches in the same way you can with a modular synthesizer. But unlike modules, Surge is polyphonic, supports MIDI Polyphonic Expression (MPE), and runs as an LV2 plugin or as a VST. And we've not even mentioned the effects that can be added to the sound output, which include beautiful reverbs and delays, a vocoder, tape simulator, distortion, flangers, chorus, and even a Nimbus effect, which is itself heavily inspired by the open source Mutable Instruments Clouds granular effect Eurorack module. It sounds amazing.



1. Oscillators: These generate the sound and are modeled on the waveforms found in classic synths. **2. Editor:** Changes almost everything about an oscillator, filter, envelope, or the mix. **3. Routing:** Handy flow charts show the signal path and how it can be adjusted. **4. Filter:** Two filters can be used at the same time and, like the oscillators, sound wonderful. **5. Output:** Mix effects and left and right channel panning. **6. Effects:** Serge includes some amazing effects with an unprecedented amount of control over how they're applied. **7. Presets:** Load up an entire sound or even presets for specific chains of effects. **8. Modulation:** Dynamically change the value of almost anything you can see in this screenshot.

Project Website
https://github.com/surge-synthesizer/surge

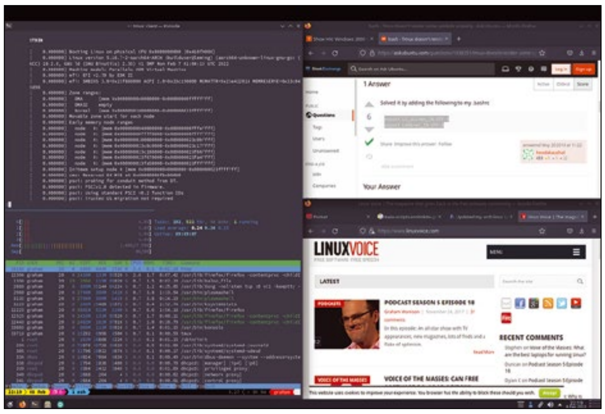
Tiling window script

Kröhnkite

In popular Linux culture, tiling window managers are the domain of the über geeks. This is because they look austere, without any kind of window control or decoration, and often lack the pleasantries of a settings application or launch panel (the tiling window managers, not the uber geeks). Tiling window managers are also unforgiving, requiring manual configuration and serious time investment to fully purge users of evil mouse clicks before promptly filling empty brain matter with keyboard shortcuts.

Things are changing, however, and their minimalism and utility is beginning to invade the traditional desktop space where many of their best features are starting to coexist with the more traditional desktop environment. This is definitely a good thing because tiling window managers are quick, efficient, and often make best use of whatever screen real estate is available, and a lot of this innovation is happening on KDE Plasma. This is because it has a window scripting engine that can be harnessed to reconfigure window positions without replacing the window manager. This means you get to pick which parts of KDE to keep and which parts of a tiling window manager you want to use.

Plasma has always had relatively good tiling solutions, with the previously covered kwin-tiling script proving popular. This humble script takes over positioning duties and optionally removes window decoration, letting you switch between common layouts and window orders with tiling-friendly shortcuts. But there's now a new breed of tiling solutions for Plasma that build on the success of kwin-tiling, and Kröhnkite is one. Kröhnkite supports Plasma's activities, virtual desktops, and multiple screens, which is something kwin-tiling struggles with. It also takes its inspiration from dwm, the dynamic window manager, with monocle, spread, stair, and standard tiling layouts. It works well but still relies on manually editing configuration files for the best experience, which leaves room for another perhaps easier-to-use option for Plasma.

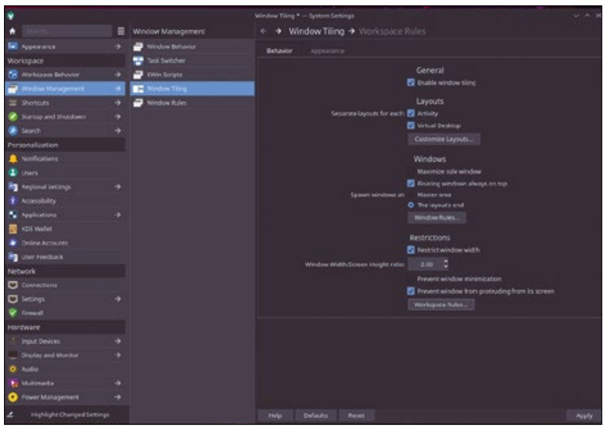

Get the benefits of both KDE Plasma and a tiling window manager with Kröhnkite.

Project Website
https://github.com/esjeon/krohnkite/

Window arranger

Bismuth

Bismuth is another tiling window manager for KDE Plasma (see Kröhnkite above). It does the same thing but with a little added refinement. In particular, it offers a convenient configuration pane as part of KDE's standard system settings application, and feels much more planned and better integrated. It's probably how Plasma would add tiling natively, especially as configuration panels are relatively rare in the world of tiling window managers. The setting page appears under the Window Management menu and lets you conveniently disable and enable tiling as well as customize tiling behavior. In particular, you can have separate layouts for different activities and virtual desktops, enable or disable specific tiling modes, choose where new panes will appear, set rules for specific window types (such as Yakuake always appearing on top without tiling), and change the appearance of the panels as they appear. It's powerful and easy to use. In many ways, this is a genuine iteration of the kwin-tiling script which offers many of the same options without making them accessible through a GUI. You're also not as burdened by keyboard shortcuts because there aren't so many, and the ones that are used are standard across several different tiling managers. If you panic and need to disable tiling quickly, there's even a panel icon you can simply click, and after a simple installation process, Bismuth quickly feels like a default part of KDE. This needs to be tempered slightly with Plasma itself, which will often only apply changes to new windows, and it can be difficult to get back to a coherent display after breaking a couple of windows out of the tiling environment or disabling Bismuth. But this is because it straddles both tiling and the desktop, and regardless of these blips, it succeeds impressively well.



General
☑ Enable window tiling

Layouts
Separate layouts for each: ☑ Activity ☑ Virtual Desktop
Customize Layouts...

Windows
☐ Maximize sole window
☑ Floating windows always on top
Spawn windows at: ○ Master area ● The layout's end
Window Rules...

Restrictions
☑ Restrict window width
Window Width/Screen Height ratio: 2.00
☐ Prevent window minimization
☑ Prevent window from protruding from its screen
Workspace Rules...

Alongside settings for which tiling modes are enabled where, Bismuth can also disable window decorations and change the game between panels.

