FREE DVD

ExaGear
Run x86 apps on Rasp Pi and other ARM systems

VirtualBox Hacks

LINUX PRO MAGAZINE

MARCH 2019

VirtualBox Hacks

Save time and extend the power of your virtual machines

Scapy
Automate packet analysis with Python

Put a Recording Studio on a Raspberry Pi

Elementary OS
Elegant Linux with an ambitious vision

Create a Cartoon
With open source tools

Tools for Writers
Find words and organize your thoughts

PDF Tricks
Clean up blotches and fading text

LINUXVOICE

- **Go For It!** Easy Tool for building ToDo lists
- **maddog:** Diligence and other FOSS resolutions for 2019

FOSSPicks
- eDEX-UI
- Auryo
- Qalculate!

Tutorials
- Bash Arrays
- Natron Video Tricks

Issue 220
Mar 2019
US$ 15.99
CAN$ 17.99

e.g. Hetzner Cloud Server CX11

- ✔ Intel® Xeon® Skylake
- ✔ 1 vCPU
- ✔ 2 GB RAM
- ✔ 20 GB NVMe SSD
- ✔ 20 TB traffic*
- ✔ Intuitive Cloud Console
- ✔ Free DDoS protection
- ✔ Located in Germany or Finland

monthly $ **2.83**

Hetzner Cloud with flexible SSD storage

Volumes offer highly available and reliable SSD block storage for your cloud servers. You can expand a Volume to 10 TB each and attach up 16 to a cloud server, so they're great for jobs with high storage requirements.

Our award-winning Hetzner Cloud combines high performance hardware with intuitive usability all at an incredibly low price.

Volumes monthly $ **4.55** per 100 GB

cloud.hetzner.com

* With 20 TB of included traffic, you'll have lots of bandwidth for your projects, regardless of which Hetzner Cloud package you choose. But if you need, you may add more for an extra $1.14 a month per TB.

BEFORE YOU SIGN

Dear Reader,

Noncompete agreements have been around for years, but they seem to be experiencing a surge in popularity. Employers love noncompete agreements because they reduce the power of the employee to bargain for higher wages. Interestingly, noncompete clauses should really be anathema to both so-called conservatives (because they interfere with competition) and so-called liberals (because they diminish the rights of workers), but they seem to have taken hold throughout the developed world, and most governments appear unwilling to confront the problem in a comprehensive way.

To be fair, noncompete agreements do have a purpose in this world. A company might make a huge investment in finding an upper-management exec, who is well compensated for any contract restrictions and whose departure could truly make chaos for corporate strategy. Sales staff sometimes have access to customer lists and other valuable information that the company has an interest in protecting from competitors. But the use of noncompete agreements has spread well beyond these special cases. A recent article in Bloomberg [1] quotes a 2014 study [2] that found nearly one in five workers in the US were bound by noncompete agreements, including in jobs such as camp counselors, night watchmen, and other gigs where the clause serves no constructive purpose other than to take power away.

Anyone can put a noncompete clause in a contract, but the enforcement of the clause depends on the jurisdiction. The different countries of the EU all have slightly different rules. In the US, different states have different statutes and common law provisions that define the enforceability of noncompete agreements. Most states have a general tolerance for noncompete agreements, although California, high-tech mecca of the USA, takes a hard line in prohibiting noncompetes, and a few other states have acted recently to limit the power of noncompete agreements, especially for hourly workers.

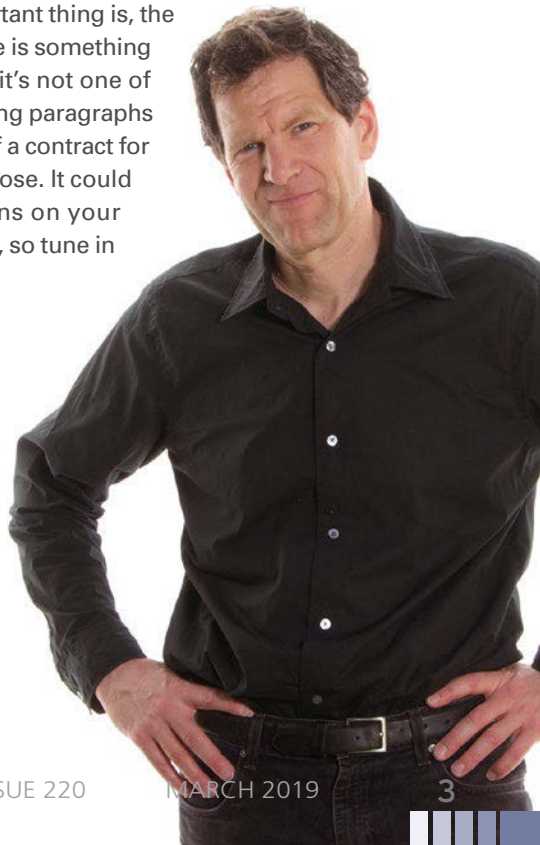
If you're on a career path in the IT industry, you might find it difficult to avoid noncompete agreements (unless you move to California). But if you're facing a contract negotiation, it is important to come armed with information, because information is power.

The first thing to know is that the noncompete clause in your contract might not be enforceable in your jurisdiction. (Check into this – don't take my word for it!) It is also a good idea to stop and consider who you are bargaining with. A big company like IBM or Oracle probably has a standard contract, hammered out by a team of lawyers and tailored precisely to the company's needs, and it will be difficult to get them to make a lot of changes just for little ol' you (although you could always try). A smaller company, however, might be using a boilerplate contract, with a few customizations by a local attorney, and they might be more flexible about changing the terms.

Some employers might balk at striking out the noncompete clause completely, but you might be able to talk them into reducing the duration or scope. For instance, if a clause tells me I can't go work for *another magazine*, I might be able to negotiate a refinement that specifies I won't work for *another Linux magazine*, and the employer might decide that their interests are adequately protected.

And of course, as any good negotiator will tell you, anything you give up should be worth something. If you're going to surrender the possibility of future professional mobility, you should let your future employer know that *you know* you are giving something up, which should be reflected somehow in the compensation.

This sounds like ivory tower stuff – easy for me to say and maybe not so easy to get an employer to accept. The important thing is, the non-complete clause is something to pay attention to – it's not one of those obscure trailing paragraphs stuck in at the end of a contract for no foreseeable purpose. It could have big implications on your future employment, so tune in before you sign.



Joe

Joe Casad,
Editor in Chief

Info

[1] "Too Many Workers Are Trapped by Non-Competes": *https://www.bloomberg.com/opinion/articles/2018-11-12/non-compete-clauses-trap-too-many-american-workers*

[2] "Noncompetes in the US Labor Force": *https://papers.ssrn.com/sol3/papers.cfm?abstract_id=2625714*

LINUX MAGAZINE

WHAT'S INSIDE

The FOSS tool known as VirtualBox is a popular alternative for easy virtualization in Linux. Many users are satisfied to spin up a virtual machine and start a system inside it, but VirtualBox can do so much more. We show you some advanced tricks with VirtualBox.

Also in this Month's issue:

- **Scapy** – This cool packet analysis app integrates easily with Python, which lets you build packet analysis into your homegrown admin scripts (page 42).
- **Programming Snapshot** – Mike Schilli explores the joys of building a homegrown progress bar (page 46).

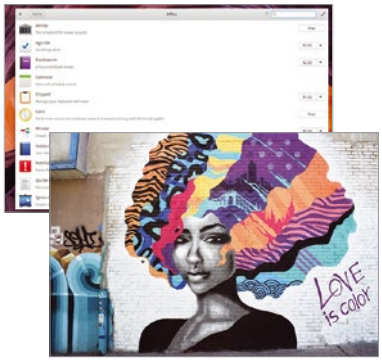
Check out MakerSpace to learn how you can use Waveform 9 to turn your Rasp Pi into a recording studio, and read on to LinuxVoice for a special article by Elvie cartoonist Mark Crutch on how you can create your own cartoon strip using open source tools.

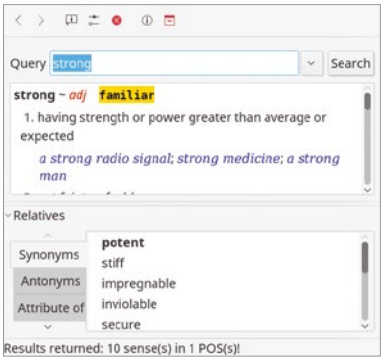
SERVICE

NEWS

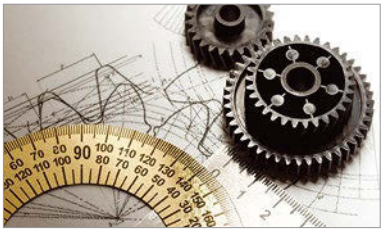
COVER STORIES

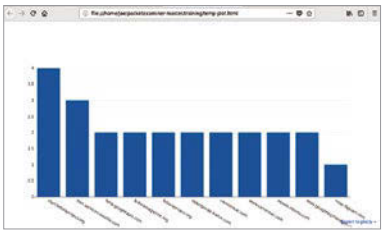
REVIEWS





IN-DEPTH





MAKERSPACE

LINUXVOICE

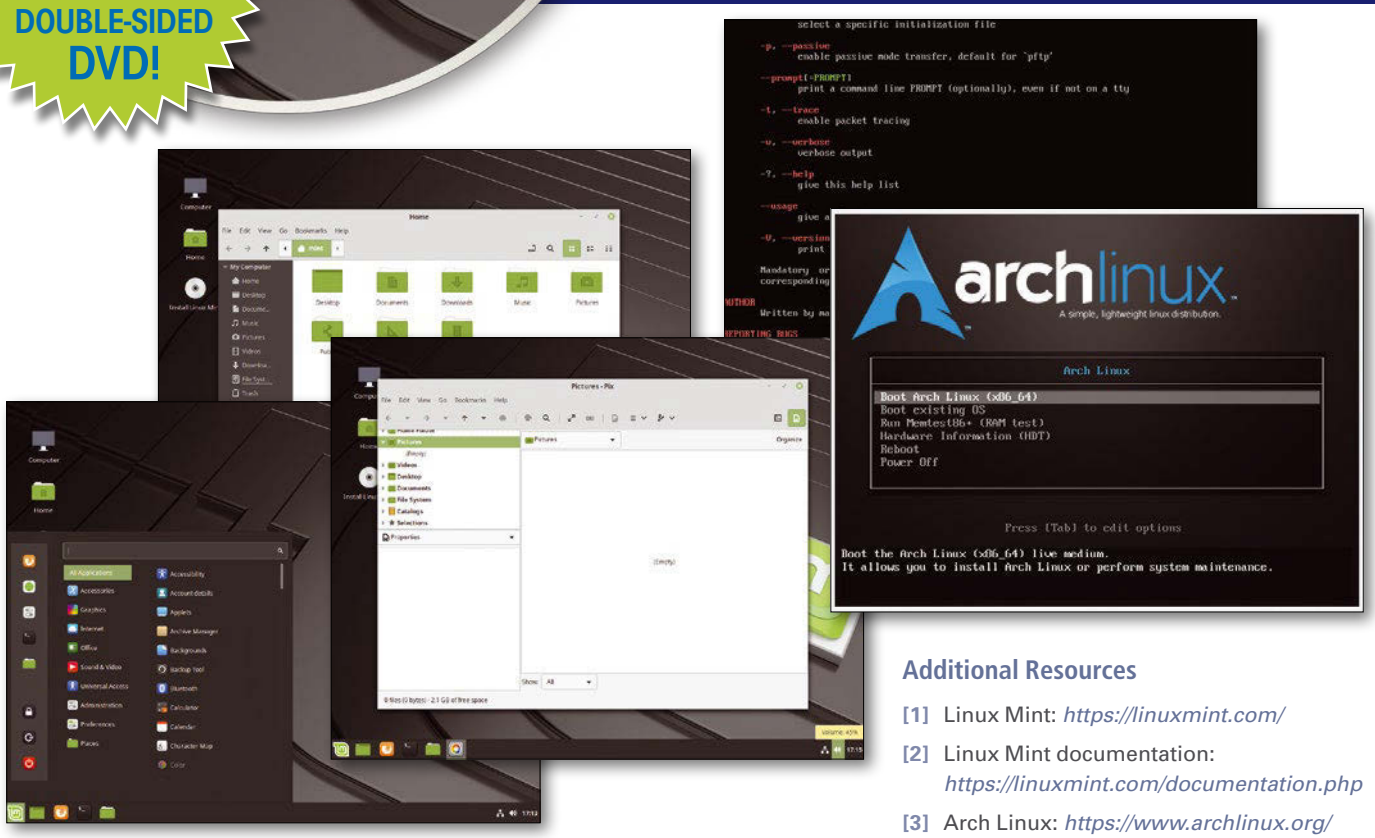
On the DVD

Linux Mint 19.1 "Cinnamon"

Linux Mint is an Ubuntu-based Linux designed for out-of-the-box ease of use. Version 19.1 is a long-term support release with updates and security patches until 2023. The Cinnamon edition is built around the popular Cinnamon desktop, a project based on Gnome 3 that seeks to preserve conventional desktop metaphors in the modern age. Version 19.1 comes with the Cinnamon 4.0 desktop and includes enhancements for the Update Manager, performance improvements for the Nemo file manager, and better support for NVidia graphics cards.

Arch Linux

Arch is the ultimate Linux hacker system. The Arch developers are a tight-knit community of experts who are devoted to a minimalist philosophy. Arch is intended for advanced users. Don't expect a cornucopia of desktop apps or even a desktop user interface when you boot into Arch. What you can expect is simplicity, elegance, code correctness, and stability. If you want to add a desktop user environment and set up your own collection of desktop apps, you can rely on Arch's sizable package repositories and its venerable pacman package manager.



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

Additional Resources

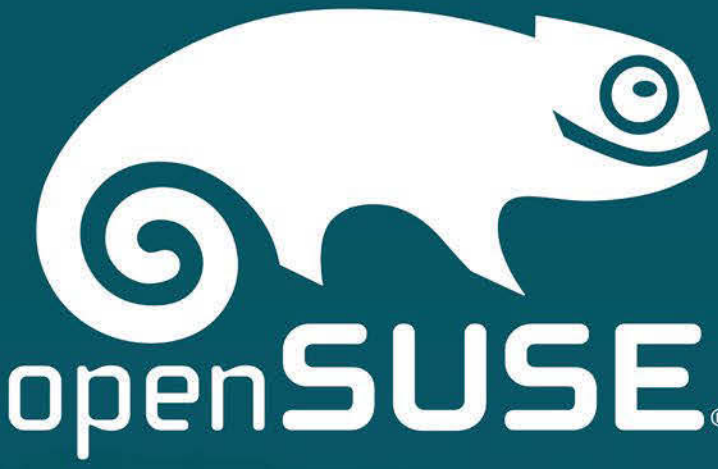
[1] Linux Mint: *https://linuxmint.com/*

[2] Linux Mint documentation: *https://linuxmint.com/documentation.php*

[3] Arch Linux: *https://www.archlinux.org/*

[4] Arch wiki: *https://wiki.archlinux.org/*

Nuremberg, Germany



Conference

Open Source Software

• Embedded Systems

• Cloud • Containers • More

May 24 - 26, 2019

events.opensuse.org

NEWS

Updates on tech...s

THIS MONTH'S NEWS

GitHub Offers Free Private Repositories

GitHub has announced that it is now taking on players like GitLab and offering free private repositories. Anyone could always set up a free repository on GitHub; the condition was that the code had to be public, which meant that projects and organizations could not set up private repositories. If they wanted private repository, they had to pay.

Now anyone can create a private repository for free (*https://techcrunch.com/2019/01/07/github-free-users-now-get-unlimited-private-repositories/*). The only caveat is that there can be at most three collaborators to the project, which means big organizations can't exploit the free service to manage their mega projects.

A private repository lets developer communities work on the codebase internally, away from the public. GitHub competitors like GitLab already offer free private repositories.

Linus Torvalds Welcomes 2019 with Linux 5

Linus Torvalds has announced the release of Linux 5.0-rc1 (*https://lkml.org/lkml/2019/1/6/178*). The kernel was supposed to be 4.21, but he decided to move to the 5.x series. Torvalds has made it clear that the numbering of the kernel doesn't make much sense, so don't get too excited about this release.

Torvalds explained in the Linux Kernel Mailing List (LKML): "The numbering change is not indicative of anything special. If you want to have an official reason, it's that I ran out of fingers and numerology this time (we're *about* 6.5M objects in the git repo), and there isn't any major particular feature that made for the release numbering either," he said.

Image © guesswho, 123RF.com

The release brings CPU and GPU improvements. In addition to support for AMD's FreeSync display, it also comes with support for Raspberry Pi Touchscreen.

Talking about the "content" of the kernel, Torvalds wrote: "The stats look fairly normal. About 50% is drivers, 20% is architecture updates, 10% is tooling, and the remaining 20% is all over (documentation, networking, filesystems, header file updates, core kernel code..)."

SQLite Database Vulnerable

The Tencent Blade security team has discovered a vulnerability in the immensely popular open source SQLite database engine. Tencent is one of the three Chinese giants known as BAT (Baidu, Alibaba, and Tencent).

"This vulnerability can be triggered remotely, such as accessing a particular web page in a browser, or any scenario that can execute SQL statements," said a Tencent blog post (*https://blade.tencent.com/magellan/index_en.html*).

Because SQLite is one of the most widely used databases, touching all modern applications, this vulnerability affects a wide range of the user base (*https://www.zdnet.com/article/sqlite-bug-impacts-thousands-of-apps-including-all-chromium-based-browsers/*).

According to ZDNet, "Firefox and Edge don't support this API, but the Chromium open source browser engine does. This means that Chromium-based browsers, like Google Chrome, Vivaldi, Opera, and Brave, are all affected." That said, Firefox is affected because it comes with a locally accessible SQLite database, allowing it to be exploited locally, but not remotely.

Hacks Abound

2018 is ending with some major hacks. Marriott International, one of the world's biggest hotel chains, announced that hackers compromised the reservation database of Starwood hotels. Hackers managed to steal personal details of about 500 million guests. According to The Hacker News, "The breach of Starwood properties has been happening since 2014 after an unauthorized party managed to gain unauthorized access to the Starwood's guest reservation database and had copied and encrypted the information" (*https://thehackernews.com/2018/11/marriott-starwood-data-breach.html*).

The second victim of another major hack is Quora, a user-driven question and answers site. According to reports, hackers gained access to sensitive information of over 100 million users (*https://thehackernews.com/2018/12/quora-hack.html*). The Hacker News wrote that the stolen data includes sensitive account information, such as names, email addresses, encrypted (hashed) passwords, and data imported from linked social networks like Facebook and Twitter.

Lead image © ducdao, 123RF.com

The third major hack was on Dell. The company said that it detected and disrupted unauthorized activity on its network attempting to extract Dell.com customer information, which was limited to names, email addresses, and hashed passwords. "Additionally, Dell cybersecurity measures are in place to limit the impact of any potential exposure. These measures include the hashing of our customers' passwords and a mandatory Dell.com password reset. Credit card and other sensitive customer information was not targeted. The incident did not impact any Dell products or services," Dell said in a blog post (*https://www.dell.com/learn/us/en/uscorp1/press-releases/2018-11-28-customer-update*).

Even though Dell was not certain if any data was stolen, the company pushed password reset for all users as a precaution.

MORE ONLINE

Linux Magazine
www.linux-magazine.com

Linux Administration Focus
http://www.linux-magazine.com/tags/view/administration

Git Started with Git • Roman Jordan
The Git version control system is a powerful tool for managing large and small software development projects. We'll show you how to get started.

Remote Git Repositories • Roman Jordan
Software projects often comprise several code branches, some of which exist in parallel. Git supports community code development through remote repositories and code branching.

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

Parallelizing Code – Loops • Jeff Layton
OpenACC is a great tool for parallelizing applications for a variety of processors. In this article, I look at one of the most powerful directives, loop.

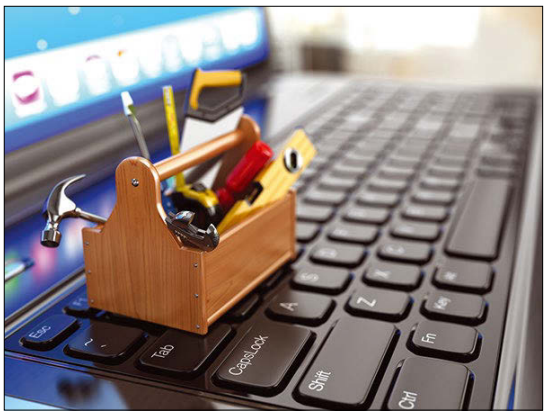
ADMIN Online
http://www.admin-magazine.com/

Exploring SQL Server on Linux
David Barbarin
SQL Server runs on Linux now. We'll show you how Microsoft developers made their massive database system Linux ready, and we'll help you get started with setting up SQL Server on your own Linux system.

Sharing threat information with MISP
Matthias Wübbeling
The Malware Information Sharing Platform lets you record and document security incidents – and share the information with users on other networks.

Kubernetes Vulnerability Found and Fixed

A critical vulnerability was discovered in the Kubernetes container orchestrator (*https://github.com/kubernetes/kubernetes/issues/71411*). The vulnerability (CVE-2018-1002105) allows non-privileged users to access Kubernetes clusters and associated data that they otherwise would not be able to access.



Lead Image © Maksym Yemelyanov, 123RF.com

Bad actors can exploit the flaw in two ways -- the first involves abusing pod exec privileges granted to a normal user, and the second involves attacking the API extensions feature, which provides the service catalog and access to additional features in Kubernetes 1.6 and later.

The flaw is already fixed and major Kubernetes vendors have already released patches. For instance, Red Hat has announced that OpenShift Container Platform 3.x and later are affected, as well as Red Hat OpenShift Online and Red Hat OpenShift Dedicated. The company suggests that users must immediately apply patches to their OpenShift deployments.

Microsoft Azure has announced that they have also fixed the vulnerability. The company said, "Azure Kubernetes Service has patched all affected clusters by overriding the default Kubernetes configuration to remove unauthenticated access to the entrypoints that exposed the vulnerability,"

The entrypoints are everything under *https://myapiserver/apis/*. If you were relying on this unauthenticated access to these endpoints from outside the cluster, you will need to switch to an authenticated path.

This is the first major vulnerability discovered in Kubernetes.

Dolphin Announces New Switch for Composable Architectures

Dolphin Interconnect Solutions has announced a new 24-port switch for I/O expansion and PCIe fabric. The MXS824 offers an innovative approach to composable architecture, a recent trend that combines the benefits of software-defined infrastructure with hardware-based device sharing and provisioning.

According to the announcement, "Dolphin's unique approach to building composable architectures is called device lending. Device lending allows access to devices installed in servers, as well as in expansion boxes or JBoFs. This creates a pool of transparent I/O resources that can then be shared among computers without any application-specific distribution mechanisms or requiring any modifications to drivers. Just as importantly, the resources can easily be reallocated whenever required, allowing for extremely flexible and ever-changing distributions of resources."

The MXS824 is designed to work with Dolphin's PCIe fabric to connect multiple servers with devices such as NVMe drives, GPUs, processors, and FPGAs in a way that allows on-the-fly software-defined configuration.

The 24-port Microsemi PFX-based 1U cluster switch delivers 32GT/s of non-blocking bandwidth per port at ultra-low latency. The switch supports various configurations, where up to four ports can be combined into a single x16 /128 GT/s port for higher bandwidth. Ports can be configured as 24x4 ports, 12x8 ports, 6x16 ports, and various configurations of each. Multiple switches can be connected to create larger port counts.

Zack's Kernel News

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

By Zack Brown

Kernel Cussing

A sort of unofficial "politeness police force" seems to be springing up in Linux kernel development. The door was opened when Linus Torvalds decided to take a short break from kernel development in order to evaluate his own behavior towards developers. In some cases, when he felt that a developer was going in the wrong direction with a given patch, Linus might lose his temper if the developer had already been given an explanation for why a different approach was better. After some of the developers confronted him directly on the harshness of his behavior, Linus took some time off to consider alternative behaviors.

That was a couple of months ago and he has since come back; but when Linus left, he also approved a patch inserting a new "code of conduct" (CoC) into the kernel source tree. Among other things, the CoC prohibited "offensive language."

Recently, Jarkko Sakkinen posted a patch to change "fuck" to "hug" in kernel code comments. Specifically, he targeted a code comment that had been put into the kernel by Linus himself, where Linus said he had "fucked up" someone else's well-done code. In Jarkko's new version, the comment would say that Linus had "hugged up" the code instead.

There was some controversy surrounding this patch. Eventually it came out that in addition to the CoC, there was another file that told how to interpret the CoC; in that file, it specifically said that the CoC only applied to patches that went into the tree after the CoC's adoption. So, Linus's old code comment would not be covered. It was at that point that Jarkko abandoned his patch.

But before then, a number of people had a variety of opinions about changing "fuck" to "hug," or about eliminating words like "fuck" from the source tree at all.

Some people supported Jarkko's idea. Kees Cook was in favor of removing instances of "fuck" from the kernel, but he felt that any replacement should preserve the original meaning. To "fuck something up," he felt, did not mean the same thing as to "hug something up." Kees suggested instead of wholesale replacement of one word with another, that each code comment be adjusted on a case-by-case basis. He said, "how about doing context-specific changes? 'This API is terrible', 'Hateful interface', 'Don't touch my freakin' code', 'What in the world were they thinking?' etc.?"

Some people were opposed to this sort of change because they were non-native speakers. Apparently the word "fuck" is very easy to understand regardless of one's native tongue, while more subtle language might be misinterpreted. Geert Uytterhoeven said that he, as a non-native English speaker, found replacements like "hugged it up" or "hecked it up" to be more confusing, and said he'd prefer the simpler, clearer language that, as he put it, is "easy to grasp for +7 year olds who were never taught English, but are exposed to modern global ways of communication."

Other people had no objection to taking "fuck" out of the source tree, but only if the comment needed to be updated anyway. Steven Rostedt felt that way about it. He pointed out that the comment changed by Jarkko's patch was actually completely out of date and could simply be removed entirely. He remarked, "that will be an accurate change with or without CoC."

Some people liked the patch or at least felt that such language should not be considered bad. Davidlohr Bueso said he hoped that Jarkko's patch was some kind of joke. John Paul Adrian Glaubitz said that there was nothing wrong with the word "fuck," and its presence in the kernel source tree was not going to make or break anyone's desire to contribute code.

Tobin C. Harding said he loved stumbling on things like Linus' off-color comments in the source code. As he put it, "this is my favorite comment to date in the kernel source tree. Surely there

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**.

are still some people working on the kernel that do so for fun. I actually laughed out loud when I first stumbled upon this file."

Jens Axboe felt that any attempt to legislate the existence of curse words was "insanity."

And David Miller took an even stronger stand, saying he absolutely refused to accept Jarkko's patch, or any other patch that censored contributors' language.

The discussion eventually came to an end. There seems to be a wide range of opinions on the issue. Maybe it would have been different if Jarkko's patch had been censoring someone other than Linus himself.

It's actually a fascinating issue – the desire to make everyone comfortable. Because of course, if one group can only be comfortable when another group changes the way they communicate and express themselves, then the group being asked to change can no longer be comfortable. How can we decide whose feelings matter more? How can we decide what constitutes a true affront to another person or group, and what are simply behaviors that the offended group should accept as legitimate diversity?

Maybe the answer is to let people say "fuck,", to let other people say "please don't say that," and simply live with the ebbs and flows of that controversy.

My personal belief is that "comfort" is overrated and is not more important than letting other people express themselves honestly. At the same time, it's nice to be encouraging rather than mean, and helpful rather than harsh.