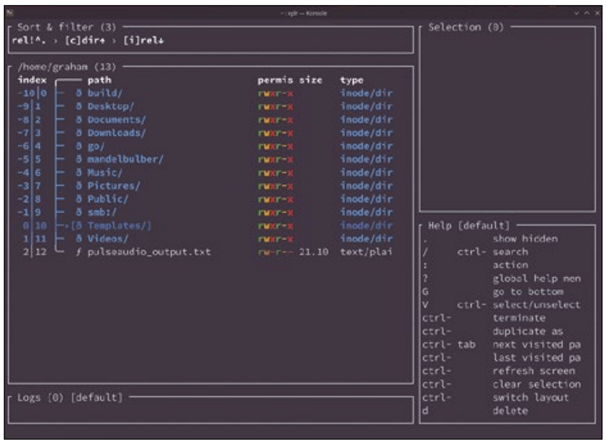
Project Website
https://github.com/Bismuth-Forge/bismuth

Terminal file explorer

xplr

Despite all the command-line file managers we've looked at, and the venerable Midnight Commander created by Miguel de Icaza, it still feels like there's space for improvement. xplr could be the tool that makes the difference, not because it's more powerful than Midnight Commander, but because it does file management differently and not in a way that tries to replicate a desktop file manager. xplr is primarily minimal and very fast. It will launch in about the same time it takes to peruse the output from the `ls -al` command, and the default view shows the same information. You can then use the cursor or Vim navigation keys to move up and down through the file and directory list, pressing right to enter a directory and left to leave for the parent directory. Other Vim bindings, such as the top and bottom, will also work, and there's also an equivalent command mode that's opened by pressing `:`.

But you don't need to be a Vim user, or have an incredible memory for shortcuts, to use xplr. A small pane in the bottom right shows all the important keyboard shortcuts, and pressing `?` will always show a complete list of which commands are available and what they do. What makes xplr powerful is that it easily lets you perform actions that might take longer on the command line. You can use the spacebar to select files and directories, for example, and selected items appear in a pane in the top right. You can then issue further commands to move (`m`) or copy (`c`) these to a different location. If you need more than this, such as a custom layout or new commands, it's also eminently hackable, with a plugin system, well-commented code, and even a hacking document to help you on your way.



```
┌ Sort & filter (3) ───────────────────────────────┐ ┌ Selection (0) ──────────┐
rel!^. › [c]dir+ › [i]rel↓
┌ /home/graham (13) ───────────────────────────────┐
 index ┌── path                     permis size  type
 -10|0  ├─ ð build/                  rwxr-x       inode/dir
  -9|1  ├─ ð Desktop/                rwxr-x       inode/dir
  -8|2  ├─ ð Documents/              rwxr-x       inode/dir
  -7|3  ├─ ð Downloads/              rwxr-x       inode/dir
  -6|4  ├─ ð go/                     rwxr-x       inode/dir
  -5|5  ├─ ð mandelbulber/           rwxr-x       inode/dir
  -4|6  ├─ ð Music/                  rwxr-x       inode/dir
  -3|7  ├─ ð Pictures/               rwxr-x       inode/dir
  -2|8  ├─ ð Public/                 rwxr-x       inode/dir
  -1|9  ├─ ð smb:/                   rwxr-x       inode/dir
   0|10 ├▸[ð Templates/]             rwxr-x       inode/dir
   1|11 ├─ ð Videos/                 rwxr-x       inode/dir
   2|12 └─ ƒ pulseaudio_output.txt   rw-r-- 21.10 text/plai
                                                      ┌ Help [default] ─────────
                                                      .           show hidden
                                                      /     ctrl- search
                                                      :           action
                                                      ?           global help men
                                                      G           go to bottom
                                                      V     ctrl- select/unselect
                                                      ctrl-       terminate
                                                      ctrl-       duplicate as
                                                      ctrl- tab   next visited pa
                                                      ctrl-       last visited pa
                                                      ctrl-       refresh screen
                                                      ctrl-       clear selection
                                                      ctrl-       switch layout
┌ Logs (0) [default] ──────────────────────────────┐  d           delete
```

Custom file options can be set to change the color of specific files in the view.

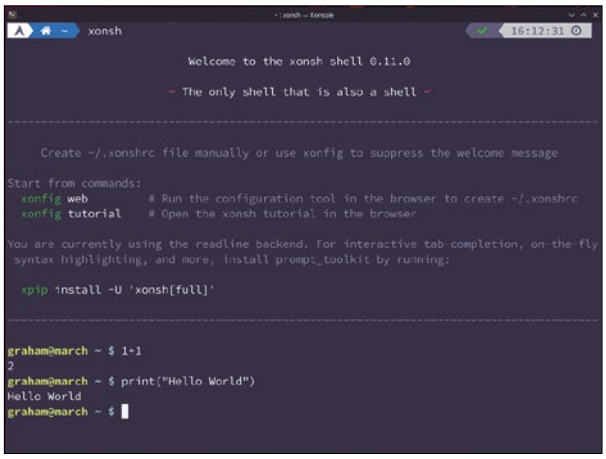
Project Website
https://xplr.dev

Python shell

xonsh

Despite most distributions defaulting to use the Bash shell, we all know there is a good selection of alternatives. There's the original sh, of course, the geeky C shell (csh), Korn shell (ksh), and the increasingly popular Z shell (Zsh). Each one of these is well established and likely to be a simple configuration option or package install away. Xonsh, however, will need to be installed manually, usually as a Python `pip` package, although there's an AppImage available too. And installing via `pip` is a clue to why you might want to install xonsh in the first place, because it's a shell that's going to be of most use to Python developers and users.

Xonsh is a little like launching the Python interactive interpreter and still being able to run regular shell commands. It lets you easily execute all the commands you might expect, such as `ls`, `mv`, and `echo`, within a Python environment. As a Python programmer, you can type `3.14*3.14` to perform a calculation or even start to write code. You can import external modules and use literal data types and multiline input. Python environment variables take the same format as those for a normal shell environment, which is useful for doing programmatic evaluation, and you can use pipes too, just like with Bash. There's a configuration file you can populate with your own options, and the prompt can be customized just like other shell environments. If you spend most of your day programming in Python, using a shell environment where this can be extended to the way you interact with the operating system makes a lot of sense, especially as Python can be a lot more opinionated about what it permits and processes. Even if you're not (yet) a Python programmer, using xonsh might be a brilliant way to learn.



```
xonsh                                                                16:12:31

                       Welcome to the xonsh shell 0.11.0

                     ~ The only shell that is also a shell ~

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

    Create ~/.xonshrc file manually or use xonfig to suppress the welcome message

Start from commands:
  xonfig web          # Run the configuration tool in the browser to create ~/.xonshrc
  xonfig tutorial     # Open the xonsh tutorial in the browser

You are currently using the readline backend. For interactive tab-completion, on-the-fly
 syntax highlighting, and more, install prompt_toolkit by running:

  xpip install -U 'xonsh[full]'

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

graham@march ~ $ 1+1
2
graham@march ~ $ print("Hello World")
Hello World
graham@march ~ $ █
```

Xonsh offers what it calls a superset of Python 3.6+ commands, including normal shell commands and environment variables.