It's a complex issue. And the Linux kernel is not just some little open source project off in the corner somewhere, with no influence or importance. The infrastructure of the entire world absolutely depends on Linux. And the choices made by its project leader and prominent contributors will ripple across the face of every country in the world. The things that happen on the Linux Kernel Mailing List are important. It sounds funny to say, but it's true.

Disk Encryption for Developing Nations

It's nice to be able to encrypt your disk in these days of stop-and-frisk. Sometimes when I travel, I wish there was a service that would store a password for me that I didn't have memorized. Then if border officials demanded that I unlock my devices, I could honestly tell them it wasn't in my power. When I got where I was going, the service would tell me the password, and I could once again change it to something that only I know.

However, passwords aren't so useful if someone has physical access to your unencrypted drives. And some Android phones are intended for developing countries, where top-of-the-line hardware would be prohibitively expensive. Trying to encrypt a drive using such low-end hardware can be a real challenge.

Recently, Eric Biggers and Paul Crowley decided to code up some disk encryption options for exactly these groups of low-end phones. Their goal was to design something in software that was truly secure, but that would also run fast enough to be usable on those systems. As Eric put it, "encryption is for everyone, not just those who can afford it. […] Lack of hardware support should not be an excuse for no encryption."

After some effort, they came up with the Adiantum encryption mode for Linux. While not a new encryption algorithm, it used D. J. Bernstein's XChaCha12 cypher and AES-256 together to meet their speed and security needs. In fact, Adiantum was an enhancement over their earlier effort, the HPolyC encryption mode, providing a 20 percent speedup compared to that, without any degradation of security.

The biggest controversy surrounding the Adiantum patches turned out to be whether or not they would use the Zinc crypto API. Zinc is very likely to go into the kernel at some point and replace the existing Cryptodev API wholesale, so from that standpoint it makes sense to build all crypto-related patches on top of Zinc. However, Zinc has problems, particularly some unresolved licensing issues for some of its assembly files. Until those are resolved, Eric said, it was still unclear whether new crypto projects should be built on top of Zinc.

But in spite of his misgivings, and at the urging of the Zinc author Jason A. Donenfeld, Eric did create a new Git

branch dedicated to reworking the Adiantum patches to use Zinc. Eric and Paul would henceforth work on two separate versions of Adiantum – one that could be adopted with Zinc, and one that would rely on the existing Cryptodev infrastructure within the kernel.

The question of how to develop good encryption is a very interesting one. For a long time, the official Linux kernel had to have a separate encryption patch hosted internationally, because import/export laws in the United States prohibited Linux from implementing military-grade encryption. That restriction has since been lifted, but the modern political danger is that governments continue to want special "back doors" built into software, so that they can gain access without needing cooperation from the owners of a given Linux system.

Of course, back doors will most certainly be exploited by hostile actors and will therefore do far more harm than good. But governments are pulling their hair out over the issue of criminals using untappable communication channels to plan crimes. They seem to think that if it's illegal to use good encryption, the criminals will somehow decide not to do it. In reality, law-abiding citizens will simply be exposed to many dangers, while the criminals will continue to communicate over strongly encrypted channels.

Fixing Up the Function Graph Editor

Steven Rostedt recently rewrote the function graph tracer. This is a debugging tool that tracks function calls and the return from those functions. Using that information, a developer can make educated guesses about what exactly the kernel was doing when a particular run-time problem occurred. The kernel's function graph tracer distinguishes itself from a regular function tracer because regular tracers usually only track function calls and not the returns from functions.

Steven originally wanted to rewrite the function graph tracer just to spruce it up a little bit; however, while researching the project, he discovered a serious design flaw – it would only allow a single user to access it at a time. Other tracers did not have that restriction, so Steven felt that now it was even more urgent for him to do this rewrite.

Even so, Steven's solution did not allow an arbitrary number of simultaneous users. The reason was that capturing the data needed by the function graph tracer was complex and difficult, so each supported user had to have its own set of infrastructure providing that support. For Steven's initial effort, he chose to support 16 independent users, which he felt would be more than enough.

However, Peter Zijlstra threw some cold water in Steven's face, saying that he personally didn't use the function graph tracer, not because it hadn't supported multiple users, but because performance was terrible. That was the feature he wanted to see improved. He said he had personally never had a problem with lack of multi-user support.

Steven replied that he had also been working on speed improvements, and that there had actually been a significant improvement in that area. But he also pointed out that multi-user support was still an important issue, with real developers who would benefit from it. Steven said, "we plan on merging kretprobe with function graph tracing. This also solves the issue that Mark Rutland has with ret protection. He has a solution for function graph tracing, but not with kretprobes. And yes, there's also the case of being able to trace to different buffers where you can have a full function graph tracing enabled and also trace a subset that you want to have."

Peter backed off, affirming that he hadn't noticed the speed improvements, because he hadn't used the function graph tracer in a while.

Masami Hiramatsu was very enthusiastic about Steven's work, as he was developing kretprobe and had a particular need of the multi-user feature. The two of them then went off on a technical discussion about various implementation details, regarding how to collect specific information about parameters and return values and include that with the function graph tracer's output.

There was no real controversy about these patches. Steven was simply giving some love to a debugging tool that needed it. Since it has no direct impact on kernel performance or behavior, there's really no objection. Its features either help kernel development or they don't. ■■■

Useful and lesser-known features of VirtualBox

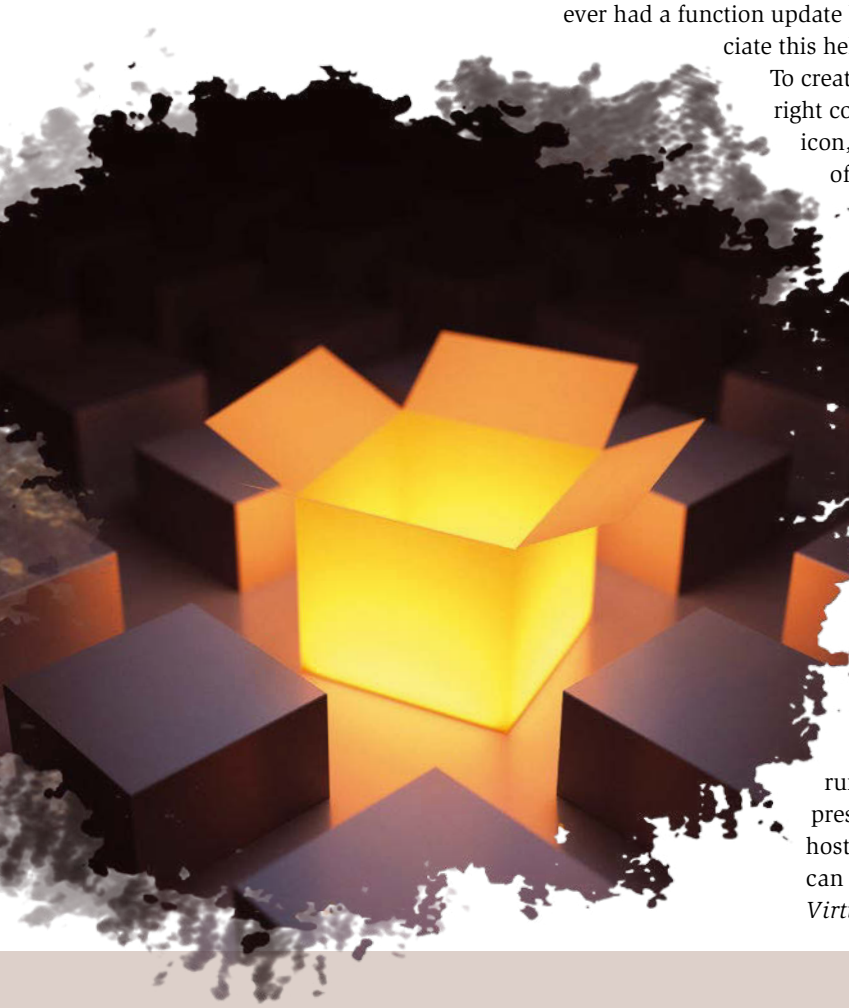
Under the Hood

The VirtualBox virtual machine tool is a familiar sight on Linux systems, but many users don't access the full range of its powers. This article highlights some advanced features of VirtualBox that could save you some time and effort. *By Tim Schürmann*

VirtualBox is a powerful virtualization tool that supports a variety of Windows and Linux guest systems. Most users control VirtualBox via the VirtualBox Manager. In this graphical user interface, you can set up a virtual machine with just a few mouse clicks. However, many VirtualBox users don't realize that the VirtualBox Manager also has some interesting advanced functions hidden in the depths of its settings menus and dialogs.

Be Friendly

Snapshots freeze the current state of a virtual machine and save it. You can then restore this state any time later with just a mouse click. Snapshots are especially useful for Windows guests: Before an update is due, you create a new snapshot to which can you return with just two mouse clicks in case of a failed update. Anyone who has ever had a function update blow up in their face under Windows will quickly appreciate this help.



To create a new snapshot, click the small black triangle in the top right corner of the VirtualBox Manager, next to the *VM Tools* icon, and select *Snapshots* (Figure 1). You will now see a list of all snapshots that have already been taken. At first, you will only see the *Current state* entry.

To freeze this state, click *Create*. In the dialog that appears, give the snapshot a name. The name should briefly summarize why you took the snapshot, such as *Windows 10 post install*. If necessary, you can add a detailed description in the large input field.

After pressing *OK*, VirtualBox creates the new snapshot. In a snapshot, VirtualBox not only remembers the state of the virtual machine but also its configuration. If you click on *Properties*, the *Information* tab reveals the settings (Figure 2).

After you create the snapshot, the virtual machine can start normally. If you want to create another snapshot, simply click on *Create* again. In the list, VirtualBox indents the snapshots, as shown in Figure 2, which means you can see the snapshot sequence at a glance.

VirtualBox also lets you quickly create snapshots in the running machine by calling *Machine* | *Create snapshot* or pressing the host key and *T* at the same time. By default, the host key corresponds to the right Ctrl key. If necessary, you can change this in *File* | *Settings* in the *Input* area of the *Virtual Machine* tab next to *Host key combination*.

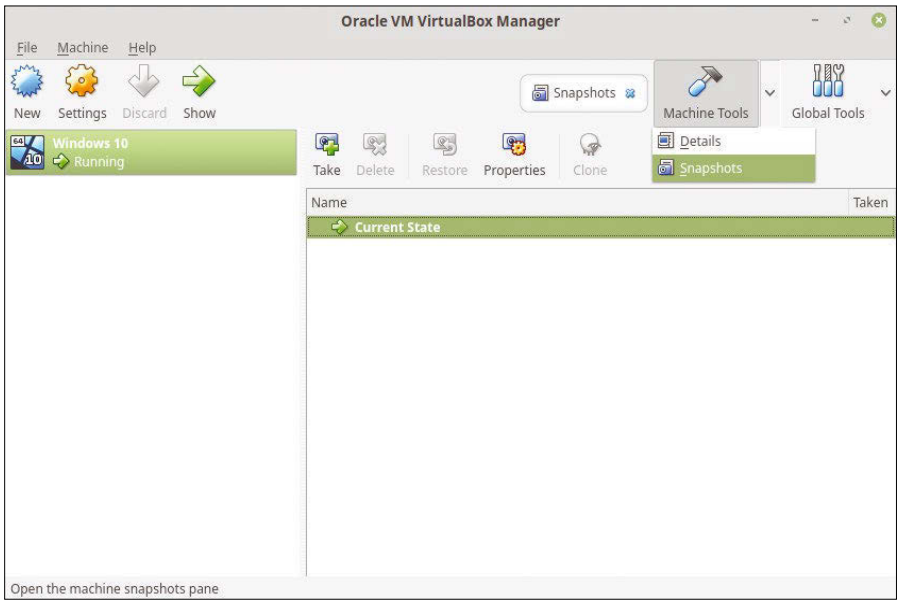


Figure 1: You can quickly switch between the displays using the *Snapshots* and *Configuration* buttons.

When creating a snapshot, VirtualBox freezes the disk image in the background. VirtualBox remembers all the subsequent modifications and newly created files in a differential image. This special hard disk image contains all the changes to the original hard disk image. Thanks to this, VirtualBox does not have to copy the entire image, which in turn saves disk space.

Since VirtualBox remembers all your snapshots, intensive use of the function results in a nested hierarchy, as shown in Figure 2. The Windows drivers were installed in the *Driver installed* snapshot. Afterwards, the user loaded Photoshop and created another snapshot.

Then the user went back to the *Driver installed* snapshot, where the Affinity Designer program was installed and a suitable snapshot was again created. In the *Photoshop* snapshot, only Photoshop is installed, but not Affinity Designer. According to the same principle, different variants of a virtual machine can be created elegantly and in a space-saving way.

If you want to return to a previous state, simply select it from the list and click *Restore*. The other snapshots are not lost – you can therefore jump back and forth between the snapshots at any time. The VirtualBox Manager on the left side of the screen indicates the state in which the virtual machine will launch in the brackets next to the virtual machine name.

If you have changed a state, VirtualBox asks if you want to save your changes in the virtual machine to a new snapshot before restoring. If you reject the offer, you will lose the changes accumulated since the last snapshot.

To remove a snapshot, select it from the list and click the *Delete* button. The *Clone* option lets you quickly duplicate a snapshot. To edit the virtual machine configuration again, click the triangle next to *VM Tools* and select *Configuration*.

Expanding the Disk

If a virtual hard disk becomes too small, you can expand it with a few mouse clicks. To do this, stop the virtual machine if necessary and then select in VirtualBox Manager *File* | *Virtual Media Manager*. In the *Drives* tab, select the image you want to expand and display the *Properties* (Figure 3).

The slider below *Size,* tells VirtualBox to resize the disk image. Confirm your change with *Save*. However, the procedure only works with image files in VDI and VHD format, which must also be created dynamically.

In addition, VirtualBox only lets you *expand* a virtual disk – you can't make it smaller, even if the disk image is not completely filled with data. Also, the operating system on the virtual machine is not yet aware of the change. You'll need to use the disk tools on the guest system to extend the partitions on the current guest to match.

Similarities

By default, VirtualBox ensures that each disk image can only be used by precisely one active virtual machine. Sometimes, however, several guests running in parallel have to access the documents on a virtual hard disk. To allow several guests to access a

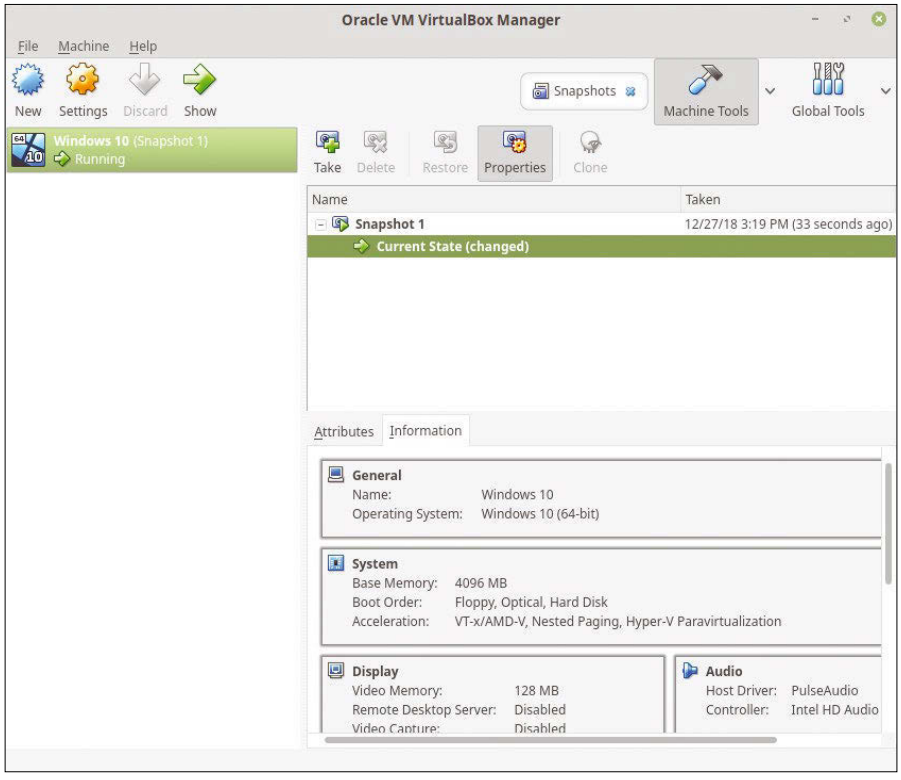


Figure 2: The *Information* tab lets you view snapshot settings.

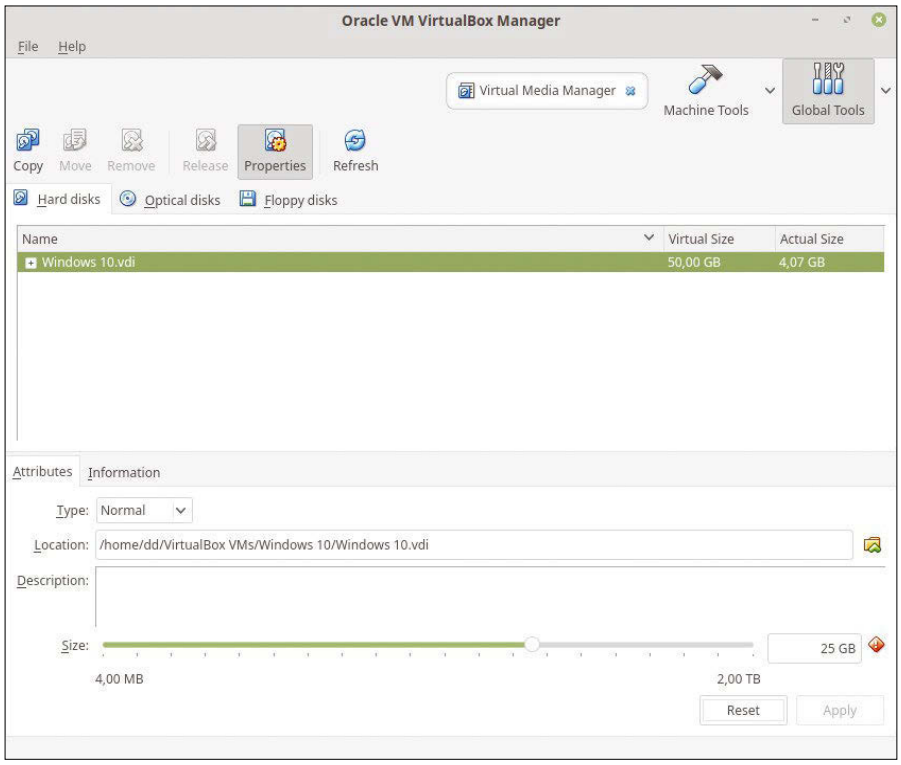


Figure 3: An attempt was made to reduce the size of the hard disk by dragging the slider to the left. VirtualBox prevents this and indicates the problem with a red warning triangle.

disk image, a few mouse clicks are required: First, shut down the virtual machines that need to access the disk image.

Then go to *File* | *Virtual Media Manager* and select the desired image in the *Drives* tab. Open the *Properties*, in the *Attributes* tab, set the *Type* to *Multiple connections*, and confirm by pressing *Save* (bottom-right corner of the window).

VirtualBox may now notify you that it needs to share the image. Allow this by pressing *Share*. As soon as the *Type* is set to *Multi-attach*, the image file can be mounted on several virtual machines as usual using *VM Tools* (Figure 4). If you have mounted the disk image on a virtual machine previously, you need to mount it there again.

If two virtual machines access a disk image at the same time, you could end up with corrupted data. To prevent this, VirtualBox intercepts all modification attempts and stores them in the background in a differential image. Since each virtual machine receives its own differential image, the guest systems no longer get in each other's way. As a side effect, the original hard disk image remains in its original state.

However, you also need to back up the differential images. To find their locations, go to *File* | *Virtual Media Manager* and search for the disk image in the *Drives* tab of the VirtualBox Manager, and then click the small triangle in front of the name. VirtualBox now unfolds all the differential images, each with a cryptic identifier. As soon as you click on an image, the *Attribute* tab below *Location* shows its storage location.

By using the differential images, the virtual machines only ever see their own changes. For example, if the virtual machine modifies a letter with Fedora, Windows does not notice the change on another virtual machine. *Multiple Connect* hard disk images are therefore not suitable for collaboration on a single document. For collaboration, you need to switch to shared folders. (See also the "Keeping Folders Tidy" box.)

Back to Square One

An alternative to *Multiple Connect* is the *Shared* type. This type works on the same principle, but VirtualBox creates a fresh differential image each time the virtual machine is restarted. The changes made by the virtual machine and all files created are lost each time the virtual machine is restarted. The *Shared* type can only be assigned to image files that do not grow dynamically.

If you are using snapshots and you do not want VirtualBox to store the disk image in the current snapshot, you can set the *Type* to *Copy*. If you set the *Type* to *Immutable* instead,

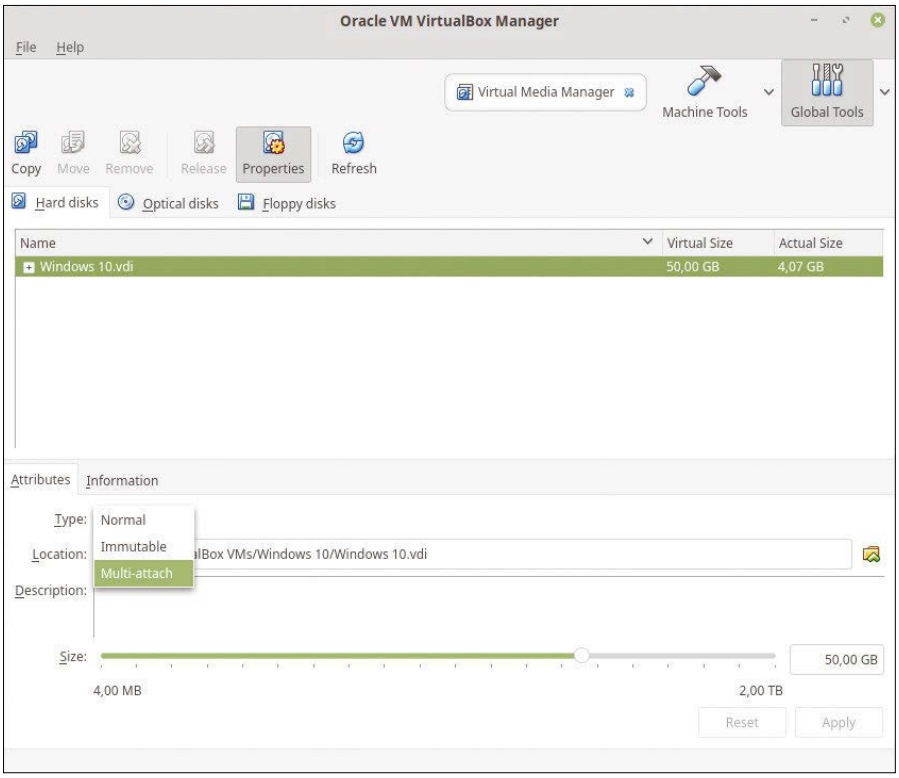


Figure 4: You can configure VirtualBox to let you mount a virtual hard disk on another virtual machine.

Keeping Folders Tidy

You can use VirtualBox Manager to change the directory where VirtualBox stores all the virtual machines. Go to the *File | Settings* menu and access in the *Default path for VMs* option below *General*. In the future, all new virtual machines will end up in the folder at the path you define; VirtualBox will retain the existing virtual machines. You can quickly jump to the directory of the currently selected virtual machine via *Machine | Show in file manager*.

VirtualBox enables write protection. But be careful: The guest system can still change the data of the hard disk image. However, when you restart the virtual machine, VirtualBox restores the image to its original state.

Data you previously thought you stored is then lost. In the background, VirtualBox uses differential images, which it deletes each time the virtual machine is restarted. The *Immutable* type is therefore only of interest in special cases, such as on publicly accessible computers or on PCs in training rooms, where you want to automatically restore the system to its original state after use.

If an error message appears when you change the type, the disk image is most likely still in use – or it belongs to a snapshot. In these cases, you first need to share the image or delete the snapshots.

Integrated Applications

You can also bring the windows running on a virtual machine to the desktop. VirtualBox calls this mode Seamless Mode, but it can only be enabled if you installed the Guest Additions on the virtual machine and are running Windows or a Linux system with X11 as a guest.

In this case, you need to press the Host key and *L*. VirtualBox then inflates the virtual machine to the screen size and hides the background of the guest system. To exit this mode, press Host + L again.

On My Command!

VirtualBox comes with the VBoxSDL command-line tool. VBoxSDL starts a virtual machine in a simple window. The VirtualBox developers only used the tool for internal tests of their own work, which is why they do not officially support it. However, it can be useful if you want to start the virtual machine from a script or if you don't want the virtual machine users to interfere with the configuration. To start a virtual machine, pass VBoxSDL the name of the virtual machine in the `--startvm` parameter (Listing 1, line 1).

As an alternative to the graphical user interface, the virtual machines can be managed with the VBoxManage command-line tool. Like VBoxSDL, VBoxManage offers the advantage that it is suitable for use in your own scripts. In addition, the tool gives you access to some functions that the VirtualBox Manager itself does not offer. To start a virtual machine with VBoxManage, pass the `startvm` command and the name of the virtual machine to the tool (Listing 1, line 2).

Instead of the name, you can enter the unique identification number (UUID) of the virtual machine. The `list` command (Listing 1, line 3) reveals which virtual machine has which UUID. Information about a virtual machine is returned

by `showvminfo` (line 4). The virtual machine can also be stopped (line 5), resumed (line 6), restarted (line 7), and turned off (line 8). The restart and shutdown actions correspond to pressing the reset button or unplugging the power plug. If you want to shut down the virtual machine in a controlled manner, send an appropriate ACPI command to the guest (line 9).

VBoxManage also provides the ability to create new virtual machines and customize the configuration of existing machines. The commands required for this don't just *look* complex and cumbersome, they are. If you don't have scripts to group them, you can put a new machine together far faster in the graphical user interface.

VBoxManage is happily faster when it comes to exporting (listing 1, line 10) and importing (line 11) a virtual machine and its settings to or from a hard disk image.

New Warehouses

VBoxManage lets you quickly create empty media in the current directory (Listing 2). The `disc` option creates a disk image; alternatively, `dvd` creates an empty DVD, or `floppy` creates an empty floppy disk image. `--size` is the size of the data carrier in megabytes.

For hard disk images, the additional `--format` parameter lets you explicitly specify the format: `VDI`, `VMDK`, or `VHD`. The media is automatically registered by VBoxManage, so you will find it immediately in the VirtualBox Manager under *File* | *Virtual Media Manager*. If necessary, click on *Update*.

VBoxManage also lets you convert a hard disk image created with `dd` to VDI, VHD, or VMDK format. The commands from Listing 3 load the contents of the disk `/dev/sdc` into the `disc.raw` image and convert it into the `image.vdi` image.

VirtualBox can also pass a real hard disk or partition directly into the virtual machine. To do this, you first need to create an image file of your physical data carrier. The command from Listing 4 creates a new image named `file.vmdk` in the `/home/tim/` folder. If you mount the image on a virtual machine, it directly accesses the `/dev/sdc` drive. However, the whole procedure requires VirtualBox to have read and write access to the `/dev/sdc` drive.

Listing 1: VirtualBox from the Command Line

```
01 $ VBoxSDL --startvm "Windows 10"
02 $ VBoxManage startvm "Windows 10"
03 $ VBoxManage list vms
04 $ VBoxManage showvminfo "Windows 10"
05 $ VBoxManage controlvm "Windows 10" pause
06 $ VBoxManage controlvm "Windows 10" resume
07 $ VBoxManage controlvm "Windows 10" reset
08 $ VBoxManage controlvm "Windows 10" poweroff
09 $ VBoxManage controlvm "Windows 10" acpipowerbutton
10 $ VBoxManage export "Windows 10" -o win10.ovf
11 $ VBoxManage import win10.ovf
```

Listing 2: Creating Empty Media

```
$ VBoxManage createmedium disc --filename platte.vdi --size
  30000
```

Listing 3: Loading the Disk Contents

```
$ sudo dd if=/dev/sdc of=disc.raw
$ VBoxManage convertfromraw disc.raw image.vdi -format VDI
```

Intervention

If the Guest Additions are installed on the guest system, you can start programs directly on the guest with VBoxManage. The command from Listing 5 logs into the virtual machine with Fedora 29 as the user *tim* and a password of *123456*; when it gets there, it launches the gedit text editor.

`--exe` is followed by the full path to the program you wish to execute. In the case of a Windows virtual machine, you need to enter the backslashes in the path twice, such as `--exe "C:\\Windows\\System32\\calc.exe"`. For a graphical X11 application to launch on a Linux guest, you need to set the `DISPLAY` environment variable using the `--putenv` parameter. You can also add further parameters to the program (Listing 6).

The parameters are always at the end following the two minus signs. In the example, `gedit/arg0` indicates that the parameters for the `gedit` program follow next. In the example, this is just the complete path to the text file that you want gedit to open.

The `guestcontrol` command gives an insight into the state of the guest system if required (Listing 7, first line). Among other things, `guestcontrol` also lists the current processes. If a process is hanging, you can kill it with `closeprocess` (second line). `--session-id` is the ID of the session in which the process is running. The number at the end represents the process ID of the troublemaker.

Conclusions and Outlook

Additional commands for VBoxManage, including numerous examples, are available in the comprehensive user manual for VirtualBox [1]. If you work regularly with VirtualBox, you should at least browse the table of contents of the user manual: It provides many hints for interesting functions that are often overlooked or forgotten in everyday life. ■■■

Listing 4: Creating a New Image

```
$ VBoxManage internalcommands createrawvmdk -filename /home/
  tim/file.vmdk -rawdisc /dev/sdc
```

Listing 5: Launching an App on the Guest

```
$ VBoxManage guestcontrol "Fedora 29" start --exe "/usr/bin/
  gedit" --username tim --password 123456 --putenv
  "DISPLAY=:0"
```

Listing 6: Adding Parameters

```
$ VBoxManage guestcontrol "Fedora 29" start --exe "/usr/bin/
  gedit" --username tim --password 123456 --putenv
  "DISPLAY=:0" -- gedit/arg0 /home/tim/brief.txt
```

Listing 7: guestcontrol

```
$ VBoxManage guestcontrol "Windows 10" list all
$ VBoxManage guestcontrol "Fedora 29" closeprocess
  --session-id 4 2022
```

Info

[1] VirtualBox user manual:
https://www.virtualbox.org/manual/UserManual.html

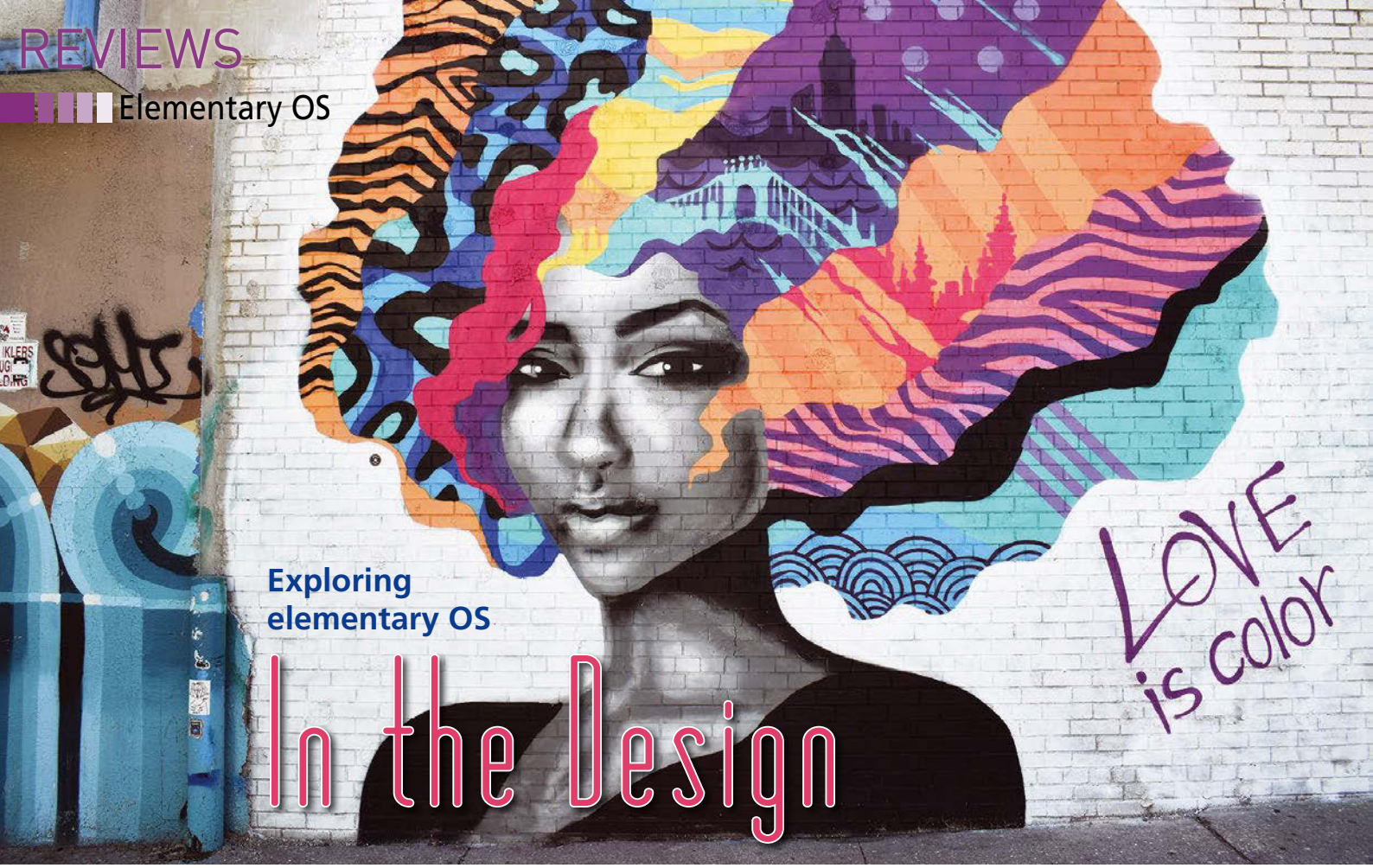
Complete Your Open Source Library with Archive DVDs!

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

Save hundreds off the print and digital copy rate with a convenient archive DVD!

Order Your DVD Now!

shop.linuxnewmedia.com



Exploring elementary OS

In the Design

Elementary OS is an elegant Linux with a long-term vision and a focus on good design. *By Swapnil Bhartiya*

"Good design makes a product useful," said the legendary industrial designer Dieter Rams. I couldn't agree more. My productivity is directly proportional to how well designed the tool is. I care about the UI elements – fonts, icons, the spacing between elements, and so on.

Unfortunately, when it comes to Linux on the desktop, the design is often an afterthought. Most projects don't have a UI designer on the team (some projects are a one-man army). As a result, what you get is all too often a patchwork that stitches disconnected components together.

However, there are exceptions. Elementary OS [1] is a distribution that attempts to bring the design principles of legendary design-first companies such as Braun, Dyson, and Apple to Linux. Good design is in the DNA of the elementary OS team, as its cofounders are both user experience (UX) designers. They have created very comprehensive design guidelines for developers building for elementary OS.

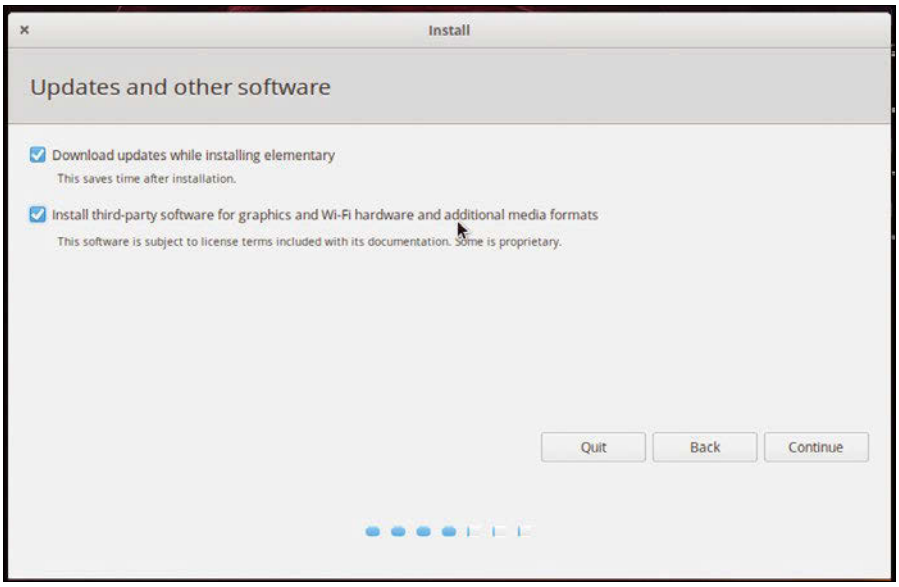


Figure 1: You can choose to install updates and codecs during installation.

In a nutshell, their basic idea is that design is an integral part of product development; the product should start with design. Reinforcing Rams' design views, elementary OS's Human Interface Guidelines (HIG) [2] clearly state that "it's not just the colors and fonts. Design is how it works. When you decide to add a button that does a thing, that is design. You made a decision to add a button with an icon or a label and where that button went and the size and color of that button. Decisions are designs."

The elementary OS HIG also makes it clear that design is more than just opinion and preference; it's testable. As stated on the HIG web page, "one design will meet a specific goal better than another design." And as someone who plays with elementary OS once in a while, I agree with their views. Elementary OS design principles make it a suitable distribution for its target users,

Photo by Chris Barbalis on Unsplash

which happen to be average macOS and Windows users.

Recently, a new version of elementary OS (aka Juno) was released [3]. I downloaded it and started playing with it.

Out of the Box Experience

Since elementary OS is based on Ubuntu, it brings simplicity and ease of use to its users. Installing elementary OS is very easy. During installation, you also get the opportunity to install third-party drivers and codecs, so you don't have to do extra work after installation, and everything will work out of the box (Figure 1). And it does.

"Elementary OS is built on the Ubuntu kernel and latest hardware enablement stack; we support a broad and ever-up-to-date range of hardware. Touch as well as 1:1 scrolling with momentum and inertia are supported out of the box [as on] almost all modern hardware," said Cassidy James Blaede, cofounder and UX architect at elementary, Inc.

I ran elementary OS Juno on three different machines, and everything worked as expected. HiDPI monitors were detected and scaling was perfect. Bluetooth, WiFi, and touchpad worked out of the box – no problems what so ever – but then I have Dell machines that are known to work great with Linux.

"We also benefit quite a bit from elementary OS itself and all curated apps in AppCenter using a shared toolkit (GTK), so that smooth inertial scrolling that is supported at a low level is available inside of apps themselves. We are also seeing cross-platform tools like Chrome, Qt, and Electron pick up support for smooth inertial scrolling and touch input, but the experience is very consistent and well-supported, particularly within GTK apps from AppCenter," explained Blaede.

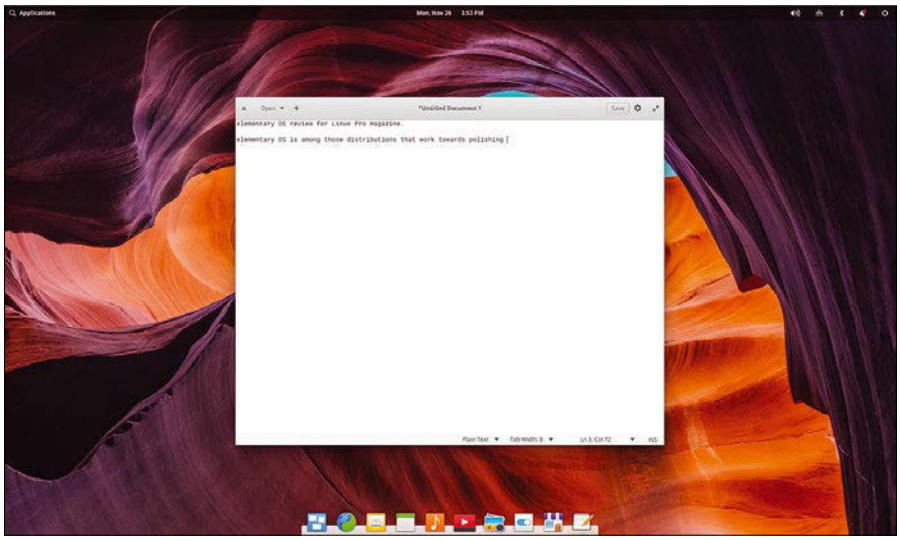


Figure 2: Elementary OS offers an elegant and visually appealing desk-

There were certain things (like full touch-screen support) that didn't work, but it was expected. In general, Ubuntu doesn't have great support for the touch screen, so it's not specific to elementary OS. If, you want a great elementary OS experience, choose hardware that's known to work well with Linux. When it comes to Linux, I don't look beyond Dell's Developer Edition machines.

Elementary uses Ubuntu as the base, but it's not a skin on top. They have built their own desktop environment – Pantheon – and other core applications. However, they are not suffering from the "not-invented-here" syndrome. They don't build everything. Whenever there is a third-party application that fits elementary OS' design principles, the team embraces it as part of the experience.

Love at First Sight

When it comes to user-experience and quality, compromise is not a word found in the elementary OS team's dictionary. Once you boot into elementary OS, you are greeted by the lovely desktop, which is minimalistic, elegant, and simple (Figure 2).

The Pantheon desktop is not much different from other desktop environments. It's designed for the windows, icons, menus, and a pointing device (WIMP) interface. Unlike Windows 10, it's not geared towards touch-based de-

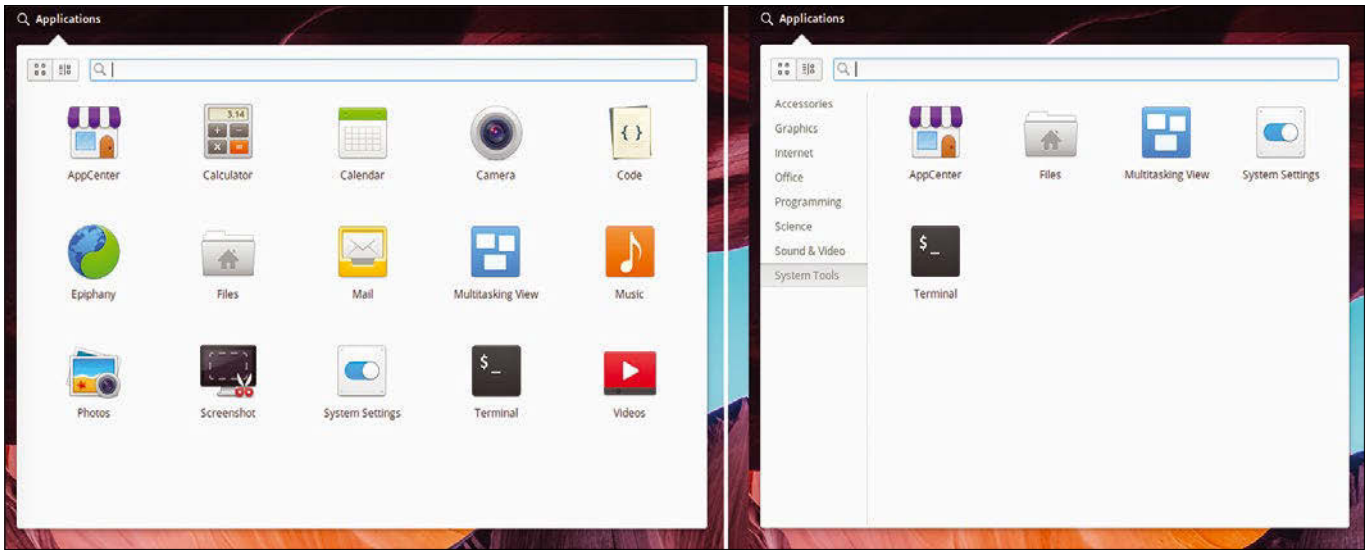


Figure 3: Change the layout of the app launcher.

vices, but you can get basic navigation on a touch device.

Like macOS (or Gnome, which is inspired by macOS), it has a panel on top and a dock at the bottom. The panel houses the app launcher (top left), date and time (center), and indicators for battery life, connectivity, and notifications (top right).

The app launcher defaults to a grid, but I changed it to a list for better discoverability (Figure 3). Now it shows categories of applications such as Accessories, Graphics, Internet, and so on. The dock serves multiple purposes. It offers quick access to apps and settings. Plus it's an app switcher. You will notice a blue dot under the running apps.

None of this needs explanation. An explanation would mean the elementary team failed at their job. Go ahead and explore it by yourself.

What's New?

If you have used elementary OS before, you will obviously notice many improvements and additions. Apple popularized the night mode, where the color temperature of the display changes to make it less stressful for eyes at night. Elementary OS is bringing that concept to its users. You can now set the time in System Settings so the display changes the color temperature during those hours.

Blaede loves the "adjustable tiling" feature, which might seem minor to other users, but he uses it frequently on his 27 inch display. "I often tile the code editor on one side of the display but want to give myself more room for another app or window, so I can just squish that code editor window down, and the other tiled window adjusts to adapt. It's super intuitive and just feels right," he said.

Daniel Fore, elementary, Inc. cofounder, loves the "Housekeeping" feature, which allows a user to automatically delete old trashed and temporary files. You can configure it in *System Settings | Security & Privacy*. He also loves Code, the elementary OS default code editor. "We spent a lot of time this cycle on redesigning and upgrading our default editor. Code has tons of new features and smarter defaults, and it's my favorite way to write apps on elementary OS," he said.

There are now parental controls that allow schools and libraries to fine-tune access to apps for children. You can limit time; you can block the website and apps. Interestingly, I blocked access to sites like YouTube and apps like Videos, but, when I logged into my son's account, I could access all those apps and services (Figure 4). I tested on macOS, and Safari blocked access to Youtube and every other app that I had blacklisted. I am assuming some wrinkles need to be ironed out.

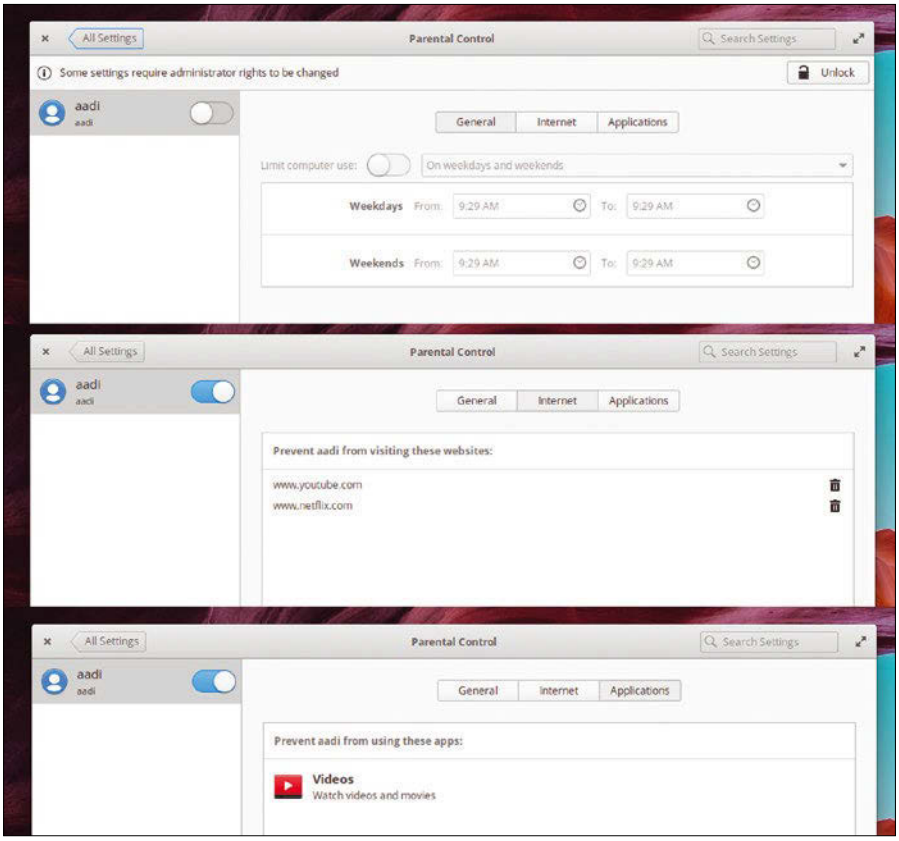


Figure 4: Parental Control allows you to restrict children's access to certain apps and sites.

Honey, Where Are My Apps?

Unlike many other Linux-based distributions, elementary OS doesn't come with a boatload of applications. You won't find LibreOffice, Firefox, VLC, and so on preinstalled. Given so many choices in the Linux world, it was difficult for the elementary OS team to choose such default apps.

"Some users prefer LibreOffice; others like Calligra; some would want AbiWord. So instead of choosing apps for them, we let users install whatever app they need," said Fore.

That's not the only reason that elementary OS comes with such a small set of preinstalled apps. Elementary OS only ships GTK+ apps to ensure a consistent user experience and desired integration with the system. Many third-party apps like Firefox don't offer full GTK+ support.

The elementary team used to ship such apps, but it resulted in a bad user experience, so they decided to stop. "Users can still install the apps they want, but we are working towards offering apps that work great with our platform," said Blaede.

The elementary team works with developers to help them curate their apps for elementary OS. They have very comprehensive documentation for developers, so that they can offer great integration with the platform [4].

As much as I appreciate elementary OS' desire to offer a consistent experience, there are many more users of Firefox and Chrome. I can't expect companies like Mozilla and Google to care about one of the many Linux distros like elementary OS. Their baseline is Ubuntu and Fedora/openSUSE. However, it's the elementary OS users who suffer. I can't, for example, watch Netflix on the default web browser.

I think the elementary OS team should look at this problem from a different perspective and refine the OS so that it can

offer the same experience that you get from Ubuntu, Fedora, openSUSE, or Zorin OS. None of them break third-party apps the way elementary OS does.

How to Install Apps

Elementary OS is based on the Ubuntu LTS versions, so every driver, codec, and app that's available on Ubuntu is available on elementary OS. There are two ways of installing applications on elementary OS: the GUI way and the command line. Since elementary OS focuses on look and feel, I chose to stick to the GUI method and see how far I could go. I used the AppCenter and installed LibreOffice, gedit, Gimp, Firefox, Thunderbird, and VLC (all I needed for my writing machine).

Just open the AppCenter (Figure 5) and search for the app you need, open the app, and hit the *Install* button – easy peasy. The AppCenter has both paid and free apps (more on that later).

You can also install "free" apps, which are in the repository, using the command-line interface. I installed LibreOffice by running the following command:

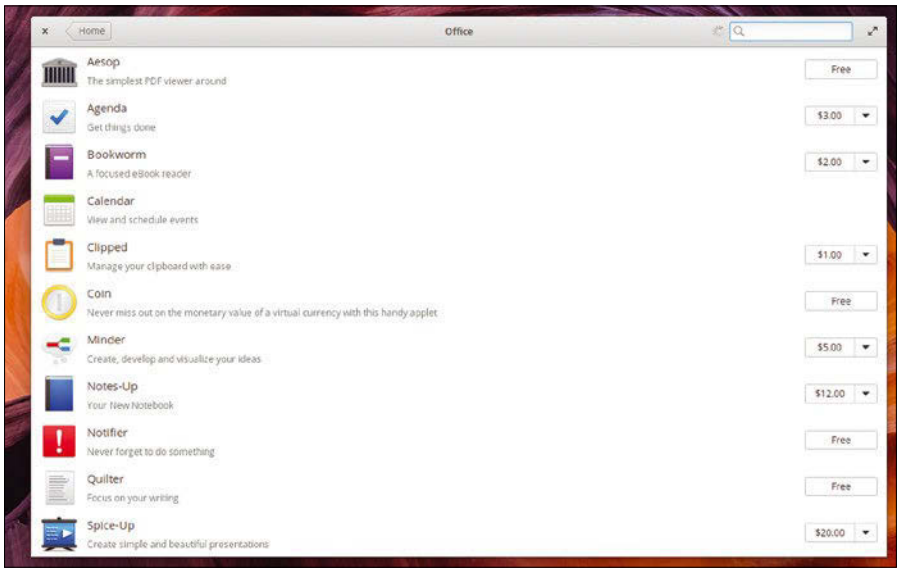


Figure 5: You can install both paid and free apps from AppCenter.

```
apt-get install libreoffice
```

Unlike other Ubuntu-based distributions, elementary OS does not offer a clutter of system update tools. AppCenter is the tool that shows you all the updates – both app level updates and OS-level updates (Figure 6). Just go to AppCenter and check if any updates are available. If there are any updates available, elementary OS also shows a notification on the top right corner.

Alternatively, you can use the command line to check and install system updates:

```
sudo apt-get update
sudo apt-get dist-upgrade
```

Since elementary OS is based on Ubuntu LTS releases, you don't have to worry about a system upgrade every six months. However, I do recommend running the system update command on daily basis.

Some Rough Edges

Not everything is rosy in the elementary OS world. One of the biggest challenges that I faced was lack of integration between mail, calendar, and contact apps (in fact, there is no contact app). I had to configure the email and calendar separately, which also meant that, unlike Apple mail, if there is a calendar invite in my mail, I couldn't add it to my calendar directly from the mail app. Since there is no contact app, and Gnome Online Accounts doesn't work in elementary OS, I could not bring my contacts to the platform.

The elementary OS team is aware of the issue. "We are actually in the process of rewriting Mail to first reach feature parity, and then soon after to enable nicer features like this. Instead of the custom Geary IMAP engine, the new Mail is using LibCamel and EDS, which means tying into contact and calendar data should be much more straightforward," said Blaede.

In the meantime, I resorted to using Thunderbird, which has contact and calendar extensions.

Despite elementary's focus on design elements, I found inconsistency in the maximize and minimize buttons. In fact, there is no minimize button. At the same time, the maximize and close buttons were at the opposite edge of the window. It was frustrating in the beginning, but soon I discovered that you can maximize an app by double-clicking on the top bar of the window. To minimize the app, click on the app icon on the dock.

In spite of the minor inconsistencies, what is appealing about elementary OS is that, unlike every other Linux distro (except for Zorin OS), you don't have to do a lot of fine-tuning to get the look and feel you want. There are no "10 things to do after installing elementary OS." It's a "just works" OS. The creators have already taken care of all the dirty work.

That leads us to the main question…

Who Is It For?

Elementary OS calls itself an OS for macOS and Windows users. Blaede believes that elementary OS is targeted at everyday computer users. However, it does target those who care about their privacy. "We develop elementary OS for users who are fed up with the privacy-invasiveness of modern devices running software like Windows, Chrome OS, and Android," said Blaede.