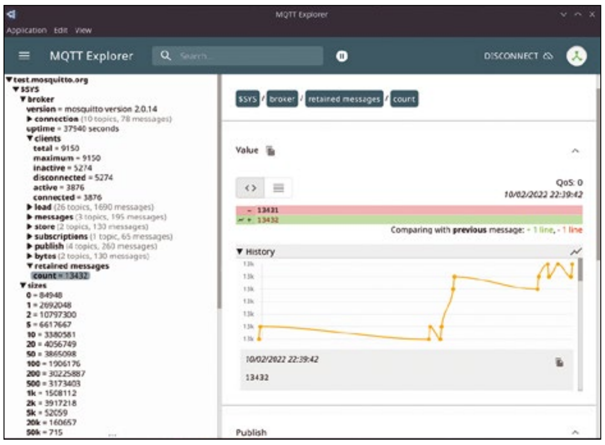
Project Website
https://xon.sh/

MQTT client

MQTT Explorer

MQTT is one of those things that initially sounds complicated but is actually quite easy to understand – and profoundly powerful when you know how to use it. The project doesn't exactly do itself any favors, either, describing itself as "an OASIS standard messaging protocol for the Internet of Things (IoT)." Put simply, however, MQTT is an easy to understand protocol and architecture for sending messages, and one that can be easily used from almost any device to send updates to another. You might have a USB temperature sensor connected to a Raspberry Pi in your garage, for example, and you'd like to share its data with your Node-RED server in your home. This is where MQTT can help. A simple command-line client tool called `mosquitto_pub` can transform the output from your sensor into an MQTT message and publish this to a local client, launched with another command called `mosquitto_sub`. The result is the aggregation of all MQTT messages which can then be connected to an MQTT broker, such as Node-RED.

One interesting aspect to MQTT is that the receiving clients don't know what each receiving message contains, and this is where MQTT Explorer can help. MQTT messages themselves are structured and easily parsed thanks to their native JSON support, and MQTT is a beautifully designed client that can unfold these messages and help you explore their contents and track their changing values. You simply enter the details of the publishing server you wish to connect to, and the incoming messages are aggregated and listed in the main window. Unfolding these will show their values in the pane on the right, along with a histogram of changes and other viewing options. It's a great way to demystify MQTT and pull the parts you need into other software.



```
▼test.mosquitto.org
  ▼$SYS
    ▼broker
      version = mosquitto version 2.0.14
      ▶connection (10 topics, 78 messages)
      uptime = 37940 seconds
      ▼clients
        total = 9150
        maximum = 9150
        inactive = 5274
        disconnected = 5274
        active = 3876
        connected = 3876
      ▶load (26 topics, 1690 messages)
      ▶messages (3 topics, 195 messages)
      ▶store (2 topics, 130 messages)
      ▶subscriptions (1 topic, 65 messages)
      ▶publish (4 topics, 260 messages)
      ▶bytes (2 topics, 130 messages)
      ▼retained messages
        count = 13432
    ▼sizes
      0 = 84948
      1 = 2692048
      2 = 10797300
      5 = 6617667
      10 = 3380581
      20 = 4056749
      50 = 3865098
      100 = 1906176
      200 = 30225887
      500 = 3173403
      1k = 1508112
      2k = 3917218
      5k = 52059
      20k = 160657
      50k = 715
```

```
QoS: 0
10/02/2022 22:39:42
- 13431
+ 13432
Comparing with previous message: + 1 line, - 1 line

10/02/2022 22:39:42
13432
```

MQTT is perfect for home automation, and MQTT Explorer can help you get the most from your sensor data.

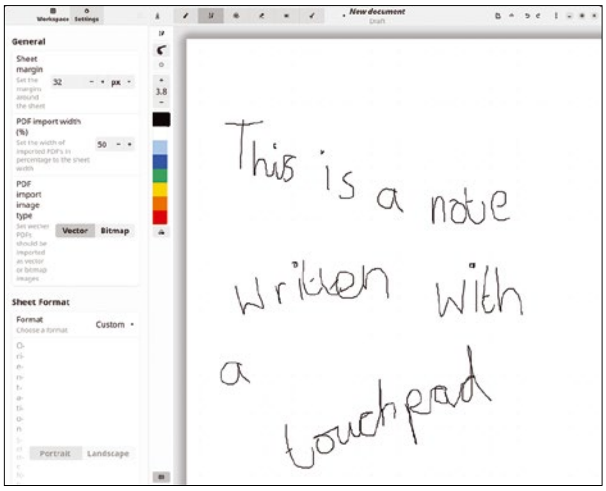
Project Website
http://mqtt-explorer.com

Note taking

Rnote

If you've used a macOS or iOS device recently, you couldn't have helped but notice that these platforms have a proliferation of note-taking applications. This could be because the iPad works with an optional pencil and is ideal for making written notes in an educational setting, but it could also be how users of those systems like to work. Either way, these note-taking applications typically look fabulous and will offer writing and PDF annotation, letting you scrawl your own notes around a document, or draw your own diagrams. But despair not, Linux users. Rnote is the closest we've seen to an open source Linux application come to feeling like one of those applications, and it does a wonderful job of capturing the clean, minimal aesthetic, while still offering much of the same functionality.

Much of the design credit goes to GTK4 because Rnote takes full advantage of this rapidly advancing toolkit. There are rounded corners, smoothly animated transitions, icons in the top bar, and beautifully rendered fonts. The main view is the drawing area, which is obviously a lot easier to use if your device supports touch input or if you use a stylus of some kind. But even without these, there are the usual drawing tools you might expect, including lines, rough and smooth shapes, and both solid and textured brush strokes. There are also several selection modes for modifying your doodles, and you can even turn on "drawing sounds" to add a little extra authenticity. You can then import an image or PDF to draw over and save the whole mess as an `.rnote` file, although you can also export as an SVG or PDF file, or even as a file for the competing Xournal++ note-taking application. The only big missing feature is the ability to add text fields, but that's being worked on.



Rnote is a beautifully minimal note-taking application that takes full advantage of GTK4.

Project Website
https://github.com/flxzt/rnote

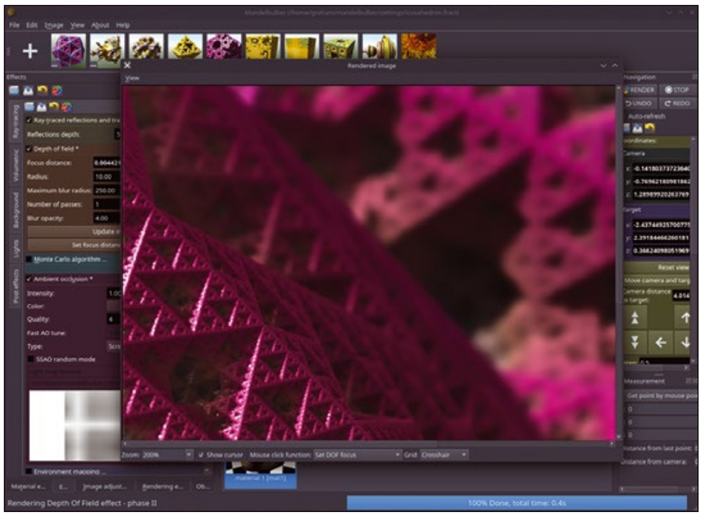
Fractal explorer

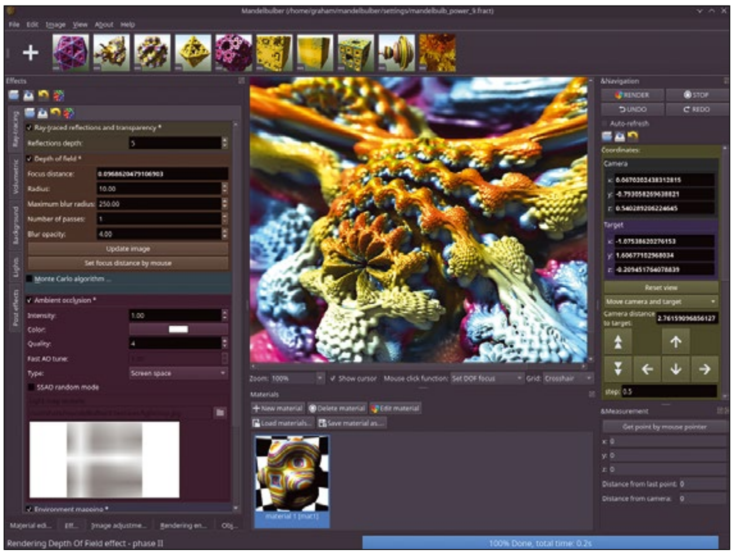
Mandelbulber v2

Fractals are amazing. Even decades after their initial popularity, when contemporary fads such as chaos theory and the butterfly effect are barely mentioned, the world of visualizing infinite complexity remains fascinating. In the 1980s and 1990s, home computers were able to open a window on this world for the first time, with fractal generators running on Amigas and early PCs capable of rendering an image over a period of minutes or hours, depending on the formula and the number of iterations you needed to calculate. They were even used in early games, generating the mountains in *Rescue On Fractalus!* and the caves in *The Eidolon*, both published by Lucasfilm Games. And the most famous of those fractal formulas was the Mandelbrot Set, whose spirals, galaxies, and curves became synonymous with fractal images.

There was once a plethora of fractal generating software, but it's fallen a little out of favor in recent years. Fortunately, however, a couple of the best applications have survived, and Mandelbulber is one of these. But it's a fractal explorer with an additional dimension, because the Mandelbulbs of its name are three-dimensional fractals derived from the original Mandelbrot set. These 3D models are quite unlike the original 2D Mandelbrot images, not least because they can be rendered fully textured, with ray tracing, depth of field effects, fog, and anti-aliasing. You can add shadows, transparency, and refraction, and materials can include luminosity, diffusion, normal maps, and displacement. While in old fractal programs you could only zoom in (or add iterations), and move up, down, left, and right, a 3D fractal is literally multidimensional. You can zoom in on but also around and into the 3D environment, with parts of the fractal often protruding in front of others.

There are a huge number of fractal presets to choose between and all can be edited, tweaked, and saved as their own presets. The main Qt 5 UI helps you navigate with buttons, cursor keys, or the mouse when it's in movement mode. The mouse can also act as a 3D cursor for choosing elements in the output render to use as the focus point, for example, or to select the area you want to explore further. There are almost as many rendering options as there are in Blender, including a panel for editing materials, a panel for effects, a panel for output rendering (including the ability to generate stereoscopic output), and a panel for exporting your view for use in other 3D applications. Most of the time you don't have to because the internal materials, ray tracing, and depth-of-field effects look amazing and can render these sci-fi worlds with a level of photorealism impossible just a decade ago. If you have the hardware, this can be accelerated with OpenCL, but even without the rendering it is quick until you get too far into the caves and creases of your fractal. But even then, you can farm out your processing to a distributed networking renderer. Mandelbulber really is fractal exploration for the 21st century.


Exploring 3D Mandelbulbs with Mandelbulber feels similar to using the Esper photo analyzer in *Blade Runner*.


Create cyber-complex, multi-dimensional images with unlimited resolutions, customizable materials, and full ray tracing.

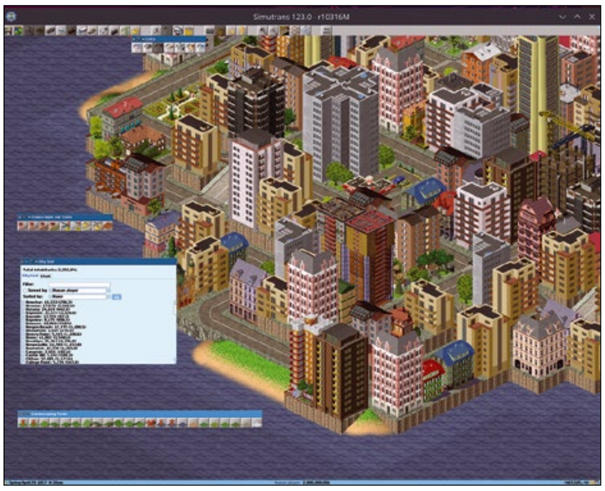
Project Website
https://github.com/buddhi1980/mandelbulber2

Transport simulator

Simutrans

As children, many of us enjoyed playing with toy railway tracks or putting together our own electricity-powered speedways. And there really isn't a good reason why we can't continue to enjoy these distractions as we get older, especially if some of that same enjoyment can be recreated in software. This is what *Simutrans* does, only in a way that feels more like *SimCity 2000* than model railways. It's a transport management game set within an environment that looks like an old version of *SimCity 2000*. "Old" is the keyword, too, because you can set which year you'd like to start in, which obviously influences the kinds of transport technology you'll be able to use. From coal-fired to the future. Unlike *SimCity*, however, you start the game by generating a random map which already contains a city or two, usually of epic proportions, and it's going to be your job to solve all their transport needs.