"We also have had a lot of success with users who are fed up with Apple's continued proprietary approach, planned obsolescence of their devices, and refusal to work with other ecosystems," opines Fore.

It may seem that elementary OS is offering a glass of cold water to those stuck in a Windows and macOS desert.

"We're interested in converting those folks over to using an open source operating system on their desktop or notebook. Over the last year, about 75 percent of people downloading elementary OS do so from a non-Linux operating system, so I'd say we're doing pretty good at reaching those users," said Fore.

It's not just availability of major apps; it's also about support for third-party plugins, hardware, and drivers. So, before switching to elementary OS, do some homework and see if the apps that you need for work are available.

Also keep in mind that elementary OS is a very opinionated OS. It has its own idea of where it wants to go, and there is no room for negotiation. It's more or less like macOS. If you agree with their design principles and the direction they have taken with the curated apps, you will find yourself enjoying this distro.

Conclusion

Elementary OS' added value to the Linux world is bringing the iconic industrial design principle to the platform. As someone who cares a lot about the polish of tools, I think elementary OS is among the most beautiful OSs.

Before you head over to download elementary OS, you should know that elementary OS is not a "band-aid" OS. It's not an OS that popped up to exploit the situation and offer a quick fix for Unity or Gnome. It's a horse in a very long race. They take their time to build things. In my discussion with the team, it was made abundantly clear that they are building a platform for the future, a platform that will become an appealing alternative to macOS, Windows, and Chrome OS. ■■■

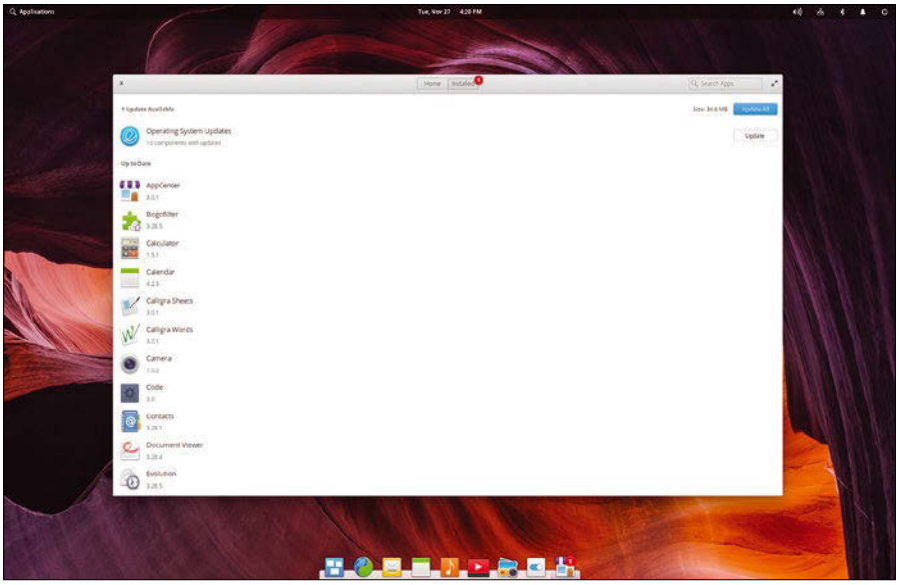


Figure 6: AppCenter is the one stop shop for installing applications and managing updates.

Info

[1] elementary OS: *https://elementary.io/*

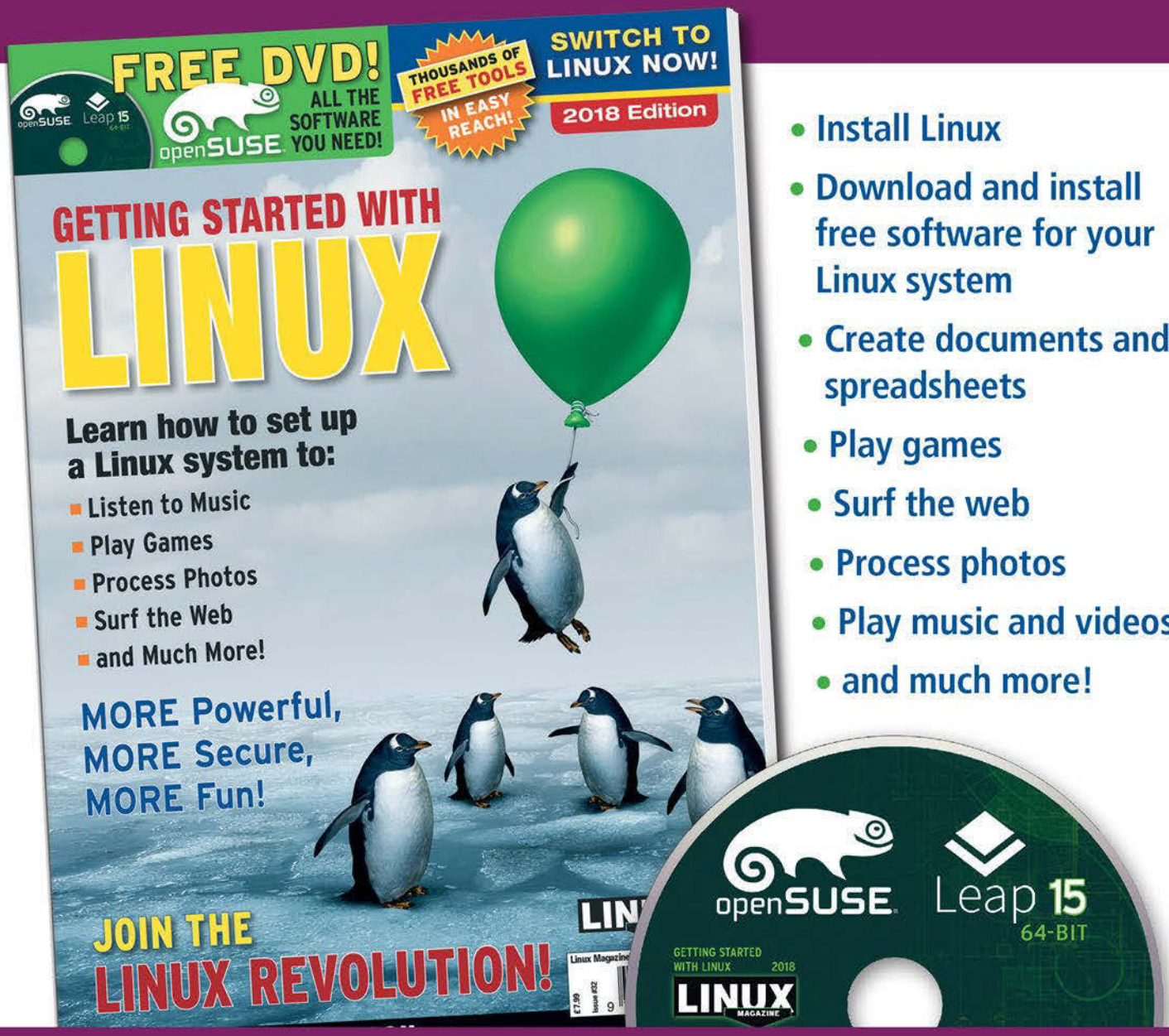
[2] elementary OS HIG: *https://elementary.io/docs/human-interface-guidelines#human-interface-guidelines*

[3] Juno release: *https://medium.com/elementaryos/elementary-os-5-juno-is-here-471dfdedc7b3*

[4] Developer guidelines: *https://developer.elementary.io*

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

This single issue shows beginners how to:



- Install Linux
- Download and install free software for your Linux system
- Create documents and spreadsheets
- Play games
- Surf the web
- Process photos
- Play music and videos
- and much more!

HELP OTHERS
JOIN THE LINUX REVOLUTION!

ORDER ONLINE:
shop.linuxnewmedia.com/specials

Open Source Tools for Writers

The Writer's Toolbox

When it comes to writing, using the right tools can free you up to focus on your content.

By Bruce Byfield

Sooner or later, open source development comes to every field, and tools for working writers are no exception. However, if you search for the topic, you will find the lists of writing tools are full of apps that are no longer in development and have been dropped from most distributions.

Accordingly, here is a list of useful writing apps that are still available as of late 2018. Some have been around for a long time, while others are newer and little known.

BrainDump

Over the last two decades, over half a dozen tools for brainstorming have been released. However, if the proprietary ones are ignored, few free-licensed ones have survived. Technically, BrainDump [1] is one of the casualties, having been removed from Calligra 3.0, apparently because of a lack of developers.

Fortunately, BrainDump remains available in places like the Debian Stable repository. It remains useful in its current state for brainstorming maps that are almost as quick as pencil and paper (Figure 1). Its support for images, charts, and diagrams gives it a versatility that allows rapid, unimpeded development of ideas.

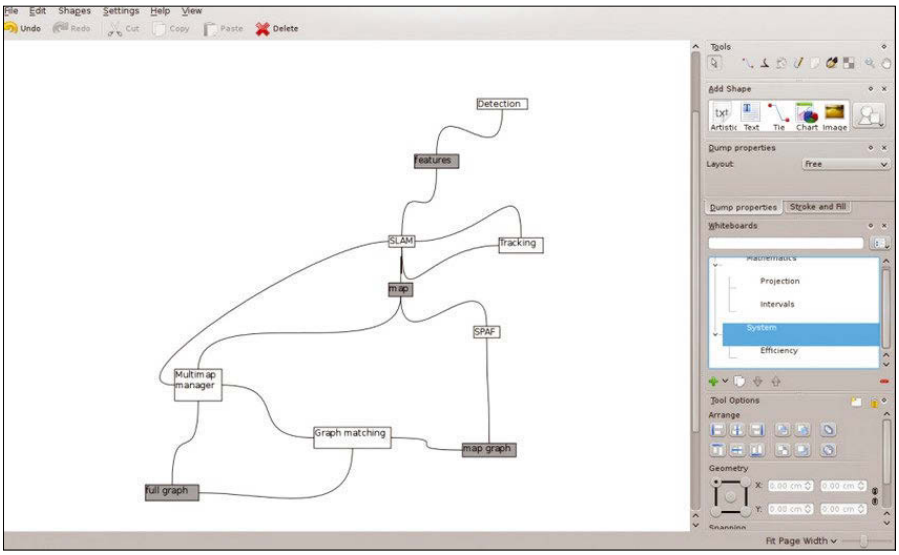


Figure 1: Originally part of Calligra Suite, BrainDump is a brainstorming tool that is likely to be available for a while.

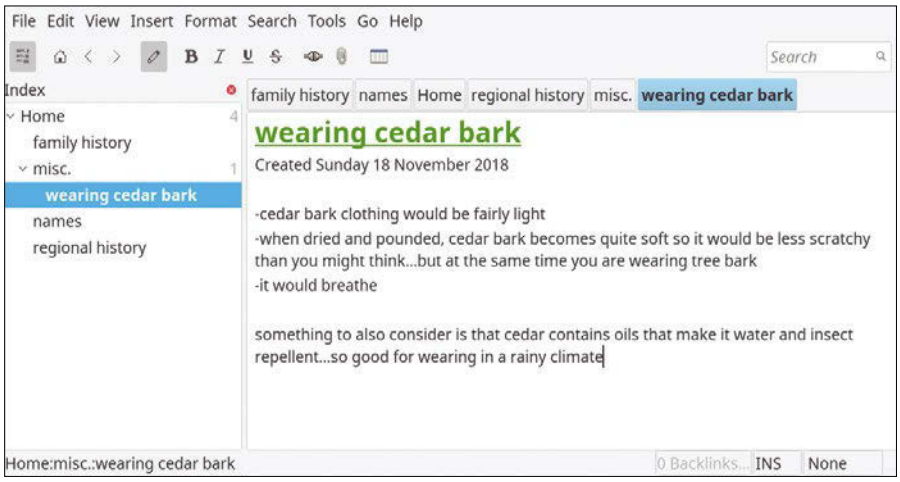


Figure 2: Zim is ideal for storing background material.

As an alternative, brainstormers might also want to look at VYM.

Zim

Longer works often require background material that the writer needs to know but which seldom finds its way into the story. This is especially true of fantasy. Often described as a desktop wiki, Zim [2] is a convenient place to store such information and to link files together for quick reference. For example, I use Zim to store files with information such as character and historical background, as well as names for different cultures in my fantasy novel attempt (Figure 2).

KDE users might use BasKet instead. Although BasKet advertises itself more humbly as an advanced note taker, its capacities are similar to Zim's.

Artha

Artha [3] promotes itself as an open source thesaurus. At first, I saw nothing in the app that suggested any benefit of being open source. Perhaps, I thought, open source's influence will only become evident over time, possibly in the speed with which new words and meanings update it.

Meanwhile, Artha is a comprehensive, local thesaurus with some valuable features (Figure 3). Like the online Thesaurus.com, it includes antonyms and alternate meanings. However, Artha also includes jargon, related words, pertainyms (forms of the word that are parts of speech, such as an adverb based on a noun), and derivatives (for instance, "clearing" for "clear"), as well as sound-alikes and regular expressions. Best of all, when you enter a word for lookup, Artha displays a drop-down list of meanings instead of going directly to an arbitrarily defined core meaning. This drop-down list allows me to use Artha as a concept thesaurus – one based on categories of meaning rather than words – which is by far the most useful structure for writing, although it is rarely seen these days. If that is not enough, Artha also has a hot key feature, which allows users to get a definition of a highlighted word on the desktop.

After discovering all these feature, I only then realized that the evidence of Artha being open source lies in its comprehensiveness – a long-time open source tradition. As soon as I discovered all it could do, within moments Artha became my online thesaurus of choice.

Klipper

Klipper [4] is the clipboard in KDE (Figure 4). What makes it stand out is that it includes a buffer of previously copied or cut items to which it can revert with a couple of clicks on its icon in the system tray. This feature makes it ideal for copy editing when the same replacements are needed repeatedly. If necessary, items can be typed in to the buffer as needed. Why a similar buffer was not added to other desktops years ago is a mystery.

When I was a university instructor, I always told students that, if they had enough knowledge to use a grammar checker properly, then they had no need for one, except possibly to catch typos.

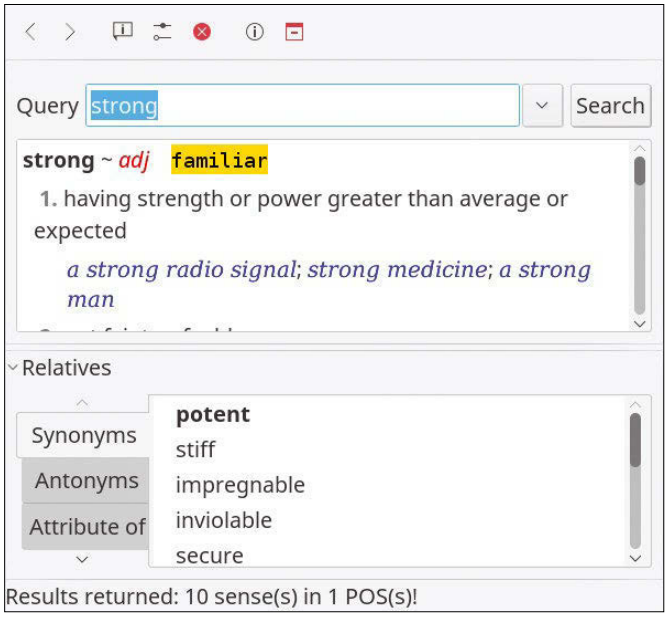


Figure 3: Artha is one of the most comprehensive thesauruses available online.

Figure 4: KDE's long-time clipboard, Klipper, supports multiple items, which is useful in editing.

```
b@nanday:~/work/journalism/2018/11-november$ diction ./tools-for-writers.txt
/tools-for-writers.txt:3: [However], if you search for the topic, you will find the l
ists of writing tools are full of apps that are no longer in development, and have bee
n dropped from [most] distributions.

/tools-for-writers.txt:5: Some have been around for a long time, [while] others are n
ewer and little known:

/tools-for-writers.txt:9: [However], if the proprietary ones are ignored, few free-li
censed ones have survived.
```

Figure 5: Diction shows where grammatical rules might apply, rather than suggesting changes.

Too often, the helpful suggestions can lead the unwary to further mistakes.

Diction

Diction [5] is an exception to this rule – and a surprising one, considering that it runs from the command line (Figure 5). What makes Diction an exception is that it flags words that are common in grammatical errors and simply gives you the general rules associated with them, leaving you to decide whether to apply them or not. Instead of trustingly clicking a button to make a change, users have to stop and think whether each grammatical rule applies. Mistakes are less likely, and, confronted with these rules, users may actually learn a few points about grammar.

Starting with a plain text file, Diction has the options to flag words associated with common beginner's mistakes and/or to suggest better wording. And Diction is thorough, averaging in my writing about 170 suggestions for about 2,000 words (most of which, I am happy to say, were false flags). In my experience, such thoroughness is unparalleled in grammar checkers, which makes the extra step of converting a file to plain text for the check well worth it.

Calibre

Many Linux users know Calibre [6] as a convenient app for storing and launching ebooks. However, if you are producing ebooks yourself, Calibre is also a one-stop app for editing ebooks and exporting them to multiple formats (Figure 6). The simplest way to edit ebooks is to write them in LibreOffice and export them to Calibre. Then, you can use Calibre to edit metadata, add graphics and tables of content, add new sections, and output the ebook to every major format. Armed with a knowledge of CSS, you can right-click to edit the raw code and validate it.

Calibre would be even more powerful if it included a guide to CSS tags. However, even so, it's a basic necessity for writers who intend to self-publish online.

LibreOffice Writer

LibreOffice Writer [7] may seem like an obvious choice, considering that it is a full-featured office suite. However, among those tools are several that are especially useful for professionals.

Admittedly, few editors accept manuscripts in LibreOffice's default Open Document Format (ODT). However, formatting for manuscripts is simple enough that exporting files to MS Word format is no trouble. Moreover, Writer also exports to PDF (Figure 7), with enough options to give you full control over the process. The last few releases have even started to support exports to ePub, the leading free ebook format. Although the support for ePub within Writer is still limited, ODF files can be

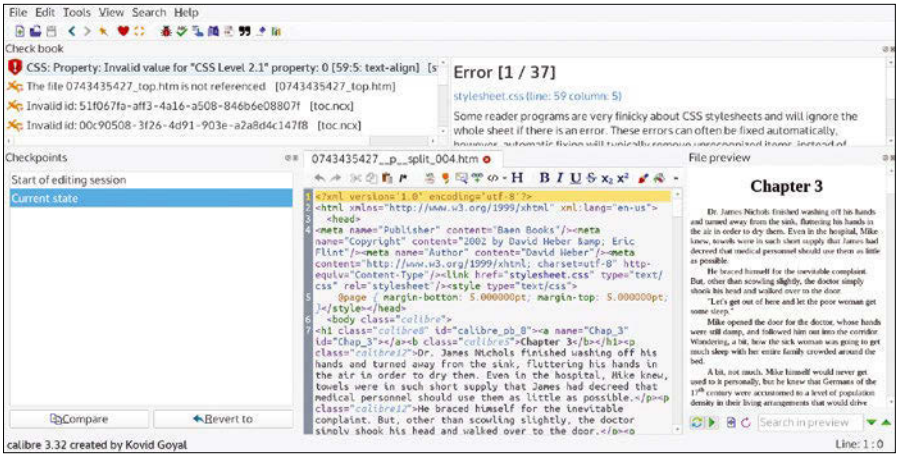


Figure 6: Besides being an ebook manager, Calibre also has tools for editing.

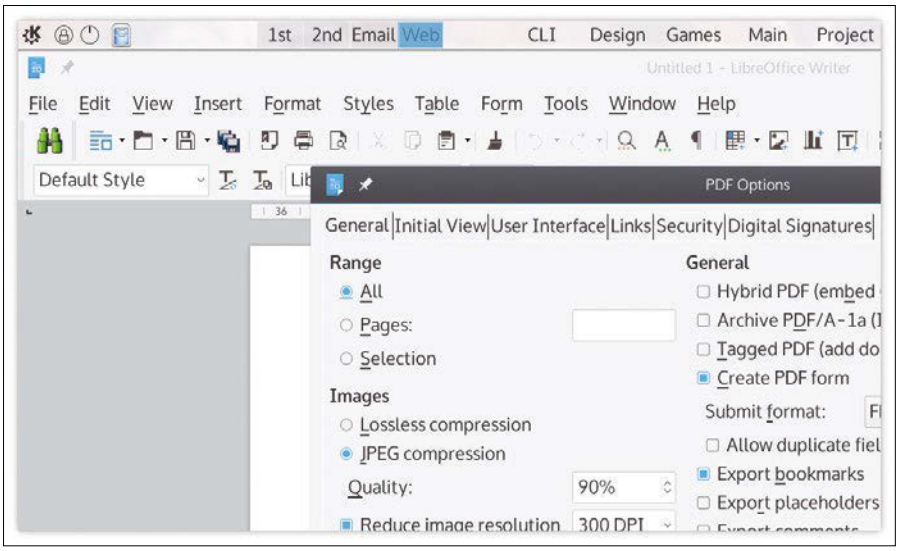


Figure 7: Extensive PDF options are one of several reasons for writers to prefer LibreOffice.

imported to the Calibre ebook manager and then converted with acceptable quality to ePub, Kindle's MOBI, or any other popular ebook format.

In addition, Writer supports comments and tracking changes, two features that enable collaboration of exactly the kind that happens between writers and editors or critiquing readers. Using these tools, writers can accept or reject revisions and easily access revisions from within their manuscripts.

For those who are writing very long books, Writer has Master Documents, which are documents that consist of multiple files. These files can be edited separately, which reduces memory requirements and allows writers to work on different parts of the complete document at the same time.

Likewise, professionals may find features like AutoText and personal dictionaries for spell checking and hyphenation useful. Should you want to self-publish, either online or to hard copy, Writer also has the tools for professional layout and design unmatched by other word processors. With this array of tools, Writer is indispensable for serious writing.

What's Missing

This list of applications is what I consider the best of the best. For example, there are countless text editors and word processors that I might mention. However some are free to use, but do not have free licenses. Neither have I mentioned any online tools, for the simple reason that when you are a writer with deadlines, the risk of Internet connection problems is too great, even though this only occasionally happens. Local apps are simply more reliable.

Also, I have left out most so-called writers' applications. Some, like FocusWriter, promise a distraction-free writing environment that I can get more conveniently in Bluefish or Vim, or even LibreOffice by using styles and templates – and at the expense of extra time spent reformatting for submission.

Another category I have left out are databases for fiction like bibisco. Such tools claim to help writers by peppering them with questions about characters, settings, unnecessary links, and organization. I remain deeply skeptical about such tools, because I have yet to hear of a professionally published writer who uses them. Just as importantly, they take much of the joy from writing for me, reducing the experience to something more akin to filling out a seemingly endless survey.

In the end, writing is about writing – or, failing that, streamlining necessary research so that you can return to writing as soon as possible. Properly used, the applications mentioned here should help you do just that. ■■■

Info

[1] BrainDump: *https://packages.debian.org/stretch/braindump*
[2] Zim: *http://zim-wiki.org/*
[3] Artha: *http://artha.sourceforge.net/wiki/index.php/Home*
[4] Klipper: *https://userbase.kde.org/Klipper*
[5] Diction: *https://www.gnu.org/software/diction/*
[6] Calibre: *https://calibre-ebook.com/*
[7] LibreOffice Writer: *https://www.libreoffice.org/discover/writer/*

The sys admin's daily grind: DNSDiag

Hello, Who Are You, Won't You Tell Me Your Name!

If some transactions take an inexplicably long time, you don't have to blame yourself for the delayed transmission of user data. Name resolution issues might be to blame. Sys admin Charly has three tools to study the DNS server. *By Charly Kühnast*

Pure randomness took me by the hand recently and led me to `dnsping`, `dnstraceroute`, and `dnseval`. The tool collection for name resolution is entitled DNSDiag [1]. You need Python 3 and `pip3` to install and run the trio and `sudo` to let it create ICMP sockets.

`dnsping` lives up to its name, repeatedly querying a DNS server and displaying the response times. The hostname to be resolved is a mandatory parameter. `dnsping` prompts you for the system's default name server, which can be changed using `-s <nameserver>`. After typing

```
sudo dnsping.py -v ➥
  -s 8.8.8.8 linux-magazine.com
```

I queried a public DNS server from Google. Its responses took 20 milliseconds to reach me, four times more than my provider's DNS.

`dnseval` queries several servers in parallel. As a competition judge, it presents the results so that you can immediately see which server responds fastest or slowest. I redirected the list of servers to be checked into a text file, with one server in each line. Lists of public DNS servers are easy to find; I used [2] and took the first five servers from the list. The call looks like this:

```
sudo dnseval.py -f ./liste.txt ➥
  -c 5 linux-magazine.com
```

The result in Figure 1 shows a remarkable discrepancy between minimum and maximum response times.

Highwayman?

`dnstraceroute` determines the path my DNS query takes to reach the target. By comparing this with a classic ICMP `traceroute`, I can identify an attacker trying to kidnap my DNS queries. My test call is:

```
sudo dnstraceroute.py --expert --asn ➥
  -C -s 8.8.4.4 linux-magazine.com
```

The result is shown in Figure 2. The `--expert` parameter provides tips if something seems to be suspicious in the output – for example, if the target server is only a hop away from a private IP address (RFC 1918). False alarms also occur if you are not working on a cloud server, but locally, and a DNS cache such as Dnsmasq [3] is running on the router.

For each hop, the `--asn` parameter shows you the autonomous system providing the network for the address. I can thus quickly see where the process crosses my provider's boundaries. ■■■

Author

Charly Kühnast manages Unix systems in the data center in the Lower Rhine region of Germany. His responsibilities include ensuring the security and availability of firewalls and the DMZ.

Info

[1] DNSDiag *https://dnsdiag.org*

[2] Free DNS: *https://www.lifewire.com/free-and-public-dns-servers-2626062*

[3] Dnsmasq: *http://www.thekelleys.org.uk/dnsmasq/doc.html*

server	avg(ms)	min(ms)	max(ms)	stddev(ms)	lost(%)	ttl	flags
209.244.0.3	225.456	5.044	630.450	276.140	%0	300	QR -- -- RD RA -- --
64.6.64.6	120.021	96.787	203.152	46.536	%0	299	QR -- -- RD RA -- --
8.8.8.8	19.474	11.814	26.253	5.138	%0	53	QR -- -- RD RA -- --
9.9.9.9	13.115	4.624	18.752	6.193	%0	300	QR -- -- RD RA -- --
84.200.69.80	165.459	17.262	568.930	237.859	%0	300	QR -- -- RD RA -- --

Figure 1: The `dnseval` utility reveals a large difference between the minimum and maximum response times.

```
dnstraceroute.py DNS: 8.8.4.4:53, hostname: linux-magazin.de, rdatatype: A
1       172.31.1.1 (172.31.1.1) 0.755 ms
2       10276.your-cloud.host (88.99.159.78) [AS24940 HETZNER-AS, DE] 1.137 ms
3        *
4       spine2.cloud1.nbg1.hetzner.com (85.10.237.197) [AS24940 HETZNER-AS, DE] 1.441 ms
5       core12.nbg1.hetzner.com (85.10.250.213) [AS24940 HETZNER-AS, DE] 0.941 ms
6       core0.fra.hetzner.com (213.239.252.21) [AS24940 HETZNER-AS, DE] 32.872 ms
7       72.14.218.176 (72.14.218.176) [AS15169 GOOGLE - Google LLC, US] 4.130 ms
8       108.170.231.245 (108.170.231.245) [AS15169 GOOGLE - Google LLC, US] 3.892 ms
9       216.239.59.123 (216.239.59.123) [AS15169 GOOGLE - Google LLC, US] 4.075 ms
10      google-public-dns-b.google.com (8.8.4.4) [AS15169 GOOGLE - Google LLC, US] 20.315 ms

=== Expert Hints ===
 [*] No expert hint available for this trace
```

Figure 2: `dnstraceroute` tracks the path of a name resolution query.

A flexible, command-line pager

Most Is More Than Less

If you like to customize your command-line file pager, check out most. *By Bruce Byfield*

Many files you may want to view from the command line require more than a single screen to display. As a result, they scroll by too quickly to read and are most conveniently read with a file pager. The oldest surviving pager is the `more` [1] command, which has a limited set of options. When a more sophisticated pager was released, it was called `less` [2], because, as the old saying goes, "less is more." A third-generation pager continues the joke by being called `most` [3], apparently on the ground that most is more than less.

As a joke, the name ranks for better or worse with recursive acronyms like GNU's Not Unix (GNU). However, as a pager, `most` offers far more flexibility than `less` or `more` in viewing, navigation, and customization.

Enabling Display Behaviors

`most` is apparently written with the assumption that it will be used most of the time to scan a file's contents. If you simply enter the basic command followed by the file to view, you will notice that the display has some peculiarities (Figure 1). For example, if a line has more characters than `most`'s terminal width, the lines are truncated with a dollar sign (`$`) as the last character in the line. To view the rest of the line, you must use the right arrow key. If you use the `-w` option to wrap the line, then a backslash (`/`) is added unless you also add the `-d` option. So, should you want to view a file without these extra characters, the basic command structure should be:

```
most -wd FILE
```

However, you can also enable or disable other aspects of the display. For instance, to save space, you can use the `-v` option to have control characters display with a prefix of `^` rather than `Ctrl-CHARACTER`. Similarly, the `-t` option renders tabs as `^I`. If a file contains several blank lines in a row, you might add `-s` to replace them with a single blank line.

If you are referring to the file in detail and moving back and forth in it, another useful option is to add line numbers using `+lineno`. Still another option is to view the file as binary using `-b` (Figure 2). When you want to find a certain passage in a long file, you can also use the option `+/STRING` to jump directly to a keyword or phrase, further refining the search by making it case-sensitive with `-c`.

If you are looking at files to decide what to delete, still another useful option is `+d`. With this option enabled, you can activate a toggle once the file is open to delete it after confirming the action.

Other options that you can toggle while viewing a file include `:c` for case sensitivity and `:o` or `:0`, which allows you to add `b` (binary), `d` (selective display), `t` (tab), `v` (verbose), or `w` (wrap). Without these toggles, upon discovering a need for these options, you would have to exit and edit the command to use them.

Navigating Basics

While viewing a file in `most`, you can navigate with just the Up and Down arrow keys and close the file with *Q*, Ctrl + X, or Ctrl + C – whichever you prefer. Each time you press an arrow key, the cursor moves one line. However, if you enter a number before pressing an arrow key, you will move that number of lines.

By default, `most` displays in a 60-column line, truncating the rest of the line. You can view the rest of the line with the right arrow plus the Tab key, or the left arrow plus Ctrl + B, depending where the cursor is on the line.

You can make larger jumps in the displayed file, too. T + Esc takes you to the top of the file and B + Esc to the bottom. When the file displays over several windows, you can use the structure `NUMBER%` (such as 3%) to jump to a specific spot in the file. If you are moving back and forth in a file, you can use Ctrl + @ or Ctrl + K to add a bookmark to the current line. Once you have one bookmark,

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. You can read more of his work at *http://brucebyfield.wordpress.com*

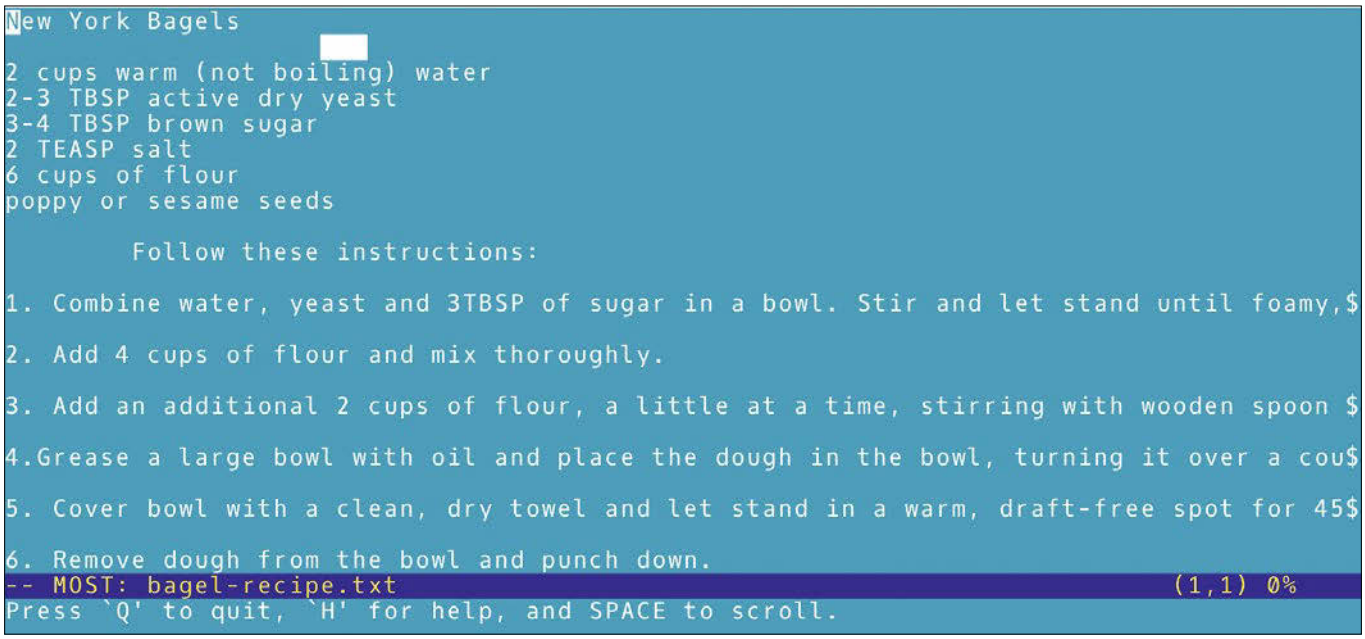


Figure 1: Some of `most`'s features include the status information on the bottom, the `$` sign that marks truncated lines, and options for line numbering and displaying tabs as `^I`.

you can return to it by pressing Ctrl + X or Ctrl + K + Enter, which also creates another bookmark at your current position if one does not already exist.

Working with Multiple Files

With all these options for viewing a file, you might almost overlook the fact that a single instance of the command can deal with multiple files. Working with multiple files isn't made easier by the fact that `most`'s man pages eke out information by separate snippets. Some users, too, might wonder whether the reference to displaying each file refers to a separate terminal or a separate screen of information (which is actually the case).

However, once you piece the information together, working with multiple files is straightforward. All that is required is a space-separated list of filenames in the command. If you want, you can specify the same file twice, so you have convenient access to different passages.

To move between multiple files, you can toggle `:n` to move to the next file in the list, selecting it by pressing any key except *Q*. Each file can have its own separate options for display, and can be searched separately. However, you can toggle *L* to lock each file, which is indicated by an asterisk (*) on the left of the status bar at the bottom of the display. All windows that are locked will scroll together, a feature that can help you compare two copies of a file for differences.

Customizing most

In addition to the viewing and navigation features, `most` also has extensive

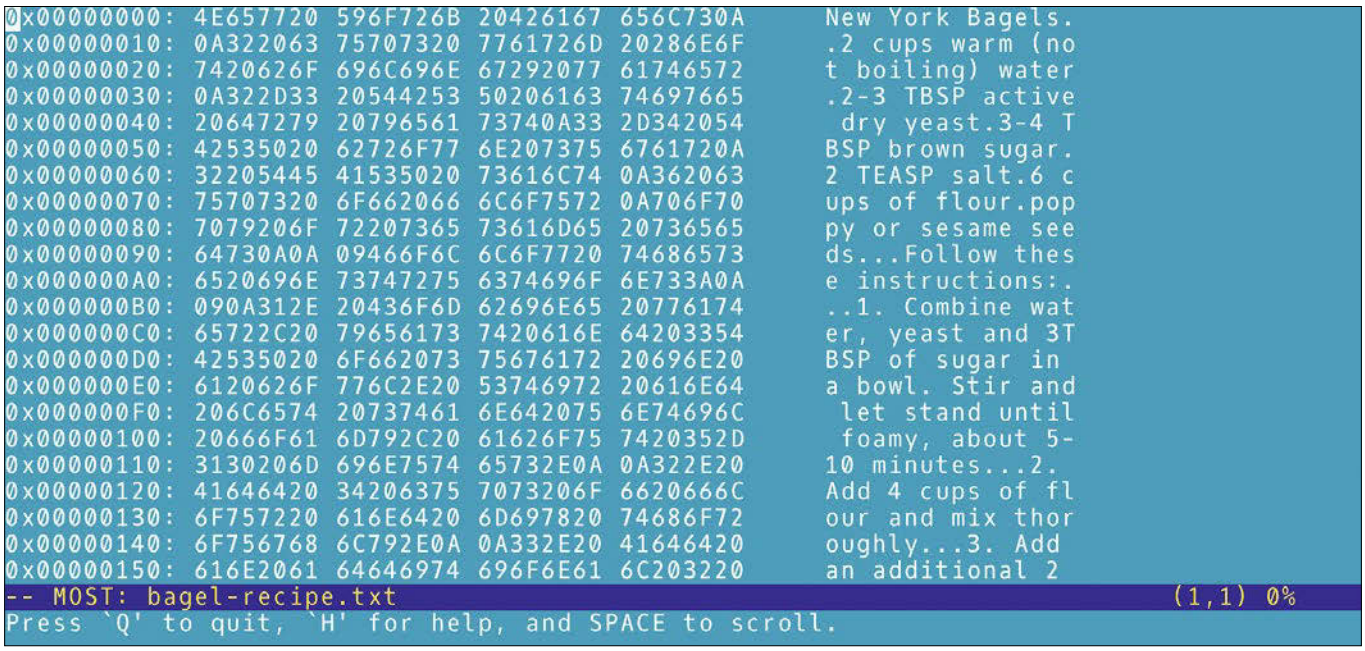


Figure 2: `most` also supports viewing files in binary format.

customization options. In several instances, these options are set by modifying environmental variables. For example, by default, `most` displays with the foreground and background colors set in your terminal. However, if you set `most` as your default pager in your environments, the command can display results in color, including on man pages. Enter the commands:

```
PAGER=most
export PAGER
```

These two commands apply only to the current terminal or tab. To set colors permanently, you must set up a configuration file (see below).

If you want to turn off colors, run `most -C`, or open another terminal with *t*.

Similarly, if you press the *E* or *e* key while viewing a file in `most`, you can open the file in a text editor. The text editor is defined by adding its name to the definition of the `MOST_EDITOR` or `SLANG_EDITOR` variable, using the same structure as in defining `most` as your system's default pager. If no editor is defined, then `most` uses Vim, assuming it is installed.

Perhaps even more useful, you can modify `most`'s default options by modifying the `MOST_SWITCHES` variable. For example, if you want to view all of a line longer than `most`'s display, you could enter:

```
define MOST_SWITCHES "-w"
```

As you might expect, you can override any default using options added at the command prompt.

Other permanent behavioral changes can be made by creating a configuration file. By default, `most` does not install with a configuration file, and many users won't likely miss one. However, if you want to configure keybindings or set colors permanently, you can create a system configuration file named `/etc/most.conf`, or a personal configuration file named `.mostrc` in your home directory.

Before you change an existing keybinding, check the man page to see whether it is already in use. If it is, start the file with a command like:

```
unsetkey "^KEYSTROKE"
```

The circumflex (^) stands for the Ctrl key, which is the command key that `most` uses for keybindings. Put the custom keybindings below all the `unsetkey` entries in this format:

```
setkey "PURPOSE" "^KEYSTROKE"
```

For example, `setkey "up" "^P"` will move the cursor up when Ctrl + P is pressed. I have yet to find a list of `most`'s purposes, but mostly they seem to be common sense.

A configuration file can also be used to set permanent colors using standard ANSI color codes with the command:

```
color OBJECT-NAME FOREGROUND-COLOR ➥
  BACKGROUND-COLOR
```

`OBJECT-NAME` can be filled in with `status` (for the status line), `underline`, `overstrike`, or `normal` (anything else) (Figure 3).

The Default Linux Should Have

`most` was first released in 1999. While many major distributions carry it, it is not installed by default, and I have to wonder why. Part of the reason may be timing. The more popular `less` was released when the basic commands were still being developed, so it was easily accepted. By contrast, by 1999, Unix and Linux users were set in their ways, so `most` has had a harder time gaining acceptance.

Or, possibly, `less` is good enough for most purposes. After all, how often do users add options to `less` when piping another command through it?

However, if you make more demanding use of a pager, or simply like to do your computing in your own way, `most`'s advantages should be more than enough to convince you that `less`, like `more`, has survived past its prime. ■■■

Info

[1] more: *http://man7.org/linux/man-pages/man1/more.1.html*
[2] less: *http://man7.org/linux/man-pages/man1/less.1.html*
[3] most: *https://linux.die.net/man/1/most*

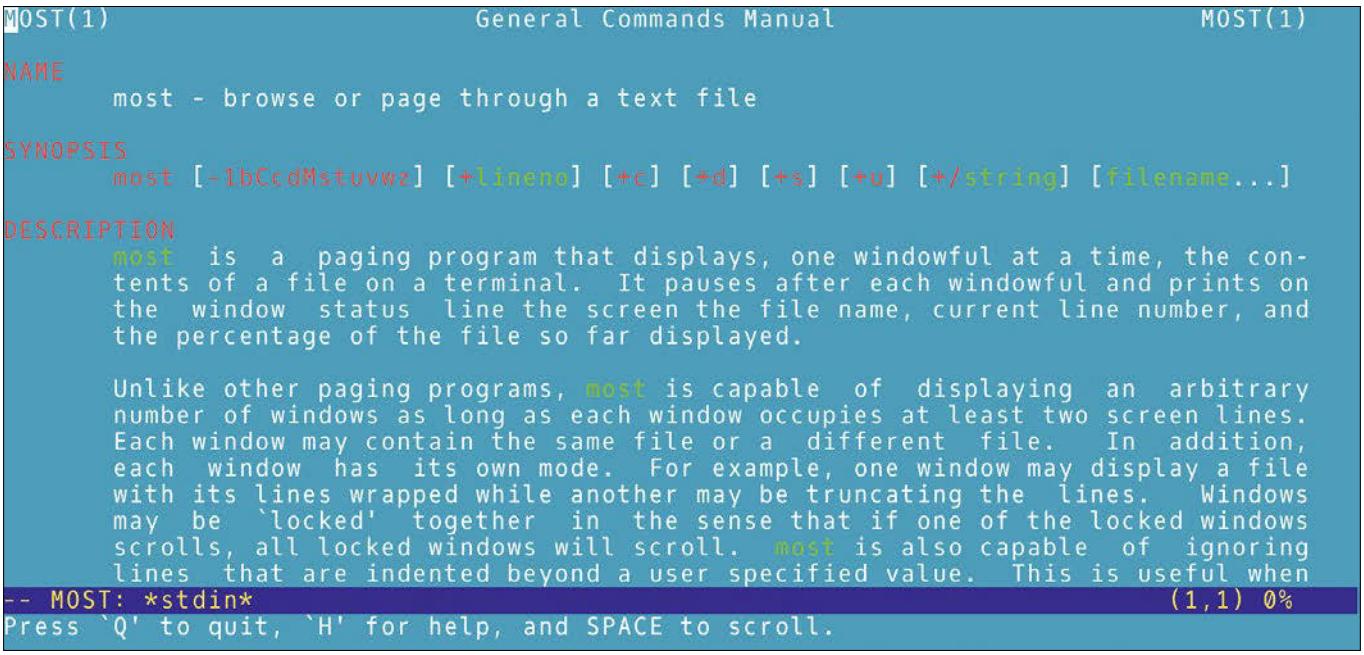


Figure 3: A man page displayed using `most` with colors enabled.

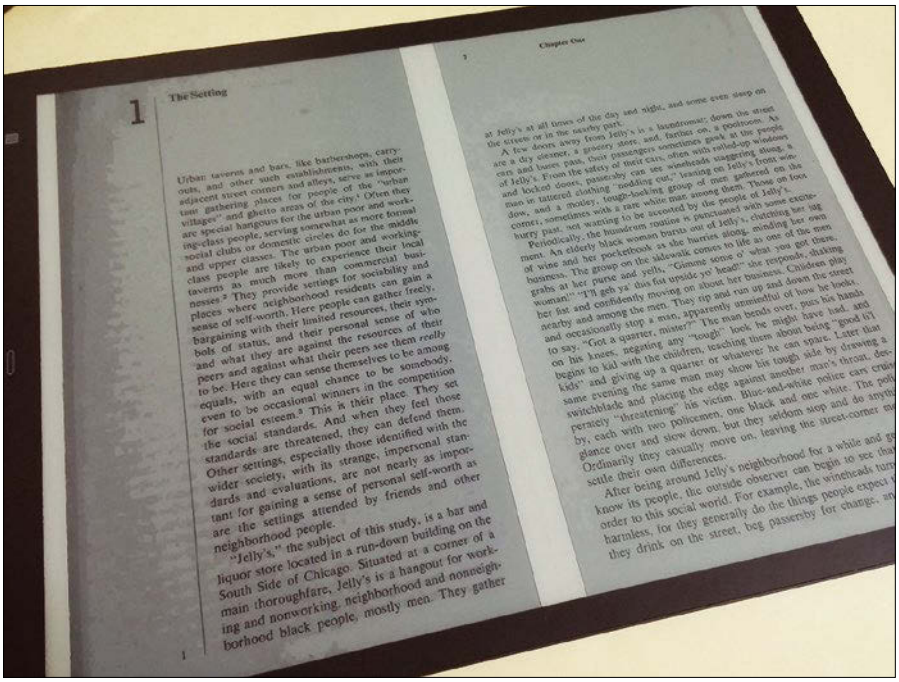
Purifying your scanned PDF files

New View

Having trouble reading that scanned PDF? You can add a little more contrast with some help from ImageMagick. *By Răzvan T. Coloja*

Gone are the days when you needed to go to the library for a book. Now you can download the book electronically, load it on your e-reader or tablet, and start enjoying it. But all electronic books are not equal. Particularly infuriating are electronic books that are actually scanned images of old print books. Scanned images of old books, which typically come in PDF format, are difficult to read on a black-and-white E Ink screen, where fading text and yellowing pages appear as blurs, blotches, and dark-gray backgrounds.

Luckily, you can clear up that blurry scanned image with a few tricks from ImageMagick. This article describes a method you can use to spruce up a scanned electronic book. *Note: If you obtained the book from a lender or through*



Lead Image © Sasin Paraksa, 123RF.com

Figure 1: The scanned PDF book as it displays on a Sony DPT-RP1/B e-reader's 13" E Ink screen.

Author

Răzvan T. Coloja is a Romanian psychologist who has worked both as an administrator since 1997 and as the editor-in-chief of the now-defunct Romanian IT magazine *MyLINUX*. He was the executive editor of the IT-centric magazine *MyCOMPUTER* for three years, and he has worked as an editor for *CONNECT* magazine. He has published Linux articles in international print magazines and online.

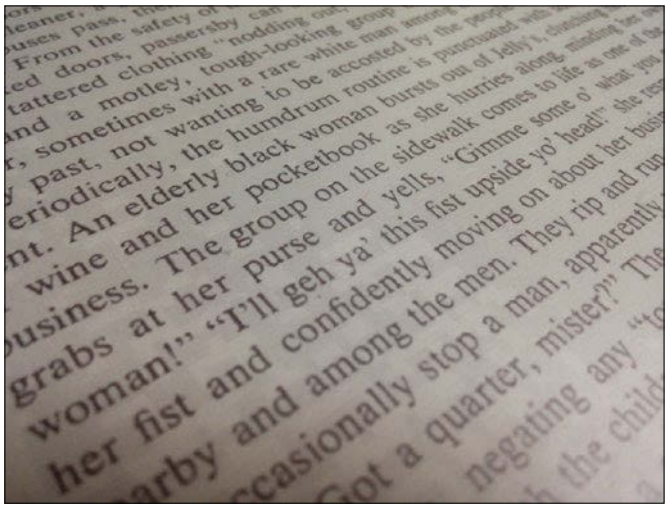


Figure 2: The contrast between the text background is insufficient, impeding readability.

another vendor, be sure the license supports this type of file manipulation.

Getting Started

I needed a copy of a sociology book, *A Place on the Corner*, by Elijah Anderson; the only place I could find it in electronic format was The Internet Archive. They had a scanned copy, so I loaded it on my Sony DPT-RP1/B e-reader, but the text was difficult to read (Figure 1). Dark spots appeared on the page, and the text was just a bit darker than the background, with poor contrast (Figure 2).

Hoping to find a better view, I installed the `pdfimages` package on my Ubuntu 18.04 system:

```
sudo apt install pdfimages
```

`pdfimages` is used to extract images from PDF files and save them as Portable Pixmap (PPM), Portable Bitmap (PBM), or another format. Since the scanned pages in the book are image files, all you need to do is drop the PDF into an empty folder and enter the following command:

```
pdfimages A_place_on_the_corner.pdf
```

This command will generate a large amount of PPM and PBM image files (Figures 3 and 4). The PPM files contain all the shadows and imperfections of the scanned pages, and the PBM files are a negative, clean, white-on-black version. Now all you need to do is reverse the colors in the PBM negatives. You don't need the PPM files, so you can remove them:

```
rm *.ppm
```

To maintain quality, convert all remaining PBM files into PNG format using `mogrify`, which is part of the ImageMagick package. (See also the "More Memory?" box for using ImageMagick with large files.) First be sure to install ImageMagick:

```
sudo apt install imagemagick
```

`mogrify` is used to manipulate graphic files: rotate, crop, flip, blur, and join. You can also use `mogrify` to convert from one format to another. Search for all PBM files in the folder and use `mogrify` to convert them to PNG format automatically:

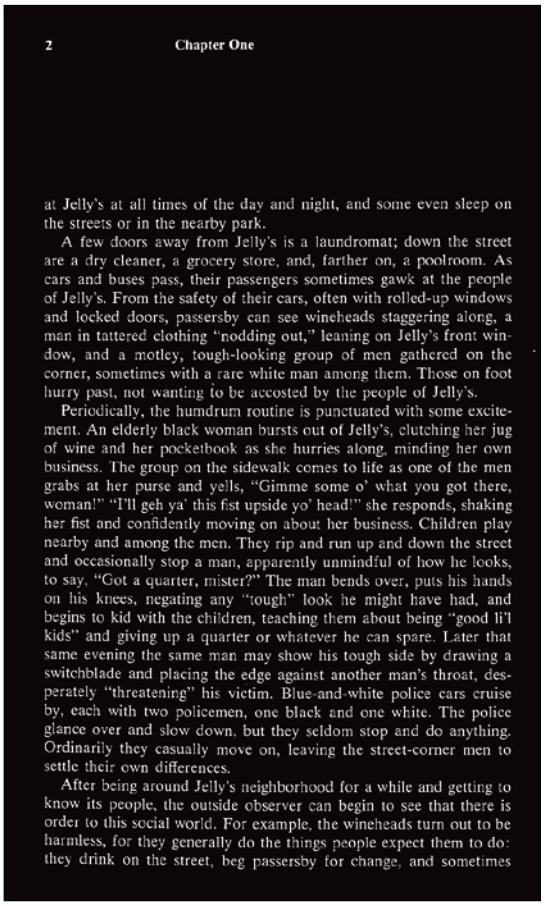
```
find -name '*.pbm' -print0 | xargs -0 -r mogrify -format png
```

Now that the pages of the book are in PNG format, you don't need the PBM files anymore, so you can delete them:

```
rm *.pbm
```

`convert` is another command-line interface tool shipped with ImageMagick that does about the same thing as `mogrify`, but it is also able to invert the colors of

Figure 3: This is how a PPM file extracted from a scanned book will eventually look.



2 Chapter One

at Jelly's at all times of the day and night, and some even sleep on the streets or in the nearby park.

A few doors away from Jelly's is a laundromat; down the street are a dry cleaner, a grocery store, and, farther on, a poolroom. As cars and buses pass, their passengers sometimes gawk at the people of Jelly's. From the safety of their cars, often with rolled-up windows and locked doors, passersby can see wineheads staggering along, a man in tattered clothing "nodding out," leaning on Jelly's front window, and a motley, tough-looking group of men gathered on the corner, sometimes with a rare white man among them. Those on foot hurry past, not wanting to be accosted by the people of Jelly's.

Periodically, the humdrum routine is punctuated with some excitement. An elderly black woman bursts out of Jelly's, clutching her jug of wine and her pocketbook as she hurries along, minding her own business. The group on the sidewalk comes to life as one of the men grabs at her purse and yells, "Gimme some o' what you got there, woman!" "I'll geh ya' this fist upside yo' head!" she responds, shaking her fist and confidently moving on about her business. Children play nearby and among the men. They rip and run up and down the street and occasionally stop a man, apparently unmindful of how he looks, to say, "Got a quarter, mister?" The man bends over, puts his hands on his knees, negating any "tough" look he might have had, and begins to kid with the children, teaching them about being "good li'l kids" and giving up a quarter or whatever he can spare. Later that same evening the same man may show his tough side by drawing a switchblade and placing the edge against another man's throat, desperately "threatening" his victim. Blue-and-white police cars cruise by, each with two policemen, one black and one white. The police glance over and slow down, but they seldom stop and do anything. Ordinarily they casually move on, leaving the street-corner men to settle their own differences.

After being around Jelly's neighborhood for a while and getting to know its people, the outside observer can begin to see that there is order to this social world. For example, the wineheads turn out to be harmless, for they generally do the things people expect them to do: they drink on the street, beg passersby for change, and sometimes

Figure 4: This is a PBM file extracted from the scanned book with `pdfimages` utility.

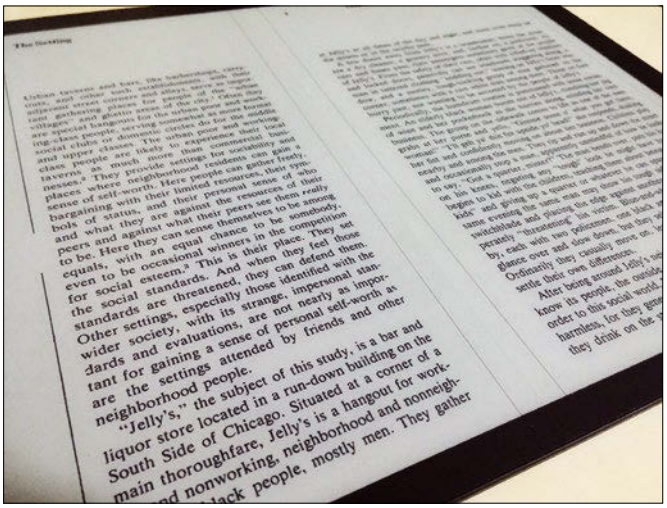


Figure 5: The output PDF file looks better on the E Ink screen and is now easily readable.

an image file. The PNG files are currently in negative, so you can convert them to look like regular book pages using the `-negate` attribute of `convert`. Output the results as JPG files to keep the file sizes low:

```
ls -1 *.png | xargs -n 1 bash -c ➥
'convert "$0" -negate "${0%.png}.jpg"'
```

You can now delete the PNG files and inspect the remaining JPG images.