You typically start by linking up suburban areas with some form of transport, and adding any form of transport means clicking on an appropriate tool palette. These palettes hold all the infrastructure you can access for each mode of transport you can access. For railways, for example, you select an appropriate rail and draw a line across your map. But the palette also includes bridges, crossing signals, and many other elements, and other modes of transport, too. Each item has a cost and obviously adds to the complexity of your transport system, but it's a lot like having the world's biggest model railway at your fingertips, and it quickly becomes hugely complicated. If you want something simpler and more immediate, an integrated panel can download and run a preset scenario from which you can build on to solve a transport problem. This is a great mission-oriented way to get started, without the complexity of the sandbox mode overwhelming you, and makes the game a lot of fun even when time is limited.


Like SuperTux, Simutrans can also be downloaded from the Steam store, which is perfect if you happen to have gotten hold of a Steam Deck.

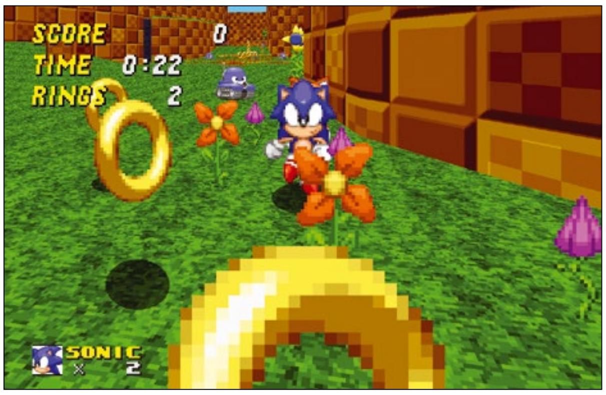
Project Website
https://www.simutrans.com

3D Sonic

Sonic Robo Blast 2

Sonic the Hedgehog appears quite often in open source gaming remakes. This must be because Sonic was a mainstay of 1990s console gaming, a decade that saw the creation of Linux, and is long enough ago to make teenagers of that time senior developers today. *Sonic Robo Blast 2* is a perfect proof of this theory, combining not only open source with Sonic, but also with another behemoth of the era, *Doom*. The game itself is entirely fan-made and even has its own roots back in the late 1990s. It features over 25 levels that are entirely created by the community and feature the same manic, sugar-fueled platforming style of the originals, but with one important difference. The game play has been transposed into three dimensions. And not just the tidy polygonal three dimensions of *Super Mario 64* either. These are the pseudo three dimensions of the crude textures and large sprites of *Doom*, or more accurately, the updated Doom Legacy engine.

If you've ever played the 2D original, then the gameplay will be familiar. There's a significant introduction sequence that provides a chunk of backstory, complete with cartoon-style graphics, before you're launched into the game proper. You get to play as any of the main characters, each with their own abilities, and your mission remains the same – collecting rings while traversing each zone as quickly as you can. You still get to tackle many of the same enemies, challenges, and hurdles as the original, only now with the ability to move in all directions. It might sound like an unnatural fit for two different styles of game, but *Sonic Robo Blast 2* plays remarkably like the original and with similarly high-quality production values. Before long, you'll be riding lifts, clinging on to out-of-control mining carts, and flying across canyons like it's 1992.



This piece of amazing fan fiction (of the gaming variety) has been in development for almost as long as its inspiration, but it's still getting updates.

Project Website
https://www.srb2.org/

Use Inkscape extensions to create 3D objects from paper

Paper Tiger

Papercraft is coming back into fashion. Linux users can turn to Inkscape and plugins such as Boxes.py, Paperfold, and Tabgen to create templates from 3D objects for printing.

BY SIRKO KEMTER

Papercraft is an old form of crafting. It requires nothing more than scissors, paper, and a little glue. Architects use the technique to visualize models quickly and inexpensively, despite digital developments such as virtual reality. Driven by trends such as low-poly modeling and Minecraft, the technique is also experiencing a revival in the hobby sector. In some areas of the world – such as Japan, which has traditionally cultivated origami, another papercraft – there are now many creative people posting their work on the various social media platforms. In Japan, this form is called pepakura, and there is also a software tool of the same name for it.

Unfolding objects and displaying the individual sides as faces is also the basis for other crafts – they usually just require a different node definition for the individual nodes. The templates you need for this can be created with Inkscape in combination with extensions such as Boxes.py [1], Quickjoint [2], Lasercut tabbed box [3], and Paperfold [4]. The tools prepare the objects in such a way that they can be further processed in Joinery [5] without any problems.

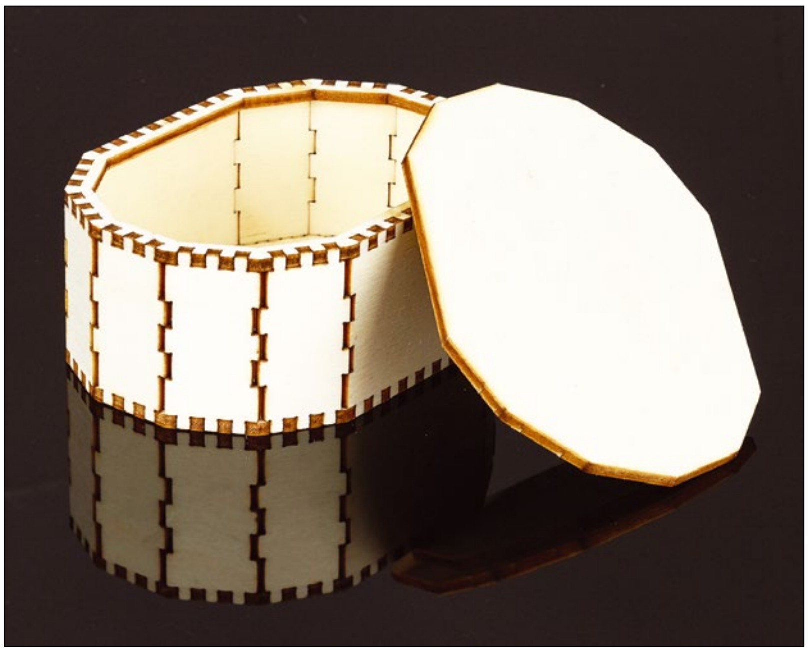


Figure 1: On Boxes.py, an active community collects numerous templates for building 3D objects. © Boxes.py

More Than Just Boxes

Boxes.py is very interesting here, particularly because it comes with a variety of predefined objects – and by no means just simple boxes (Figure 1). For example, rendering a Raspberry Pi case is child's play. But more complex designs with curves and similar features are also available in Boxes.py, you just have to adapt them to your own needs. Pepakura [6] and other software tools for this work are only available for Windows, which forces Linux users to find other ways to indulge in their hobbies.