Listing 1: policy.xml Settings

```
01 <policy domain="resource" name="memory" value="2256MiB"/>
02 <policy domain="resource" name="map" value="512MiB"/>
03 <policy domain="resource" name="width" value="16KP"/>
04 <policy domain="resource" name="height" value="16KP"/>
05 <policy domain="resource" name="area" value="128MB"/>
06 <policy domain="resource" name="disk" value="20GiB"/>
```

Should you feel they need more contrast, you can use the `-level` argument of the `convert` command to bulk-modify the contrast of your files:

```
ls -1 *.jpg | xargs -n 1 bash -c ➥
'convert "$0" -level 60 ➥
"${0%.jpg}.jpg"'
```

Replace `60` with a value ranging from `1` to `100`, with a higher value for more contrast, or reduce the value for less contrast. The next step is to insert the now-cleaned and easily-readable scanned pages into a PDF file. Should you wish to keep the resulting PDF as small as possible, you can batch-resize the JPG files beforehand to 50 percent of their actual size using `mogrify`:

```
ls -1 *.jpg | xargs -n 1 bash -c ➥
'mogrify "$0" -resize 50% ➥
"${0%.jpg}.jpg"'
```

All the files were extracted from the scanned PDF book and numbered in order, and the file names haven't changed through all the conversion steps, so the entire book is ready to insert, page by page, back into a PDF. You can easily add the converted pages back in with:

```
convert *.jpg ➥
a_place_on_the_corner-purified.pdf
```

The result is a PDF file that has crisp black text on a white background, and the difference is noticeable (Figure 5). You can load it on your e-reader or tablet and read it without eye strain.

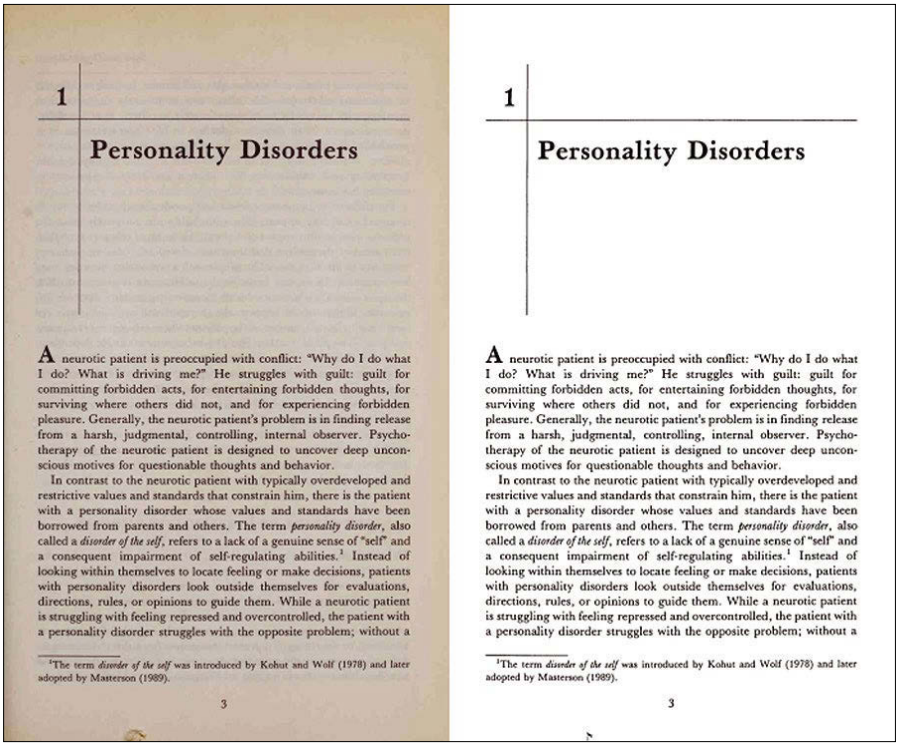


Figure 6: A side-by-side comparison of the purified page with the original. Except for a tiny black spot that remained from an ink drop on the original page, the converted version is black and white, and the faint outline of text from the other side of the printed page is gone.

Conclusion

Using this PDF purifying method, you can get rid of pinkish or yellow backgrounds from scanned documents. Figure 6 shows a comparison of the purified text with the original version of the book. This method will also help you remove other artifacts of the scanning process, such as the faint text lines that appear from the text on the other side of the scanned page, as well as dirty fingerprint marks, light coffee stains, or pencil marks. ■■■

More Memory?

On some Linux systems, you'll receive an error message when you use the ImageMagick command-line tools with large files. This message is due to a memory limitation that you can address by editing the `/etc/ImageMagick-6/policy.xml` file so that the `memory` and `disk` values get more power. Everything should work fine with the `policy.xml` settings that have resource values modified as shown in Listing 1.

Available Now

* 2018 EDITION *

Linux Magazine Archive DVD

Searchable Linux Archive: 19,000 Pages of Practical Know-How

214 issues of *Linux Magazine* on one handy DVD!

ORDER NOW!

shop.linuxnewmedia.com

Python network data visualization

Data Harvest

The Scapy packet manipulation program lets you analyze and manipulate packets to create incident response reports or examine network security. *By Joe McManus*

Most folks have pulled up Wireshark a time or two to troubleshoot an application or system problem. During forensics, packet captures (PCAPs) are essential. Often you are looking at things like top talkers, ports, bytes, DNS look-ups, and so on. Why not automate this process with Python?

Scapy [1] is a great tool suite for packet analysis and manipulation. It is most often talked about in the realm of packet manipulation, but its ability to analyze packets is also top-notch.

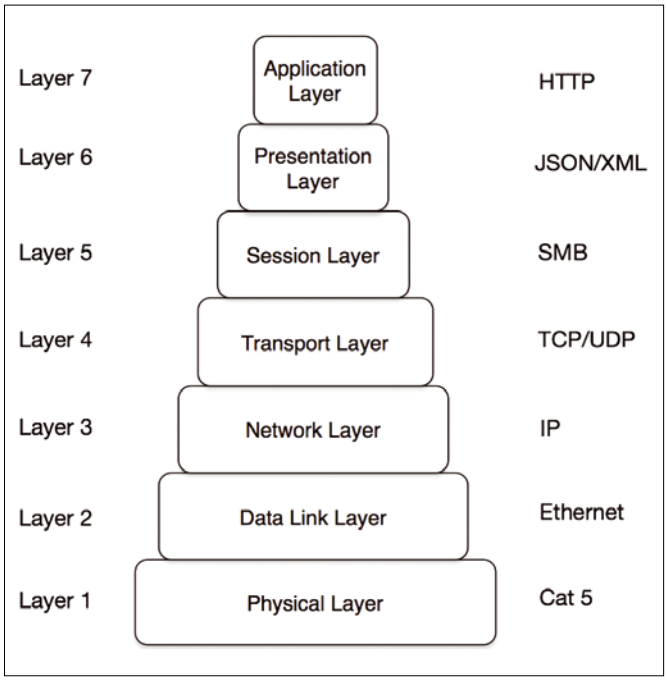


Figure 1: Check OSI Layer.

Make Ready

First, you need to make sure you have Python 3 installed along with the following packages:

```
sudo pip3 install scapy scapy_http ➥
                  plotly PrettyTable
```

To get started, you will want a PCAP to analyze. To capture 1,000 packets and save them to the file `example.pcap`, enter:

Listing 1: Looking for Layers

```
01 #Step 1: Import scapy
02 from scapy.* import all
03
04 #Step 2: Read the PCAP usimg rdpcap
05 packets = rdpcap("example.pcap")
06
07 #Step 3: Loop and print an IP in a packet in Scapy by
   looking at Layer 3
08 for pkt in packets:
09   if IP in pkt:
10     try:
11       print(pkt[IP].src) // Source IP
12     except:
13       pass
```

```
~$ sudo tcpdump -c 1000 -w example.pcap
tcpdump: listening on enp0s3, ➥
         link-type EN10MB (Ethernet), ➥
         capture size 262144 bytes
1000 packets captured
1010 packets received by filter
0 packets dropped by kernel
~$
```

Scapy can handle all parts of the OSI model except Layer 1 (Figure 1). Listing 1 shows the Hello World! of packet reading. To begin, you need to read a raw packet (line 5), see if it has the layer your want (line 9), and then act on it. Because you are using Python, if you try to print out `pkt[IP].src` when no IP is present, Python will throw an error, so you need to wrap it in a `try/except` (lines 10-13).

Listing 2: Adding a Counter

```
01 #Step 1: Imports
02 from scapy.all import *
03 from prettytable import PrettyTable
04 from collections import Counter
05
06 #Step 2: Read and Append
07 srcIP=[]
08 for pkt in packets:
09   if IP in pkt:
10     try:
11       srcIP.append(pkt[IP].src)
12     except:
13       pass
14
15 #Step 3: Count
16 cnt=Counter()
17 for ip in srcIP:
18   cnt[ip] += 1
19
20 #Step 4: Table and Print
21 table= PrettyTable(["IP", "Count"])
22 for ip, count in cnt.most_common():
23   table.add_row([ip, count])
24 print(table)
25
26 +-----------------+-------+
27 |        IP       | Count |
28 +-----------------+-------+
29 |    10.0.2.15    |  482  |
30 |   52.84.82.203  |   93  |
31 |     8.8.8.8     |   82  |
32 |   104.16.41.2   |   76  |
33 |  216.58.216.232 |   30  |
34 |  104.20.150.16  |   20  |
35 |  52.84.133.105  |   16  |
36 |  209.132.181.15 |   16  |
37 | 140.211.169.196 |   15  |
38 |   72.21.91.29   |   12  |
39 |  104.244.46.103 |   12  |
40 +-----------------+-------+
```

Sorting

If you ran the code in Listing 1 with your `example.pcap` file of 1,000 packets, your terminal printed ~1,000 lines, which is obviously not very useful. To improve, you can read all the IPs, append them to a list, then run a counter, and print the results using the `PrettyTable` module (Listing 2). As before, you import Scapy, but now you will also import the collection module and PrettyTable (Step 1). Next, add an empty list, and append (Step 2). Now you can use the counter to loop through the list of IPs and create a count (Step 3); finally, using the `PrettyTable` module, you print out the results in a clean table (Step 4).

Visualize

Now that you know how to read packets and do some counting, you can use the Plotly package to make graphs by building on the last example (Listing 3). First, you have to add the `plotly` import to Step 1 (line 1); then, after going through Steps 2 and 3 as before, you replace Step 4 in the previous example of Listing 2 with new code that creates two new lists to hold *x* and *y* data (Listing 3, lines 4-5) and loops through the IPs again, adding them to the lists (lines 7-9).

By default, Plotly uses its web UI to create charts, but if, like me, you use this data in a incident response situation, you do not want to share that data with a cloud system. Therefore, I use the offline version to plot my data in a new Step 5. When run, it will open your default web browser (Figure 2).

DNS Data

If you modify the previous code slightly, you can print DNS lookups. Instead of `pkt[IP].src`, you use `pkt.haslayer(DNS)`. Again, you create an empty list and append to it; then use Scapy to check for `DNS` and affirm that the packet is a query (with `0` as the QR type) and not a response, which would have a 1 in the QR field. (Listing 4). Again, count and print (Figure 3).

Listing 3: Making Graphs

```
01 import plotly
02
03 #Step 4: Add Lists
04 xData=[]
05 yData=[]
06
07 for ip, count in cnt.most_common():
08   xData.append(ip)
09   yData.append(count)
10
11 #Step 5: Plot
12 plotly.offline.plot({
13   "data":[plotly.graph_objs.Bar(x=xData, y=yData)] })
```

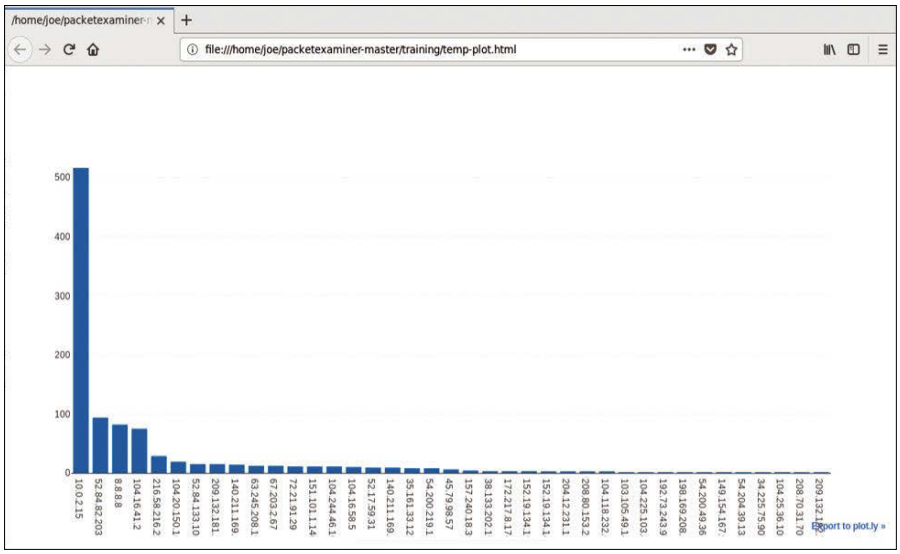


Figure 2: Offline plot of IPs.

Listing 4: DNS Lookups

```
01 from scapy.all import *
02 from collections import Counter
03 import plotly
04
05 packets = rdpcap("example.pcap")
06
07 lookups=[]
08 for pkt in packets:
09   if IP in pkt:
10     try:
11       if pkt.haslayer(DNS) and pkt.getlayer(DNS).qr == 0:
12         lookup=(pkt.getlayer(DNS).qd.qname).
                 decode("utf-8")
13         lookups.append(lookup)
14     except:
15       pass
16
17 cnt=Counter()
18 for lookup in lookups:
19   cnt[lookup] += 1
20
21 xData=[]
22 yData=[]
23
24 for lookup, count in cnt.most_common():
25   xData.append(lookup)
26   yData.append(count)
27
28 plotly.offline.plot({
29   "data":[plotly.graph_objs.Bar(x=xData, y=yData)] })
```

Packets Through Time

At first glance, plotting packets over time is an easy problem to solve. Just grab the packet and use `pkt[IP].len`; however, if you have a reasonable data collection, you will almost always print data of 1500 bytes (the default MTU in most routers), which produces an uninteresting graph. With the *pandas* Python data analysis library, you can make human-readable dates from the packets, which are in epoch (unix time) and then bin the date and time. (Listing 5). First, you have to install *pandas*:

```
sudo pip3 install pandas
```

As before, you create lists to hold data (lines 10-11) and, this time, store the length of bytes in a packet and the timestamp of the packet. Next, you will get the length of the packet with (`pkt[IP].len`) and convert the time using `datetime` (lines 13-25). With the *pandas* library, you convert the list to a *pandas* series and then convert to a timestamp, create the *pandas* dataframe, and organize the data in to two-second bins (lines 21-41). Now you can use Plotly to print the chart. Lines 46-48 add a title with `graph_objs.Layout`. The time (*x*) axis was created during resampling, with the *y* axis data in bytes (Figure 4).

Conclusion

You can do much more with Scapy, such as grab URLs, pull files from PCAPs, and more; by slightly modifying the examples in this article, you can add more features. The open source PacketExaminer project offers a pre-made harness for PCAP analysis [2], and all of the code in these examples can be found in the `training` folder of the repo. If you have any questions, just let me know at *joe.mcmanus@canonical.com*. ■■■

Info

[1] Scapy: *https://scapy.net*
[2] PacketExaminer project on GitHub: *https://github.com/joemcmanus/packetexaminer*

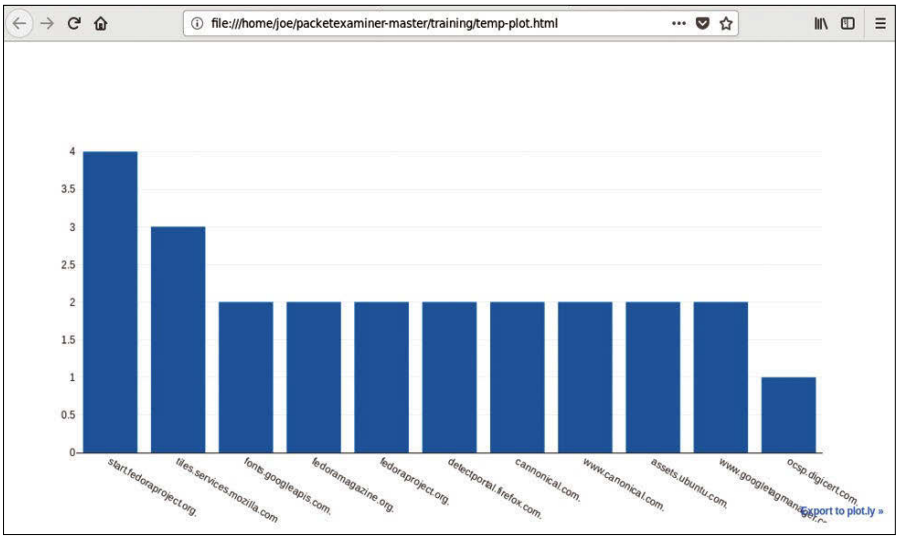


Figure 3: Graph of DNS lookups.

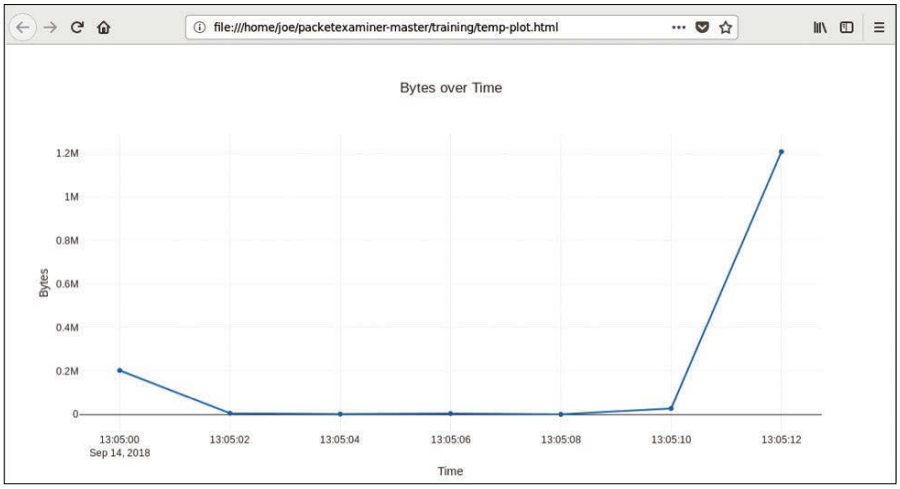


Figure 4: Flow of packets over time.

Listing 5: Using the pandas Library

```
01 from scapy.all import *
02 import plotly
03 from datetime import datetime
04 import pandas as pd
05
06 #Read the packets from file
07 packets = rdpcap('example.pcap')
08
09 #Lists to hold packet info
10 pktBytes=[]
11 pktTimes=[]
12
13 #Read each packet and append to the lists.
14 for pkt in packets:
15   if IP in pkt:
16     try:
17       pktBytes.append(pkt[IP].len)
18
19       #First we need to covert Epoch time to a datetime
20       pktTime=datetime.fromtimestamp(pkt.time)
21       #Then convert to a format we like
22       pktTimes.append(pktTime.strftime("%Y-%m-%d
                        %H:%M:%S.%f"))
23
24     except:
25       pass
26
27 #This converts list to series
28 bytes = pd.Series(pktBytes).astype(int)
29
30 #Convert the timestamp list to a pd date_time
31 times = pd.to_datetime(pd.Series(pktTimes).astype(str),
          errors='coerce')
32
33 #Create the dataframe
34 df = pd.DataFrame({"Bytes": bytes, "Times":times})
35
36 #set the date from a range to an timestamp
37 df = df.set_index('Times')
38
39 #Create a new dataframe of 2 second sums to pass to plotly
40 df2=df.resample('2S').sum()
41 print(df2)
42
43 #Create the graph
44 plotly.offline.plot({
45   "data":[plotly.graph_objs.Scatter(x=df2.index,
           y=df2['Bytes'])],
46   "layout":plotly.graph_objs.Layout(title="Bytes over Time ",
47     xaxis=dict(title="Time"),
48     yaxis=dict(title="Bytes"))})
```

Develop a DIY progress bar

Progress by Installments

Desktop applications, websites, and even command-line tools routinely display progress bars to keep impatient users patient during time-consuming actions. Mike Schilli shows several programming approaches for handwritten tools. *By Mike Schilli*

It's not only hyperactive millennials; even veteran Internet users lose patience when it takes longer than a few seconds for a website to load in the browser. What is especially annoying is when there isn't a clue to what is going on and how long it's going to take. Some 40 years ago, this prompted a smart programmer to invent the progress bar [1], reassuring the user: "Already 10 percent down, 90 to go, and we'll make it through the rest at the following speed."

Hollywood thrillers also love progress bars (Figure 1). When the movie spy downloads sensitive data onto a USB stick, it seems to take forever, and the progress bar keeps ticking really slowly, while the bad guys are approaching, just about to barge in at any moment and blow the spy's cover!

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.

Some Unix command-line tools already feature built-in progress bars. For example, `curl` typically learns at the outset how many bytes a web page contains and, thanks to the `-#` (or `--progress-bar`) option, shows you the real-time data flow:

```
$ curl -# -o data http://...
########### 50.6%
```

As another example, the `dd` tool, which is often used to copy disk data, recently started showing the progress (as of the GNU Coreutils v8.24) if the

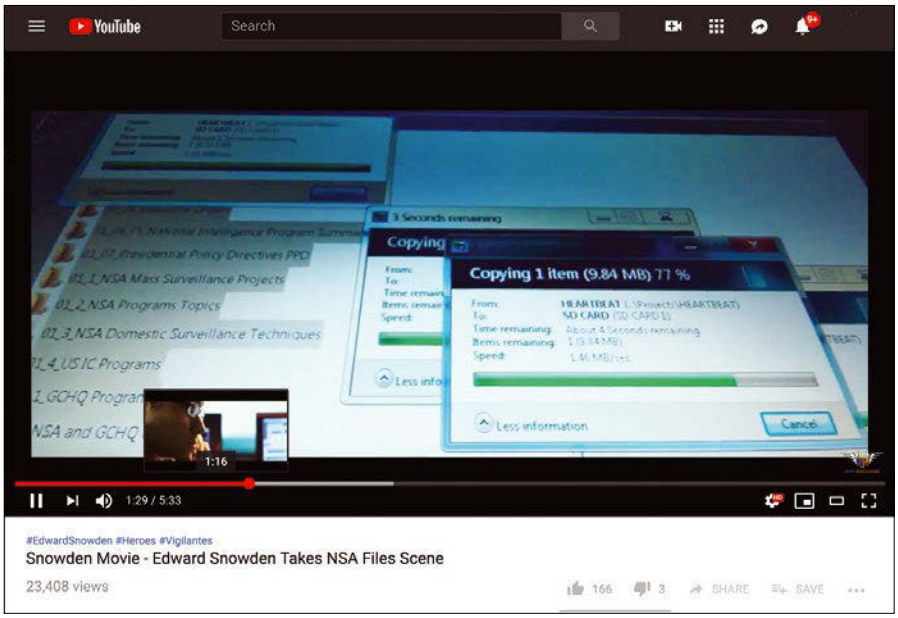


Figure 1: Some Hollywood movies would only be half as exciting without the progress bar.

```
$ dd if=/dev/zero of=out bs=1M count=10000 status=progress
10340007936 bytes (10 GB, 9.6 GiB) copied, 19.0555 s, 543 MB/s
```

Figure 2: Since Coreutils v8.24, `dd` starts showing progress with the `status=progress` option.

```
  # unknown length
$ dd if=/dev/zero bs=1m count=1000 | pv | wc
35GiB 0:00:06 [ 230MiB/s] [       <=>    ]

  # known length
$ dd if=/dev/zero bs=1m count=1000 | pv -s1g | wc
393MiB/s] [===============>       ] 76% ETA 0:00:00
...

  # known length
$ pv bigfile > backup/bigfile
1000MiB 0:00:02 [ 382MiB/s] [========   >] 83%
```

Figure 3: `pv` shows the progress of reading a file as a bar.

user sets the `status=progress` option (Figure 2).

Bash & Co.

Friends of shell programming use the Linux `pv` tool, which helps out utilities without built-in progress bars. Slotted in as an adapter between two sections of a pipe, it uses an ASCII bar to indicate the progress of the data through the pipe by counting the bytes flowing through it. To be able to discover what fraction of the expected total amount has flowed through, and what still needs to be done, it needs to know the total amount of data in advance, to then accurately update the progress bar in regular intervals. It simply calculates the percentage value from the division of the bytes counted so far by the total quantity.

Simply inserted between two pipe sections, however, `pv` knows nothing about the total byte count to be expected and can therefore only count bytes that have already flowed so far (Figure 3, top). If you want to help out `pv` in this role as a "pipe gauge," you can specify the total expected amount of data (if known in advance) with the `-s bytes` option, in which case `pv` draws and updates a nice progress bar.

But if you give `pv` the name of a file, it acts as a `cat` command and can determine how large the file is, before forwarding its data byte by byte, and will display the progress bar correctly without further assistance, as shown by the last backup command in Figure 3.

Doing It Yourself

If you like to put together your own tools, chances are you will find a suitable progress bar library for your programming language on GitHub. For Go, for example, you can use the simple ASCII art displaying tool `progressbar`. The command

```
$ go get github.com/schollz/
  progressbar
```

will retrieve it directly from GitHub and install it in your Go path. Listing 1 [2] shows a simple web client titled `webpgb`, which fetches a URL from the web and at the same time shows in a progress bar how far the download has progressed:

```
$ ./webpgb http://...
13%[##----------][16s:1m49s]
```

Line 7 imports the progress bar library as `pb` into the main program, which uses `os.Args[1]` to gobble up the first command-line parameter passed to it, the URL, which it then fetches off the web with the `Get()` function from the standard `net/http` package.

For piecing the data together from the incoming chunks, line 12 defines a buffer as a Go slice with a length of 4096 bytes; the infinite loop starting in line 36 uses `Read()` to fill it with up to 4,096 characters from the arriving HTTP response part, until the web server is done and sends an `EOF`, which line 40 detects and goes ahead to exit from the `main` function. Meanwhile, line 39 uses `Add()` to refresh the progress bar display by sending it the number of bytes received in the buffer.

Previously, line 23 defined a new bar structure in the `bar` variable and initialized its maximum length to the total

Listing 1: webpgb.go

```
01 package main
02
03 import (
04   "os"
05   "net/http"
06   "io"
07   pb "github.com/schollz/progressbar"
08 )
09
10 func main() {
11   resp, err := http.Get(os.Args[1])
12   buffer := make([]byte, 4096)
13
14   if err != nil {
15     panic(err)
16   }
17
18   if resp.StatusCode != 200 {
19     panic(resp.StatusCode)
20     return
21   }
22
23   bar := pb.NewOptions(
24     int(resp.ContentLength),
25     pb.OptionSetTheme(
26       pb.Theme{Saucer: "#",
27         SaucerPadding: "-",
28         BarStart:      "[",
29         BarEnd:        "]"}),
30     pb.OptionSetWidth(30))
31
32   bar.RenderBlank()
33
34   defer resp.Body.Close()
35
36   for {
37     n, err := resp.Body.Read(buffer)
38     if err == nil {
39       bar.Add(n)
40     } else if err == io.EOF {
41       return
42     } else {
43       panic(err)
44     }
45   }
46 }
```

number of bytes expected from the web request. Lines 24 to 30 also define cosmetic settings, such as the ASCII character for the saucer, that is, the previously unidentified flying object that illustrates the progress (`#` in this case), the bar frame as `[]`, and the fill character for the empty bar as `-`.

Since the data bytes from the web request arrive in chunks from the web anyway, the progress bar and the logic to refresh it can be organically integrated into the code. On the other hand, if long running system calls determine the program's run time, they need to be rewritten so that the bar can progress step by step, instead of pausing until shortly before the end, before jumping to the finish at warp speed.

Retro Look

If you are looking for more eye candy than just command-line characters, you can impress your users with a terminal UI, such as the `termui` project I introduced in a previous article [3]. Figure 4 illustrates how Listing 2 displays its progress while copying a large file.

Since GUIs and thus the progress bar are running in an event loop, data must

Listing 2: cpgui.go

```
01 package main
02
03 import (
04     ui "github.com/gizak/termui"
05     "io/ioutil"
06     "os"
07     "fmt"
08     "log"
09 )
10
11 func main() {
12   file := os.Args[1];
13   err := ui.Init()
14   if err != nil {
15     panic(err)
16   }
17   defer ui.Close()
18
19   g := ui.NewGauge()
20   g.Percent = 0
21   g.Width = 50
22   g.Height = 7
23   g.BorderLabel = "Copying"
24   g.BarColor = ui.ColorRed
25   g.BorderFg = ui.ColorWhite
26   g.BorderLabelFg = ui.ColorCyan
27   ui.Render(g)
28
29   update := make(chan int)
30   done := make(chan bool)
31
32     // wait for completion
33   go func() {
34     <-done
35     ui.StopLoop()
36   }()
37
38    // process updates
39   go func() {
40     for {
41       g.Percent = <-update
42       ui.Render(g)
43     }
44   }()
45
46   go backup(file, fmt.Sprintf("%s.bak", file),
47             update, done)
48
49   ui.Handle("/sys/kbd/q", func(ui.Event) {
50     ui.StopLoop()
51   })
52
53   ui.Loop()
54 }
55
56 func backup(src string, dst string,
57  update chan int, done chan bool) error {
58
59  input, err := ioutil.ReadFile(src)
60  if err != nil {
61    log.Println(err)
62    done <- true
63  }
64  total := len(input)
65  total_written := 0
66
67  out, err := os.Create(dst)
68  if err != nil {
69    log.Println(err)
70    done <- true
71  }
72
73  lim := 4096
74  var chunk []byte
75
76  for len(input) >= lim {
77    chunk, input = input[:lim], input[lim:]
78    out.Write(chunk)
79    total_written += len(chunk)
80    update<- total_written*100/total
81  }
82  out.Write(input)
83
84  done <- true
85  return nil
86 }
```

be read and written in a non-blocking, asynchronous fashion, such as with Node.js. When data arrives in this programming paradigm, the code regularly triggers callbacks when more data becomes available. In addition to gobbling up and processing data, these callbacks are handy to update our progress bar, to reflect the amount of data read or written so far.

Channels and Routines

Go offers so-called goroutines and channels (as discussed in a previous issue [4]) for firing off concurrent program parts and their synchronization. Listing 2 shows how to illustrate the long copying process for a large file in Go with a progress bar from the `termui` package. After installing the UI package like this

```
$ go get github.com/gizak/termui
```

and building the program like this

```
$ go build cpgui.go
```

the call to `cpgui foo` copies a file named `foo` to `foo.bak`. If it is relatively large, you can follow along by watching the terminal UI's progress bar drawn on the terminal, gradually growing to the right until it reaches the end of the display element, at which point the file has been copied and the program terminates (Figure 4).

Line 19 of Listing 2 defines a new structure using `NewGauge()` from the `termui` package; this represents the UI element of a progress bar. The initial value of the bar growing from left to right is set to `0` with the `Percent` attribute; in other words, the bar is at zero percent and thus completely on the left and invisible. Lines 21 to 26 define further attributes, such as the colors of the bar's individual components, their size, or the caption for the graphic element.

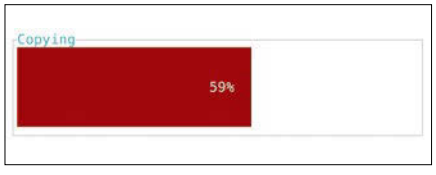


Figure 4: Listing 2 updates a progress bar while copying a large file in Go with the `termui` library.

Communication Channels

Listing 2 uses the channels defined in lines 29 and 30, named `update` and `done`, for the communication between the different program parts that write the data or refresh the bar.

The `backup()` function as of line 56 receives the `update` channel as a parameter from the main program and uses it to update the progress bar by sending the completed percentage into it, as soon as it has written a chunk of bytes to the file. On the receiving end of the channel, the `main()` function waits for new messages in a concurrent goroutine running an infinite loop starting in line 40. The read function on the channel in line 41 blocks if no new values have been written to the channel yet by the `backup()` function. When a new integer value arrives in the channel, line 41 updates the progress bar's `Percent` attribute in `g` to the new percentage value and then redraws the graphic element with `ui.Render(g)`.

The second channel, `done`, allows the `backup()` function, which also receives this channel as a parameter from the main program, to initiate the end of the program. For this to happen, the main function asynchronously waits for data in the `done` channel in a goroutine starting in line 33. As soon as data becomes available (because lines 60, 70, or 84 have sent a `true` into the channel), line 35 triggers `ui.StopLoop()` in order to close the UI event loop, which tears down the GUI in line 53 and terminates the `main()` function.

In this simplified example, the `ReadFile()` function reads the source file data via the `ioutil` package in one fell swoop. If the package isn't installed yet, you do so by typing `go get io/ioutil`. The data ends up in an array slice holding elements of the `Byte` type. The `len()` function in line 64 determines the length of the file in bytes and stores it in the `total` variable. Line 67 creates the new file with the `.bak` extension on the filesystem and returns a writer interface in `out`. The interface is then passed onto the `Write()` function, which sends 4096-byte chunks in line 78 until all the bytes of the original file have been successfully copied. Reading all data into memory at once is obviously not a good idea for large files; in a real-world application, you'd want to read the data in chunks and write it as it becomes available, updating the progress bar accordingly.

In the write loop, the next 4096-byte long chunk from the buffer `input` is retrieved by the

```
chunk, input = input[:lim], input[lim:]
```

statement in line 77 and dumped into the `chunk` array slice, while simultaneously erasing the data from the original `input` buffer. Since Go's array slices are just lightweight constructs referencing underlying arrays (which won't change in this case), this is actually very efficient.

The loop starting in line 76 repeats until only a remainder with less than 4096 bytes is left in the `input` array slice; line 82 writes the rest to the new file outside the For loop, thus completing the copy process.

To allow the user to interrupt the copy process manually if necessary, line 49 defines a keyboard handler for the *Q* key, which uses `StopLoop()` to close the GUI and terminate the program cleanly.

At the end of the copy process, line 84 sends a true value to the `done` channel, which causes the main program in line 34 immediately to unblock and proceeds to terminate the program via `ui.StopLoop()`.

One thing to be aware of is that entertaining the user during the copy process is expensive: Writing data in 4096-byte blocks slows things down considerably in the case of larger files. And, as noted previously, Listing 2 reads the file contents to be copied into memory in a single action, which may not be a good idea for gigabyte-sized masses of data. But it's definitely good enough for illustrative purposes, such as for Hollywood's film industry. ■■■

Info

[1] Progress bar: *https://en.wikipedia.org/wiki/Progress_bar*

[2] Listings for this article: *ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/220/*

[3] "Programming Snapshot – Classics Repackaged: termui" by Mike Schilli, *Linux Magazine*, issue 218, January 2019, p. 42

[4] "Programming Snapshot – Simulatenous Runners: goroutines" by Mike Schilli, *Linux Magazine*, issue 219, February 2019, p. 40


LINUX MAGAZINE
INTERNET OF THINGS
Tools for home automation
Bluetooth Tricks
Linux on Old Hardware
Tablets on Linux
Raspberry Pi 3 B+
Python Gambling
Logism
PIRATEBOX
MYCROFT
MYCROFT
Can an open source personal assistant compete with Alexa?
Scanners and Linux
Peppermint OS
Meltdown & Spectre
Knight's Tour
Node-RED
LINUXVOICE
FOSSPicks
BROWSER SHOOTOUT
IS FIREFOX FINALLY BACK?
IS FIREFOX BACK?
We compare Mozilla's reborn browser with recent versions of Chrome
Kernel Self-Protection
Font Tricks
Write a script that changes your YouTube metadata
Lakka
NoMachine
Tkinter Library
4 COLLABORATIVE OFFICE SUITES: Keep your team in sync
LIBREBOOT and the liberation of firmware
FREE BOOT!
MAPPING TOOLS
Create and edit digital maps with QGIS and QMapShack
Kubuntu Up Close
MakerSpace
Caddy
ODROID-C2
Ogg Vorbis
A Python Script
FEDORA 27
We test drive the newest Fedora
TERMINAL TUNING
GIT WORKSHOP
.NET CORE
QUBES OS
INNOVATIVE DISTROS
Timers
WILL SYSTEMD REPLACE CRON?
INNOVATIVE DISTROS
Systemd
Getting more from the elusive new Linux init system
10 Top PDF Readers
Arduino Programming for Open Hardware Projects
Homegrown Facial Recognition
Maker Tricks
Cover Your Ass(ets)
DVD INSIDE! ALL 214 ISSUES ON A SEARCHABLE DISC!
PRIVACY
PRIVACY
Stop snoopers and protect your identity
Rockbox
RSS Readers
PiXtend
Rasp Pi Sailboat
Ready to Go?
Secrets of data compression
maddog: Life of Linux
Pass: Protect your accounts with this free password manager
Tutorials
Docker 101 for End Users
WWW.LINUX-MAGAZINE.COM

MakerSpace

Running x86 Programs on the Rasp Pi

Out of This World

Some of your favorite programs might not run on Raspberry Pi because no version is available for the ARM architecture. ExaGear Desktop changes that by acting as a translator between the ARM and x86 worlds. *By Bernhard Bablok*

The open source advantage was evident from the beginning with the Raspberry Pi: Since free software usually runs on many different platforms, almost all Debian packages could be ported to the new system in a very short time. In addition, there was software that exploited the Rasp Pi's special hardware features.

But for proprietary programs, including most Windows applications, the path to the Raspberry Pi is much less assured. Unless the developers want to port their code to the Rasp Pi, users are out in the cold – unless they know about ExaGear.

ExaGear is a tool that provides a special translation layer that converts x86 commands to ARM equivalents. You can use ExaGear to run software written for x86 systems on the ARM-based Raspberry Pi. Support for the Wine integration layer means you can even run old Windows programs directly on the Rasp Pi.

Installation

For a proprietary program, the download and installation is extremely easy [1]: ExaGear can be found in the Raspbian package sources. You can thus

Listing 1: Installing ExaGear Desktop

```
$ sudo apt update
$ sudo apt install exagear-desktop
$ exagear
Starting /bin/bash in the guest image /opt/exagear/images/debian-8
```

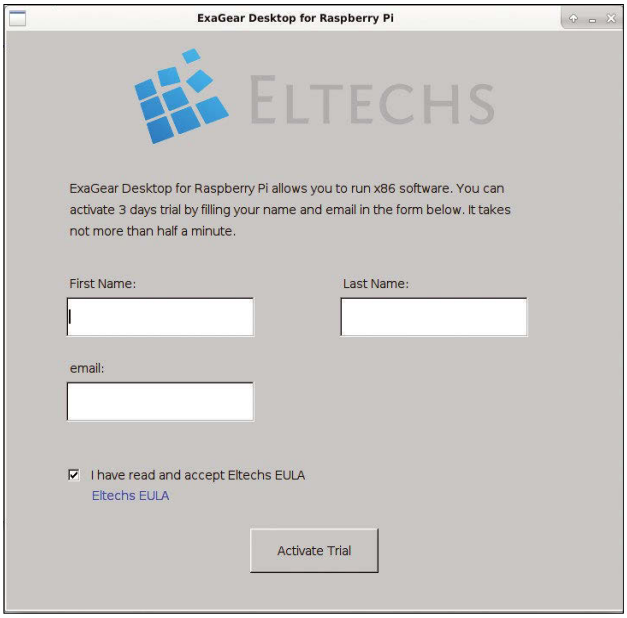


Figure 1: As a trial version, ExaGear can be used in full for three days without a paid license.

Lead Image © innovari, fotolia.com

install it conveniently via the package manager and then start the emulator (Listing 1).

The last command in Listing 1 switches to the x86 environment. The first time you call it, a form appears, prompting you for your name and email address (Figure 1). The trial version can then be used in full for 72 hours. The license is tied to the CPU's serial number, which prevents users from simply reinstalling.

With a full license, the procedure is somewhat different (see the "Versions" box). Download a tarball specific to the Rasp Pi model used and unpack it in the same directory as the license file. Then start the installation command (Listing 2). (Eltechs kindly provided us with a license for this article.)

The tarball essentially contains a number of deb packages with guest systems, from which the installation script then selects the most suitable alternative for the installation. It also checks the license and activates the program.

First Steps

After starting the ExaGear environment with the command `exagear`, nothing happens except a welcome message. The output of the commands `uname -a` and `cat /etc/os-release` (Figure 2) show that the system now runs – with identical kernel versions – under an x86 architecture instead of an armv7l architecture. With the command `exit`, you leave the x86 environment again.

Later on, you can avoid the detour via the `exagear` command for installed x86 programs, because the kernel identifies the binary format and launches the wrapper automatically. Of course, this is not automatically the case, but needs to be configured using the ExaGear installer.

```
[bablokb@pi3:~] > uname -a
Linux pi3 4.14.34-v7+ #1110 SMP Mon Apr 16 15:18:51 BST 2018 armv7l GNU/Linux
[bablokb@pi3:~] > arch
armv7l
[bablokb@pi3:~] > cat /etc/os-release
PRETTY_NAME="Raspbian GNU/Linux 9 (stretch)"
NAME="Raspbian GNU/Linux"
VERSION_ID="9"
VERSION="9 (stretch)"
ID=raspbian
ID_LIKE=debian
HOME_URL="http://www.raspbian.org/"
SUPPORT_URL="http://www.raspbian.org/RaspbianForums"
BUG_REPORT_URL="http://www.raspbian.org/RaspbianBugs"
[bablokb@pi3:~] > exagear
Starting /bin/bash  in the guest image /opt/exagear/images/debian-8
[bablokb@pi3:~] > uname -a
Linux pi3 4.14.34-v7+ #1110 SMP Mon Apr 16 15:18:51 BST 2018 i686 GNU/Linux
[bablokb@pi3:~] > arch
i686
[bablokb@pi3:~] > cat /etc/os-release
PRETTY_NAME="Debian GNU/Linux 8 (jessie)"
NAME="Debian GNU/Linux"
VERSION_ID="8"
VERSION="8 (jessie)"
ID=debian
HOME_URL="http://www.debian.org/"
SUPPORT_URL="http://www.debian.org/support/"
BUG_REPORT_URL="https://bugs.debian.org/"
[bablokb@pi3:~] >
```

Figure 2: The system will identify itself as an x86 environment after you start ExaGear.

Listing 2: Installation Command

```
exagear-desktop-v-2-2/exagear-desktop-rpi3.tar.gz
$ tar -xvzpf exagear-desktop-v-2-2/exagear-desktop-rpi3.tar.gz
$ sudo ./install-exagear.sh
```

Listing 3: Remounting the Original Root Partition

```
$ sudo mkdir /opt/exagear/images/debian-8/host
$ sudo mount /dev/mmcblk0p2 /opt/exagear/images/debian-8/wirt
```

Installing ExaGear puts a minimal Debian "Jessie" distribution on your system. Unlike real emulators or virtual machines, the x86 environment makes use of the host system environment – ExaGear is thus explicitly unsuitable for clean program isolation.

The Jessie filesystem is located below `/opt/exagear/images/debian-8/`. From the normal Raspbian system, you can access the data stored there without any problems. Conversely, you can access the home directories of the host system: ExaGear integrates both `/home/` and `/dev/`.

If you need access to the complete host system, simply remount the original root partition completely (Listing 3). After that, the guest system sees the complete host filesystem below `/host` – whether or not this makes sense is up to you to decide.

The `dpkg -l` command shows the few installed packages of the frugal ExaGear Debian. However, the retroactive installation of further software works without

Versions

ExaGear Desktop costs between EUR11.95 for the Pi Zero/RPi1-compatible basic version and EUR40.95 for the RPi3-compatible enterprise version, depending on the platform and scope of functions. However, there are always special offers; for example, at the beginning of the 2018 FIFA World Cup, you could get two licenses for the price of one.

In addition to the focus on a specific Rasp Pi model, the versions also differ in terms of flexibility and support. The basic version is tied to a specific device – if your Rasp Pi dies, the license is also lost. The Pro version, on the other hand, allows you to transfer the license to another device of the same class. The basic version costs EUR23.95 Euro, and the pro version costs and additional 12 euros. The enterprise version supports use in the enterprise, offers better support, and simplifies the installation via volume licenses.

The cheapest license (for the Rasp Pi Zero/RPi1) is not really all that useful even for general use. Many programs fail due to meager resources. For special programs, however, the combination of Pi Zero and ExaGear can be a sensible alternative to a dusty Windows XP PC that only runs because there is no modern replacement for the beloved old laboratory software.

problems and – in the usual Raspbian style – relies on the command:

```
sudo apt-get install <package>
```

The guest system also relies on the host for services and udev rules. It would thus be pointless to reinstall services like cron inside the emulator. It only makes sense that the software prevents this.

Architecture

ExaGear Desktop is not a classic emulator that provides a complete operating system environment – in particular, it lacks its own kernel. In principle, the software works in a similar way to Wine, which provides a Windows environment under Linux. The run-time environment intercepts x86 commands and translates them into corresponding ARM commands. For graphics output, ExaGear uses the host's X server; the Linux desktop's architecture helps the emulator here (see the "X Architecture on Linux" box).

The advantage of the lean emulation layer is that it automatically supports multithreading; however, you still experience a loss of performance. Eltechs promises up to 80 percent of native performance, with the loss depending on the specific commands an application uses. In extreme cases, up to two-thirds of the Rasp Pi performance is lost.

On the Raspbian side, you should definitely take some precautions to ensure optimal conditions. This includes at least a newer generation RPi2 (the same processor as the RPi3) or higher. A fast hard disk (SSD) for the system also helps. You should also give the GPU enough memory, at least 256MB, which you can set with the parameter `gpu_mem` in the `/boot/config.txt` file.

Graphic-intensive applications also benefit from enabling hardware-supported graphics acceleration. You can do this in `raspi-config`, where you will find the setting *GL Driver* below *Advanced Options*. Select *GL (Full KMS) OpenGL desktop driver with full KMS*. Applications such as Skype (32-bit Linux version) have a problem with this. When Skype accesses the video camera, whether it's a Pi or USB camera, the X server crashes.

On the network side, emulation is less of a performance brake. Cloning a 260MB GitHub project took 140 seconds natively and 231 seconds in the emulator. The download rate in the emulator was only seven percent lower, the rest of the loss was attributable to the CPU for encryption and processing the Git metadata. With NFS or Samba transfers, on the other hand, the loss is in the range of measurement uncertainty.

X Architecture on Linux

On Linux, the operating system traditionally does not draw the graphical output, but instead relies on a separate program known as the X server. Applications that want to output a graphical user interface communicate with the X server and tell it what to draw. The X server then calls the kernel's low-level system driver interfaces. Communication between the X clients and the X server takes place via the network interfaces.

On workstations, the X server and its clients usually run on the same system. The X clients could also run on a computer at the other side of the world and send their graphics commands via the Internet to the home computer, which then displays the interface. Thanks to the standardized protocol, this also works across operating system boundaries. A Rasp Pi program's graphical output could therefore be easily displayed on an X server running Windows.

In relation to ExaGear Desktop, this means that the emulator does not have to take care of the graphical user interface. The x86 programs are ultimately equivalent to additional X clients. They send the output commands to the host's X server, which then displays the output. The server returns keyboard and mouse events to the X client in the emulator.

Listing 4: Installing Wine

```
$ sudo apt update
$ sudo full-upgrade
$ sudo apt install wine
$ wine --version
wine-2.0-eltechs
```

First Tests

With ExaGear Desktop, all (32-bit) programs running under Debian 8 can be run on the Rasp Pi. Eltechs has a collection of instructions for popular programs on its website, such as Skype for Linux or Wine as the basis for Windows programs. Often a video also shows the necessary steps. Anyone who has already installed programs under Raspbian will usually be able to do without these instructions.

The first candidate for the emulator was the open source Visual Studio Code (VSC) editor, which was developed by Microsoft and has since found a large community on GitHub. In addition, VSC runs on many platforms and has what it

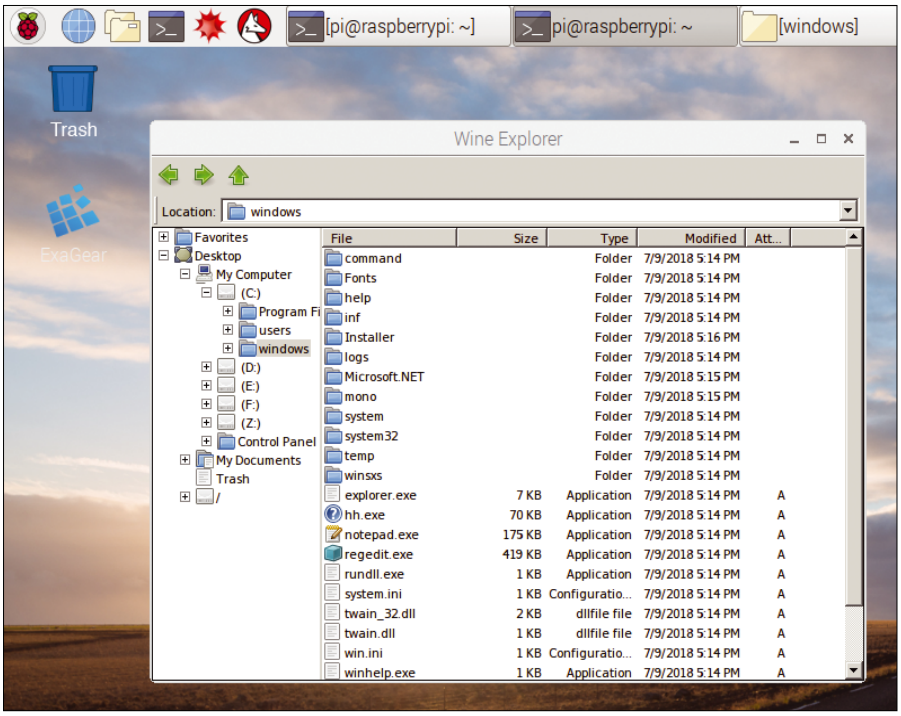


Figure 3: Windows Explorer in action on Raspbian.

takes to send even my own XEmacs editor into retirement.

There are no precompiled binaries for Raspbian for the editor, and building your own binaries is tricky because of the many dependencies. Fortunately, the installation under ExaGear with the standard installation instructions from the VSC page [3] worked with virtually no problems. I had to install two dependencies manually, but this was not due to ExaGear.

However, the result of this first attempt was disappointing. VSC started via ExaGear could not be used under Raspbian. Launching the program consumed so many resources that the system seemed to hang. A little research on the Internet shows, however, that even a native ARM build under Raspbian can be very slow.

The lesson: Don't expect miracles from ExaGear (or Raspbian itself). Even though many programs run smoothly on a RPi3/RPi3B+, there are limits, and an additional emulation layer doesn't make things any better.

Saving Old Treasures

Much more important than modern and accordingly resource-hungry applications are old treasures, which you want to launch on Raspbian every now and then. This above all includes Windows programs from the last and penultimate generation.

For this you install the already mentioned Windows run-time environment Wine (Listing 4) under ExaGear. Eltechs provides a custom version whose package name contains the string *"eltechs"*. The Windows Explorer celebrates its resurrection under Raspbian (Figure 3).

This additional emulation layer costs surprisingly little power. This is because only Windows system calls have to be translated to the analog kernel commands, but not the x86 to the ARM command set. After the Wine installation, supported Windows programs (32-bit) can be installed in this environment. This does not always work without complications, but is well documented for a large number of programs [4] and independent of the Raspbian and ExaGear substructure.

Wine typically supports older Windows programs, as the Windows command set has expanded over time and the Wine developers have to program for it. The advantage: Older programs were developed for weaker PC generations, so that a Rasp Pi's computing power including the emulator is usually sufficient. The freeware mp3DirectCut [5] for lossless editing of MP3 files, for example, runs smoothly and without problems (Figure 4).

Even old games may run smoothly with the combination of Wine plus ExaGear, if you believe the reports on the Internet. Due to my lack of experience in this field, however, this was not tested.

Conclusions

ExaGear's license terms are not very customer-friendly, especially in the basic version with the binding to a single Raspberry Pi board. One should therefore consider the choice of license carefully. However, if you don't want to do without an important program under Raspbian, you can pay the required amount after a performance test. ■■■

Info

[1] Installing the ExaGear Desktop trial version: *https://docs.eltechs.com/install-and-configure-exagear-desktop/how-to-install-exagear-desktop-trial*

[2] ExaGear Desktop versions and pricing: *https://eltechs.com/product/exagear-desktop/exagear-desktop-features-and-prices/*

[3] Installing VSC: *https://code.visualstudio.com/docs/setup/linux*

[4] Installing Windows software under Wine: *https://appdb.winehq.org/*

[5] mp3DirectCut: *http://mpesch3.de1.cc/mp3dc.html*

Author

Bernhard Bablok is an SAP-HR developer at Allianz Technology SE. When he's not listening to music, riding his bike, or walking, he focuses on topics relating to Linux and object orientation.