One option from the world of free software for designing 3D objects is Blender. Of course, it can do far more than is required for the task. For Papercraft, the 3D objects also need to be unfolded. This step is handled by a flattener, and there is even an add-in for Blender to match. But if you want to print the unfolded object later on and perhaps also decorate it with graphics, a program like Inkscape is more suitable.

There are many extensions for Inkscape for this kind of work. Often, however, these are not coordinated with the other extensions because each author has their own way of working or view of how to do things. One of the most important extensions of this kind is Paperfold. There is another extension from the same developer with similar functions, but the author considers Paperfold to be the better choice.

Unfolding with Paperfold

As usual with Inkscape, the Paperfold extension is installed by copying the appropriate files either to

`~/.config/inkscape/extensions/` or to `/usr/share/inkscape/extensions/` for a global installation. Download the `paperfold_1zSsmQj.zip` file by pressing the down arrow on the project's website. For Paperfold to work, it requires OpenMesh and the appropriate Python bindings. You can use the commands in Listing 1 to install the packages on a Fedora system.

CMake and the GNU C++ compiler are required because the following commands not only download the appropriate bindings, but also adapt them to the installed version. The two `pip3` commands from Listing 1 also work on other distributions. You will need to modify the first command there. After restarting Inkscape, Paperfold is available in *Extensions* | *FabLab Chemnitz* | *Papercraft Flatteners* | *Paperfold* (Figure 2).

To get started, it is best to choose a simple object because it takes a while to develop a feel for objects spread out onto surfaces. Paperfold expects either an OFF, OBJ, PLY, or STL file as the initial object. The number of allowed faces has to be adjusted in most cases. However, you should not set the value too high because this will push up your memory consumption. All other values can be safely enabled. They will help you later when you go about gluing the model.

The rendered object consists of grouped single objects, so you have to ungroup for editing. After ungrouping the first group, you will see more groups: One contains all the label elements, the other the paths. After ungrouping again, they too can be edited.

The author has thought of post processing using Joinery; this is hidden away in the *Post Processing* tab. But papercraft and the required adhesive tabs were not considered. If required, you need to add them to the right edges manually or use another extension, such as Tabgen [7].

Adhesive Tabs with Tabgen

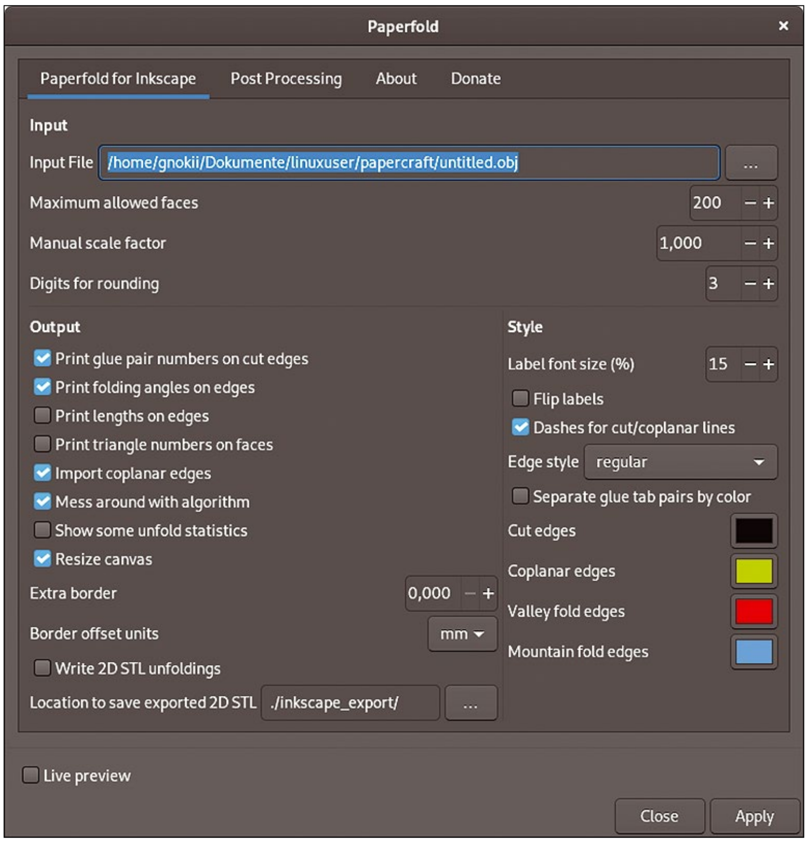
Installing Tabgen is much like installing Paperfold. The next time you launch Inkscape, the plugin is available in *Extensions* | *Papercraft* | *Tabgen* (Figure 3). But some preliminary work is needed to be able to apply Tabgen to objects created with Paperfold. The two tools work in different ways. Tabgen expects closed paths, while Paperfold creates each page as a single path. You'll need to create an appropriate closed path object using Inkscape's tools – not too tricky once you know how.

To use the *Edit* | *Select Same* function, you first need to select the individual paths. This depends on whether the numbering was rendered in Paperfold. If so, you have to change its appearance a little before you can select the paths of the edges with this function. To do this, select one of the circles around the numbering and use the *Edit* | *Select Same* | *Fill and Outline* function. Because the paths of the edges have no fill, they are dropped from the selection.

Once you have selected all the circles of the numbering, simply change the color of the contour by selecting a different color while holding

Listing 1: Installing OpenMesh

```
$ sudo dnf install OpenMesh cmake g++ python3-devel
$ pip3 install pybind11
$ pip3 install openmesh
```



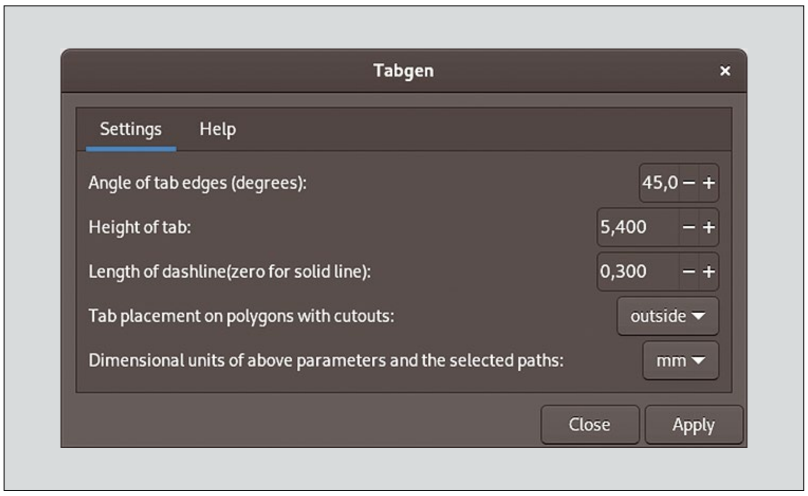
Input
Input File /home/gnokii/Dokumente/linuxuser/papercraft/untitled.obj ...
Maximum allowed faces 200
Manual scale factor 1,000
Digits for rounding 3

Output
✓ Print glue pair numbers on cut edges
✓ Print folding angles on edges
☐ Print lengths on edges
☐ Print triangle numbers on faces
✓ Import coplanar edges
✓ Mess around with algorithm
☐ Show some unfold statistics
✓ Resize canvas
Extra border 0,000
Border offset units mm
☐ Write 2D STL unfoldings
Location to save exported 2D STL ./inkscape_export/ ...

Style
Label font size (%) 15
☐ Flip labels
✓ Dashes for cut/coplanar lines
Edge style regular
☐ Separate glue tab pairs by color
Cut edges ⬛
Coplanar edges 🟨
Valley fold edges 🟥
Mountain fold edges 🟦

☐ Live preview

Figure 2: Paperfold unfolds 3D objects to create a flat image.



Angle of tab edges (degrees): 45,0
Height of tab: 5,400
Length of dashline(zero for solid line): 0,300
Tab placement on polygons with cutouts: outside
Dimensional units of above parameters and the selected paths: mm

Figure 3: Tabgen adds glue tabs to the flattened object.

down the Shift key. After that you can select an edge and execute the function *Edit* | *Select Same* | *Contour*, which tells Inkscape to include all the paths of the edge in the selection.

Since the following steps are difficult to undo, create a duplicate of all the selected paths with Ctrl+D. After that, apply *Path* | *Stroke to Path* and then *Path* | *Combine*. Now you should be able to apply Tabgen to the object.

Doing so renders another path object, enlarging it by the dimensions chosen in the extension's settings for the glue tabs (Figure 4). But there is a small shortcoming here as well: The developer of this extension seems to glue every single side of an object and never fold it. So you get glue folds even where you don't need them and have to delete them manually from the object.

The developer needs to put in a little more work to make the extension useful for a wider audience. In addition, the tool shows some errors, due to the fact that the code does not catch any errors and the returned error messages are of little help even to experienced users. The developer is probably just getting into programming and creating these extensions for their own use. A little feedback, praise, and also help could probably go a long way here.

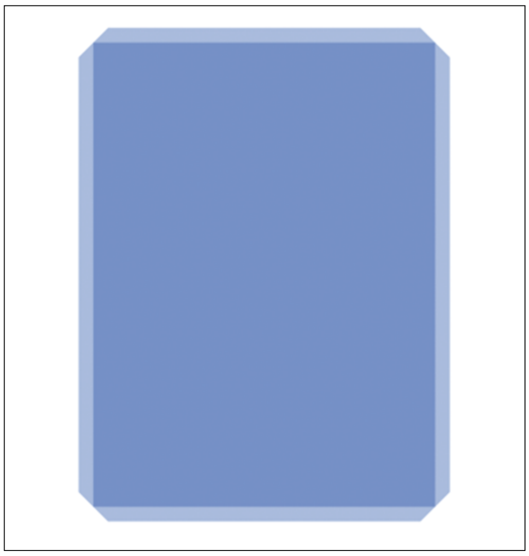


Figure 4: One side of a cube: The tabs were created by Tabgen.

Conclusions

Papercraft may have been old hat some time ago, but with laser cutting and a little bit of creative thinking, it's currently gaining renewed popularity. With Inkscape and the extensions we looked at in this article, you can definitely indulge in this hobby and certainly have some fun, too. ■■■

Info

[1] Boxes.py: *https://www.festi.info/boxes.py/*

[2] Quickjoint: *https://inkscape.org/~Jarrett/%E2%98%85quickjoint*

[3] Lasercut tabbed box: *https://inkscape.org/~Neon22/%E2%98%85lasercut-tabbed-box*

[4] Paperfold: *https://inkscape.org/~MarioVoigt/%E2%98%85paperfold*

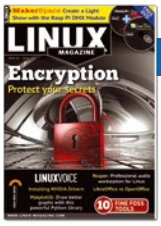
[5] Joinery: *https://clementzheng.info/Joinery*

[6] Pepakura: *https://tamasoft.co.jp/pepakura-en*

[7] Tabgen: *https://inkscape.org/de/~Shoshanaz/%E2%98%85tabgen-3d-papercraft-extension*

The Author

Sirko Kemter has been using Inkscape since the project's inception and loves discovering the new options that this software offers.



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 *info@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.

Linux App Summit

Date: April 29-30, 2022

Location: Rovereto, Italy and Online

Website: *https://linuxappsummit.org/*

The Linux App Summit (LAS) brings the global Linux community together to learn, collaborate, and help grow the Linux application ecosystem. Through talks, panels, and Q&A sessions, we encourage attendees to share ideas, make connections, and join our goal of building a common app ecosystem.

ISC High Performance

Date: May 29-June 2, 2022

Location: Hamburg, Germany

Website: *https://www.isc-hpc.com/*

ISC is the event for high performance computing, machine learning, and data analytics. ISC brings together HPC practitioners, users, and vendors to engage in discussions on technologies that didn't seem possible a short time ago. This event will be hosted on-site and remotely. Be among the 3,000 attendees and 100 exhibitors for a brand-new technical program and exhibition.

Events

ODSC East 2022	April 19-21	Boston, MA + Virtual	https://odsc.com/boston/
LinuxFest Northwest 2022	April 22-24	Virtual Event	https://lfnw.org/conferences/2022
DrupalCon Portland 2022	April 25-28	Portland, Oregon	https://events.drupal.org/portland2022
DeveloperWeek Europe 2022	April 27-28	Virtual Event	https://www.developerweek.com/europe?utm_source=LMcalendar
Linux App Summit	April 29-30	Rovereto, Italy + Virtual	https://linuxappsummit.org/
Linux Storage, Filesystem, MM & BPF Summit	May 2-4	Palm Springs, California	https://events.linuxfoundation.org/lsfmm/
Cloud Expo Europe Frankfurt	May 11-12	Frankfurt, Germany	https://www.cloudexpoeurope.de/en
KubeCon + CloudNativeCon Europe 2022	May 16-20	Valencia, Spain	https://events.linuxfoundation.org/
ISC High Performance 2022	May 29-June 2	Hamburg, Germany	https://www.isc-hpc.com/
OpenJS World 2022	June 6-10	Austin, Texas	https://events.linuxfoundation.org/openjs-world/
cdCon	June 7-8	Austin, Texas + Virtual	https://events.linuxfoundation.org/cdcon/
ODSC Europe 2022	June 15-16	London, UK + Virtual	https://odsc.com/europe/
ITEXPO Florida	June 21-24	Fort Lauderdale, Florida	https://www.itexpo.com/east/
Open Source Summit North America	June 21-24	Austin, Texas + Virtual	https://events.linuxfoundation.org/
Xen Developer & Design Summit	June 28-30	Bucharest, Romania + Virtual	https://events.linuxfoundation.org/xen-summit/
USENIX ATC '22 & OSDI '22	July 11-13	Carlsbad, California	https://www.usenix.org/conference/
The Open Source Infrastructure Conference	July 19-20	Berlin, Germany	https://stackconf.eu/

CALL FOR PAPERS

We are always looking for good articles on Linux and the tools of the Linux environment. Although we will consider any topic, the following themes are of special interest:

- System administration
- Useful tips and tools
- Security, both news and techniques
- Product reviews, especially from real-world experience
- Community news and projects

If you have an idea, send a proposal with an outline, an estimate of the length, a description of your background, and contact information to *edit@linux-magazine.com*.

The technical level of the article should be consistent with what you normally read in *Linux Magazine*. Remember that *Linux Magazine* is read in many countries, and your article may be translated into one of our sister publications. Therefore, it is best to avoid using slang and idioms that might not be understood by all readers.

Be careful when referring to dates or events in the future. Many weeks could pass between your manuscript submission and the final copy reaching the reader's hands. When submitting proposals or manuscripts, please use a subject line in your email message that helps us identify your message as an article proposal. Screenshots and other supporting materials are always welcome.

Additional information is available at: *http://www.linux-magazine.com/contact/write_for_us.*

Authors

Zack Brown	11
Bruce Byfield	6, 32, 49
Joe Casad	3, 22
John Cofield	81
Jon "maddog" Hall	78
Sirko Kemter	92
Ralf Kirschner	54
Marina Köhn	18
Rubén Llorente	60
Christoph Meinel	14
Pete Metcalfe	72
Graham Morrison	86
Hartmut Noack	26
Dmitri Popov	40
Mike Schilli	44
John Schwartzman	64
Markus Stubbig	36
Ferdinand Thommes	79
Jack Wallen	8, 22

Contact Info

Editor in Chief
Joe Casad, jcasad@linux-magazine.com

Copy Editors
Amy Pettle, Aubrey Vaughn

News Editor
Jack Wallen

Editor Emerita Nomadica
Rita L Sooby

Managing Editor
Lori White

Localization & Translation
Ian Travis

Layout
Dena Friesen, Lori White

Cover Design
Dena Friesen

Cover Image
© frenta, 123RF.com

Advertising
Brian Osborn, bosborn@linuxnewmedia.com
phone +49 8093 7679420

Marketing Communications
Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
4840 Bob Billings Parkway, Ste 104
Lawrence, KS 66049 USA

Publisher
Brian Osborn

Customer Service / Subscription
For USA and Canada:
Email: cs@linuxpromagazine.com
Phone: 1-866-247-2802
(Toll Free from the US and Canada)

For all other countries:
Email: subs@linux-magazine.com

www.linuxpromagazine.com – North America
www.linux-magazine.com – Worldwide

Printed in Nuremberg, Germany by hofmann infocom GmbH.

Distributed by Seymour Distribution Ltd, United Kingdom

LINUX PRO MAGAZINE (ISSN 1752-9050) is published monthly by Linux New Media USA, LLC, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA. Periodicals Postage paid at Lawrence, KS and additional mailing offices. Ride-Along Enclosed. POSTMASTER: Please send address changes to Linux Pro Magazine, 4840 Bob Billings Parkway, Ste 104, Lawrence, KS 66049, USA.

Published monthly in Europe as Linux Magazine (ISSN 1471-5678) by: Sparkhaus Media GmbH, Bialasstr. 1a, 85625 Glonn, Germany.

Approximate	
UK / Europe	May 07
USA / Canada	June 03
Australia	July 04
On Sale Date	

Please note: On sale dates are approximate and may be delayed because of logistical issues.

Issue 259 / June 2022

Zero Trust Security

Who can you trust on your network? According to the Zero Trust security model, no one. Next month we take a close look at this powerful new vision of a network where no one is on the inside.

Preview Newsletter

The Linux Magazine Preview is a monthly email newsletter that gives you a sneak peek at the next issue, including links to articles posted online.

Sign up at: ***https://bit.ly/Linux-Update***

Image © skorzewiak, 123RF.com

TRANSFORMING
THE FUTURE

ISC IS BACK IN PERSON!

CONFERENCE

PARALLEL PROGRAMMING

SYSTEM ARCHITECTURE

WORKSHOPS

MACHINE LEARNING

EXHIBITION

QUANTUM COMPUTING

APPLICATIONS & ALGORITHMS

AND MORE...

TUTORIALS

Come join 3,000 high performance computing (HPC) practitioners, vendors, and enthusiasts.

WHAT TO EXPECT?

- Technical program (in person and online)
- HPC exhibition (in person and online)
- Reunion with peers, friends, and collaborators
- Safe and health-protocol-compliant environment

After two years of an online presence ISC returns to you as an in-person event. If you're involved or interested in HPC, don't miss out on making valuable connections at ISC 2022.

ISC High Performance
The HPC Event.

ISC 2022 | MAY 29 – JUNE 2, 2022
HAMBURG, GERMANY

isc-hpc.com