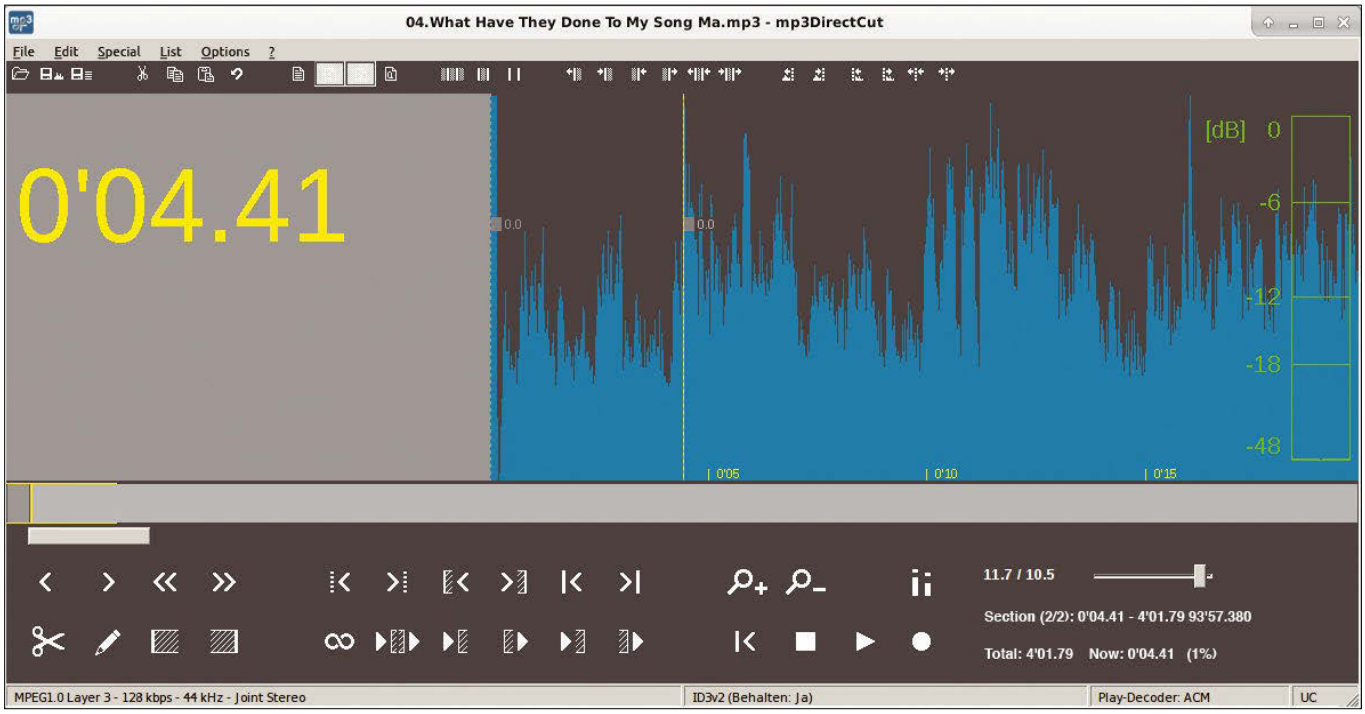
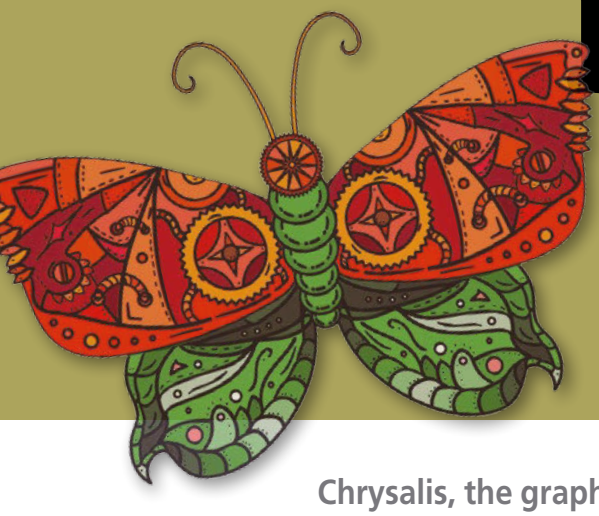


Figure 4: You can run older Windows programs like mp3DirectCut on the Raspberry Pi using Wine plus ExaGear.



MakerSpace

Keyboardio's graphical interface

An Open Hardware Butterfly

Chrysalis, the graphical interface for Keyboardio's Model 01, offers an easy way to customize open hardware keyboards. *By Bruce Byfield*

By definition, open hardware requires open source software. However, the software is often overlooked, because the hardware is more tangible. Yet the software being written for open hardware often leads to new results and industry standards – everything from utilities for securing mobile devices to applications for prosthetics. A case in point is Chrysalis [1], the graphical interface for the Model 01 keyboard from Keyboardio [2]. Recently, I talked to Gergely Nagy (aka Algernon), the contractor who is the lead developer for Chrysalis, about the history of the project and where its development is heading.

Keyboardio is a small company that began shipping its stylish, ergonomic, and programmable keyboards in November 2017. It has shipped thousands of Model 01 keyboards, each powered by an Arduino ATmega3244 microcontroller. The advantage of using an Arduino is that the Arduino IDE includes a feature for flashing the firmware, which allows for both programmable keys and layers or multiple keyboard layouts that can by swapped in and out in one or two keystrokes. In fact, the Model 01 supports up to 32 layers. Flashing the firmware makes all this customization possible.

The Arduino IDE is easy to use if you have programming experience. However, for ordinary users, as Nagy points out, "Flashing can be very scary, for a multitude of reasons, and you need special tools for it, often with arcane incantations to make them do what you need them to do. And programming keyboards? Eeeh, not friendly at all. For the average person, who just wants to rearrange their layout, having to install Arduino and edit source code is well past their comfort zone. Some are willing to go there and end up enjoying it, [but] many are not" (Figure 1).

Chrysalis is named for Keyboardio's butterfly logo, which is derived from the shape of the Model 01's divided keyboard. You do not have to be an English major to infer from the name the suggestion that Chrysalis will transform open hardware keyboards in general and allow them to take flight and realize their full potential. What Chrysalis does is allow the editing of the firmware already on the keyboard, rather than uploading a new file from the Arduino IDE.

"With Chrysalis," Nagy says, "you don't need to know what keycodes are, or what you can or cannot put on a keymap. You don't have to deal with syntax errors. You don't have to deal with source code, nor compilers, nor flashing tools. You just launch the application and click your way through. It's much more limited than programming, indeed. But for the things it offers, it's so much friendlier, too." In effect, Chrysalis opens up keyboard customization to users who might not otherwise attempt it.

The Road to Chrysalis

Nagy can pinpoint the date he started using Linux: December 26, 1996. He started using Debian two years later and became a Debian Developer in late 2000. What he calls his "drive-by contributions" include `syslog-ng`, Riemann, GStreamer, and the Hy programing language. His introduction to keyboard firmwares came from contributing to QMK Firmware, which develops firmware for open source keyboards and is used by at least half a dozen projects and companies, including Planck, ErgoDox EZ, and Atreus [3]. For QMK, Nagy developed Tap Dance [4], a feature that allows keys to type different characters depending on the number of times they are struck.

Nagy's involvement with Keyboardio began around the time that the company was running its crowdfunding campaign in 2015. As he details in his blog [5], his current keyboard needed replacing; he pre-ordered a Keyboardio Model 01 and bought an ErgoDox EZ to use while waiting for his Model 01 to ship. "That led me down a deep, deep rabbit hole," Nagy says. "I like to tweak things, and having open source firmware at my finger tips was too much temptation to resist."

Nagy found that the ErgoDox EZ was not to his liking, but none of the other open hardware keyboards could match the ErgoDox EZ's features. "A lot was missing, and the firmware wasn't set up

the way I like," he says. After a short-lived effort to write his own firmware, Nagy was hired as a contractor for Keyboardio in late 2017, where he continues to work on Keyboardio's Kaleidoscope firmware. He began working on Chrysalis "because I felt there's a need for it, and no one else was working on anything similar. Most GUIs for open source keyboards were inadequate, as they allowed one to save a HEX file one could later flash. I wanted something that can directly talk to the keyboard, to allow people to change layouts, colormaps, you name it, without having to compile anything, [and] without the need to reflash the keyboard."

Nagy did look at Agent, the configuration tool for the Ultimate Hacking Keyboard [6]. However, he concluded that "the protocol it uses is way too complicated and rigid for what I had in mind. I wanted the protocol to be easily extensible." In consultation with Jesse Vincent, Keyboardio's cofounder and CTO, Nagy developed "a very simple protocol between keyboard and host over USB. A human readable one, too. One that is easy to implement on the firmware side, and easy to work with on the host side, too. One that is also incredibly easy to extend, because it doesn't have many rules and is more about convention than about policy. Chrysalis is built on top of this protocol, but the protocol itself is not tied to it in any way. I can talk to the keyboard from the shell, from Emacs, from Python, or anywhere else, too."

A development team quickly formed. Besides Jesse Vincent, in the early stages, James Cash developed code and was responsible for the original keymap editor. Simon-Claudius Wystrach contributed to the interface, while Matt Venn of Dygma Labs contributed ideas and mock-ups.

In addition, Nagy credits his wife, Csilla Nagy. "Plenty of the polish that's going into Chrysalis recently [is] going in because we discussed the UI, and she gave me ideas, or pinpointed pain points. You see, I'm not a UI developer. I'm not even a JavaScript hacker. I'm much more comfortable in lower-level things. My preferred form of working on my layout is C++ code (I'd say Lisp, but I've yet to build a keyboard with an MCU that can run Lisp). I don't even particularly *like* UI development, either."

```
enum { QWERTY, NUMPAD, FUNCTION }; // layers

/* This comment temporarily turns off astyle's indent enforcement
 *   so we can make the keymaps actually resemble the physical key layout better
 */
// *INDENT-OFF*

const Key keymaps[][ROWS][COLS] PROGMEM = {

  [QWERTY] = KEYMAP_STACKED
  (___,          Key_1, Key_2, Key_3, Key_4, Key_5, Key_LEDEffectNext,
   Key_Backtick, Key_Q, Key_W, Key_E, Key_R, Key_T, Key_Tab,
   Key_PageUp,   Key_A, Key_S, Key_D, Key_F, Key_G,
   Key_PageDown, Key_Z, Key_X, Key_C, Key_V, Key_B, Key_Escape,
   Key_LeftControl, Key_Backspace, Key_LeftGui, Key_LeftShift,
   ShiftToLayer(FUNCTION),

   M(MACRO_ANY),  Key_6, Key_7, Key_8,     Key_9,         Key_0,         LockLayer(NUMPAD),
   Key_Enter,     Key_Y, Key_U, Key_I,     Key_O,         Key_P,         Key_Equals,
                  Key_H, Key_J, Key_K,     Key_L,         Key_Semicolon, Key_Quote,
   Key_RightAlt,  Key_N, Key_M, Key_Comma, Key_Period,    Key_Slash,     Key_Minus,
   Key_RightShift, Key_LeftAlt, Key_Spacebar, Key_RightControl,
   ShiftToLayer(FUNCTION)),


  [NUMPAD] =  KEYMAP_STACKED
  (___, ___, ___, ___, ___, ___, ___,
   ___, ___, ___, ___, ___, ___, ___,
   ___, ___, ___, ___, ___, ___,
   ___, ___, ___, ___, ___, ___, ___,
   ___, ___, ___, ___,
   ___,

   M(MACRO_VERSION_INFO),  ___, Key_Keypad7, Key_Keypad8,   Key_Keypad9,        Key_KeypadSubtract, ___,
   ___,                    ___, Key_Keypad4, Key_Keypad5,   Key_Keypad6,        Key_KeypadAdd,      ___,
                           ___, Key_Keypad1, Key_Keypad2,   Key_Keypad3,        Key_Equals,         Key_Quote,
   ___,                    ___, Key_Keypad0, Key_KeypadDot, Key_KeypadMultiply, Key_KeypadDivide,   Key_Enter,
   ___, ___, ___, ___,
   ___),

  [FUNCTION] =  KEYMAP_STACKED
  (___,      Key_F1,           Key_F2,      Key_F3,     Key_F4,        Key_F5,           XXX,
   Key_Tab,  ___,              Key_mouseUp, ___,        Key_mouseBtnR, Key_mouseWarpEnd, Key_mouseWarpNE,
   Key_Home, Key_mouseL,       Key_mouseDn, Key_mouseR, Key_mouseBtnL, Key_mouseWarpNW,
   Key_End,  Key_PrintScreen,  Key_Insert,  ___,        Key_mouseBtnM, Key_mouseWarpSW,  Key_mouseWarpSE,
   ___, Key_Delete, ___, ___,
   ___,
```

Figure 1: Without Chrysalis, flashing firmware requires editing a text file while referring to a list of key codes.

By the time the Model 01 shipped in late 2017, Chrysalis was starting to become usable. However, at the same time, Nagy realized that, as currently structured, the project was not sustainable.

One problem was that Nagy had become almost the sole contributor – and at the time, he was working a full-time job and had recently become the father of twins. The situation improved when he became a Keyboardio contractor, but he adds that "most of my time was still spent on firmware, as that was deemed more important at the time."

Neither was the situation helped by Nagy's early decision to develop in ClojureScript [7]. Nagy made the decision because he was familiar with the language and enjoyed writing in it. "This," he says, "had the advantage that I was able to come up with a proof of concept pretty fast. But it also had a huge downside: Very few people know ClojureScript, and even fewer have Model 01s. So the pool of potential contributors was small." In addition, Nagy admits that the first version of Chrysalis "also showed signs of me not being a UI person. It wasn't designed well and had very rough edges. It wasn't easy to work with, even for someone familiar with ClojureScript."

Because of these problems, around the time that the Model 01 shipped, Keyboardio made the decision to rewrite Chrysalis in JavaScript. Nagy remains the main contributor, but, as I write, he has recently released version 0.0.6 of Chrysalis [8] for Linux, macOS, and Windows. It features a revamped interface, but its support for other features remains limited.

A Quick Tour of Chrysalis Today

According to Nagy, Chrysalis "talks a very simple, text-based protocol over USB serial. The most interesting parts are the firmware, really. With the factory firmware that the Model 01 ships with, the keymap is stored in read-only memory, [and] we're obviously unable to overwrite that without flashing. We can't use RAM to store the keymap either, because that doesn't persist through keyboard reboots – and RAM is also the most limited resource on the board. So we store a few layers in EEPROM 9 (Electrically Erasable Programmable Read-Only Memory) [9]. Chrysalis asks the keyboard for the key-

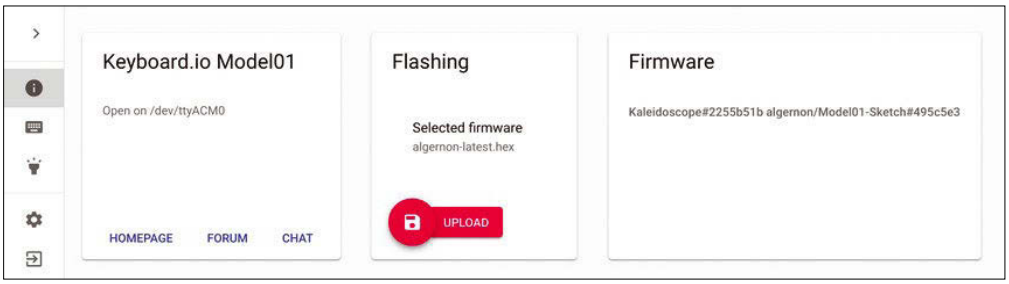


Figure 2: Using Chrysalis begins with selecting the input device.

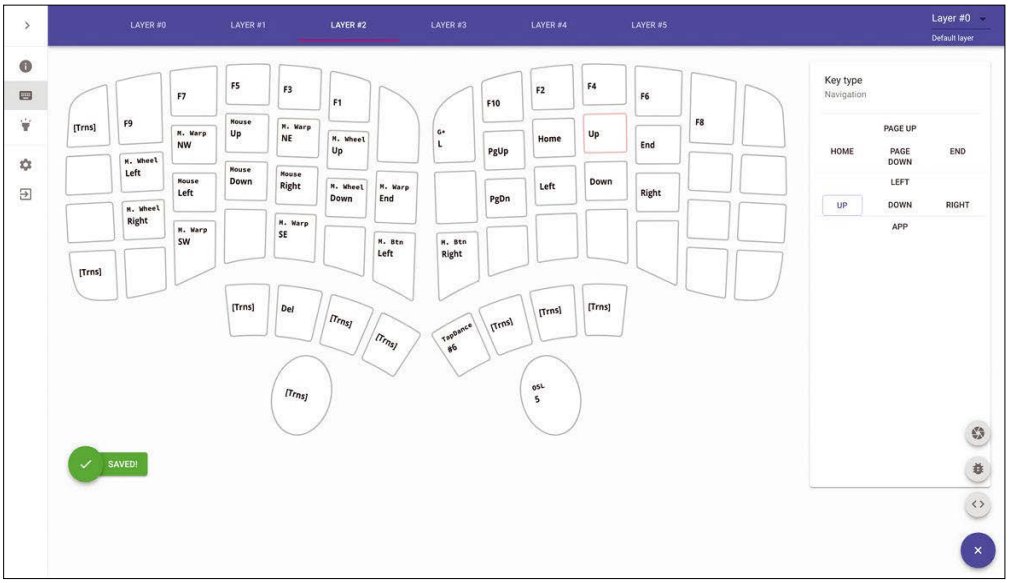


Figure 3: Individual keybindings and entire layouts are supported.

map, how many read-only layers it has, and which one is the default one. So it can present all the layers, and a few extra with which one's free to do as desired. [Y]ou do have to pre-allocate space in EEPROM for the extra layers."

Currently, Chrysalis allows users to select a keyboard (Figure 2) and then make minor adjustments to a keymap or create a new layer (Figure 3). Macros are not supported. If you have the Colormap plugin installed [10] – which must be enabled by flashing – you can also set the LED backlights behind each key (Figure 4). It also can flash new firmware, allowing users to customize without using the Arduino IDE, although the firmware still has to be edited elsewhere.

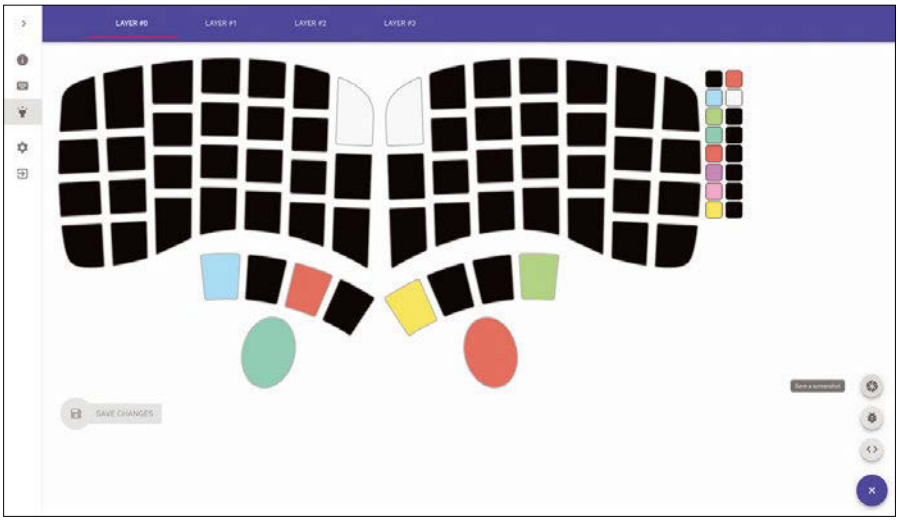


Figure 4: The backlights of keys can be customized if the keyboard's firmware includes the Colormap plugin.

The results of editing in Chrysalis are not imported to a new firmware file, but rather saved directly to the keyboard.

At this stage, Chrysalis is at a late alpha or early beta state. However, it has already started to make its influence known. Dygma [11], which participated in early development, plans to use Kaleidoscope for the firmware in its keyboards and continues to follow the development of Chrysalis as well. In addition, two years ago, I was told that Kaleidescope and Chrysalis are the basis of the interface for Input Club's customizable keyboards [12]. Despite the rewrite, Chrysalis appears well on its way to answering the need for an easy way to customize open hardware keyboards. With contributions from one or two programmers, its influence would be felt faster, but for now, at least progress continues, even it if is slower than customers would probably prefer. ■■■

Info

[1] Chrysalis: *https://github.com/keyboardio/chrysalis-bundle-keyboardio/releases/tag/chrysalis-bundle-keyboardio-0.0.6*

[2] Keyboardio: *https://shop.keyboard.io/*

[3] QMK: *https://qmk.fm/*

[4] Tap Dance: *https://beta.docs.qmk.fm/features/feature_tap_dance*

[5] Nagy's blog: *https://asylum.madhouse-project.org/blog/2015/11/20/looking-for-a-keyboard/*

[6] The Ultimate Hacking Keyboard: *https://ultimatehackingkeyboard.com/*

[7] ClojureScript: *https://clojurescript.org/*

[8] Chrysalis repository: *https://github.com/keyboardio/chrysalis-bundle-keyboardio/releases/tag/chrysalis-bundle-keyboardio-0.0.6*

[9] EEPROM: *https://en.wikipedia.org/wiki/EEPROM*

[10] Colormap plugin: *https://github.com/keyboardio/Kaleidoscope-Colormap*

[11] Dygma: *https://www.dygma.com/*

[12] Input Club: *https://input.club/*

MakerSpace

Waveform 9 turns the Rasp Pi into a recording studio

Sound Machine

The Rasp Pi has enough performance to serve as a small digital audio workstation. Waveform 9 provides the necessary software. *By Hartmut Noack*

Author

Hartmut Noack works in Celle and Hanover, Germany, as a lecturer, author, and musician. For him, free software and homemade music have always been a great match. At *http://lapoc.de*, you can find some CC-licensed results of his work with free music software.

Tracktion.com has been selling suites for music production – Digital Audio Workstations (DAWs) – for many years. Waveform 8 was the first Tracktion DAW that supported the ARM processor used with the Raspberry Pi. Since Spring 2018, the new Waveform 9 for the Rasp Pi (Figure 1) has proven that the experiment is a success.

Collecting Berries

To download and activate Waveform 9, visit the Marketplace section of Tracktion's

Lead Image © Alexander Gorlov, 123RF.com

Figure 1: Waveform 9 is a DAW that runs on the Rasp Pi.

website [1], for which you need an account. You'll need to purchase a license for $109 – around EUR93.50.

The virtual instruments and effects are only available for Linux on the x86 platform. However, the Rasp Pi variant contains all internal instruments and effects. You can install the Debian package for Waveform with `dpkg -i` on Ubuntu MATE for the Rasp Pi (see box entitled "Setup").

Download purchased packages as needed, choosing the desired platform. The license is valid for all platforms. In other words, if other people in the band absolutely want to stay with Windows, or if you want to edit sessions recorded on the Rasp Pi at home on your Linux PC in the rehearsal room, you simply take the projects with you on a USB stick and continue the work elsewhere.

The website is elaborate and spectacularly designed, but somewhat confusing: It does not reveal at first glance where and how you can access the sample packages.

Some samples are already available for download in the installed and registered Waveform 9 on first launch. However, the download list is not well programmed: After you click on *Download*, the software searches in the background for an application registered for ZIP files using the XDG tool (which is not maintained on many systems). If your system does not have an application registered for ZIP files, nothing happens. You will not even see an error message; you can only see that something is wrong by popping up a terminal.

In KDE, Plasma Ark is registered for ZIP and opens with a click on the download for the sample packages. Afterwards, however, nothing seems to happen. The reason for this is that the 2GB package – not unusual for such collections – has to be stored in a temporary directory first. While this is happening, Ark waits without any activity and only displays the content after about 10 minutes. From there, you can drag it into a directory below `~/Tracktion/`.

The downloads and installations of some other additional packages are a little less of an adventure; these packages include the drum loop collections, which you download individually in your web browser.

However, really serious problems did not occur during the many installation steps. Depending on your Internet connection, you will probably have successfully completed the installation and registration for the entire Waveform package and several gigabytes of templates within about one to two hours.

Since forcibly opening the downloads with a ZIP program adds memory, and KDE Plasma or Gnome are not necessarily the ideal desktop environments for the Rasp Pi, it is a good idea to download the packages on a Linux PC.

Unlike Bitwig's DEB packages or Ardour's installer, Tracktion's DEB packages don't drop the software into `/opt/`, but into `/usr/`. Waveform itself goes by the name of `Waveform9` in `/usr/bin/`, which makes it possible to keep `Waveform8` if necessary.

Setup

The CPU, graphics chip, and RAM of the Rasp Pi support operation of the Waveform interface without delays. The real-time sound engines from JUCE and JACK, however, have requirements that are not met by the BCM sound chip built into the Rasp Pi. Even with very conservative settings for latencies of more than 30 milliseconds, you cannot expect normal operation.

The crude BCM codec is not genuinely suitable for recordings anyway, because input produces a huge amount of background noise on the analog side. The solution lies in a USB sound interface, which is available in a form suitable for music applications starting at around EUR30.

In our lab, we used a UCA222 audio interface by the proven but inexpensive vendor Behringer. The computer detects the device as a USB codec and configures it accordingly. Alsamixer controls the stereo output as a PCM channel, but there's a physical volume control on the unit itself and reasonably acceptable output for the headphones, plus RCA connectors for stereo input and output.

We started the device with QjackCtl and set it for 10 milliseconds latency. This setting means that, even if musicians accompany previously recorded material in sync, recordings can still be made properly. If you use the Rasp Pi as a musical instrument, low latency is even more important. Our configuration allowed us to use the Waveform built-in synthesizers and effects live.

Wave Editing

Waveform 9 is a sequencer, rather than a recorder. Even more so than with Tracktion, creating and arranging loops is the focus. It is possible to use Waveform to record tracks in the traditional way, but effective recording requires you to do some editing and polishing with effects.

Special tools for the mouse, such as those offered for editing by Ardour [2] or Audacity, are not available in Waveform 9. You can instead use the small arrows in the upper frame of tracks while holding down the mouse button. Press *S* to split the track at the cursor position.

After you have split a recording, it is usually necessary to remove sounds of breathing, quips, and the like from the beginning and end of the recording. Simply select the track, move the play cursor to the desired position, and press *S*. Select and then delete the parts that have been split off in this way. Caution: After editing, both parts are automatically selected and both react to actions such as remove and shorten/extend.

For precise editing, it is not absolutely necessary to switch off the snap to beat function. Waveform supports adaptive snapping, which means that the grid for the snapping function is finer at higher magnification. If an edited section doesn't snap into place, use the mouse wheel on the bottom scrollbar to zoom in, and the snap-in edges of the section will almost always snap into place.

However, the newer version warns against opening newly edited projects again with the older one.

Beautiful, but Moody

Like its predecessors, Waveform 9 integrates very well with Linux: It automatically sets up the audio connections to JACK and the MIDI devices of the ALSA Sequencer, as did Waveform 8. What's new is a prompt to send diagnostic and usage data to the manufacturer, which only appears at first launch. Under *Settings* | *Maintenance*, you can give or withdraw your consent later on.

Another new feature is the list of downloads for samples and updates. Waveform 9.0.3 on the Rasp Pi tells you

that version 9.1.1 is already available. This does exist – but only for the PC. A download attempt therefore terminates without an explanation.

Waveform 8 still had a small but annoying problem when managing windows: Opened plugins and other subwindows disappeared somewhere outside their own box behind the main window. In the new Waveform the windows stay nailed to the foreground.

At startup, the software automatically detects installed plugins in all directories you have set up in *Settings* | *Plugins*. It finds the venerable LADSPA plugins and native Linux VSTs (LxVSTs). The more modern LV2 format is still not supported. You can use LV2 and Windows VST if needed via LxVST from Felipe Coelho's universal Carla plugin host. However, the additional load caused by the emulation for Windows VST puts a burden on the limited RAM of the Rasp Pi.

In our lab, we discovered two problems that affect Waveform 9 on a Rasp Pi 3 (RPi3). If you create a project with ALSA as the audio driver, the program crashes if you change to JACK and try to open it for editing. In general, however, it is advisable to run audio software on Linux with JACK and stick to it. To fix the problem, reset Waveform by deleting the configuration in `~/.config/Tracktion/Waveform` if necessary. After that, you'll need to reconfirm the license and the software setup.

What was genuinely annoying was that the software had serious problems recording MIDI notes (see the "Wave Editing" box). Drawing notes in tracks with the pen tool does not work at all; recordings from a connected keyboard are ignored by the application for the time being. However, you will find them in the track after restarting the application. See the box entitled "MIDI Despite Everything" for more on what still works with MIDI and Waveform on the Rasp Pi.

More Is Better

Although Rasp Pi users are denied the big standalone instruments from Collective and BioTek, Waveform still offers new sound generators and many smaller, new plugin tools in its internal pool.

Multi Sampler is still quite simply designed, but it already has a number of features that go far beyond what was possible with Tracktion's old standard sampler (Figure 3).

A click on the Multi Sampler built into the audio clip toolbox at the bottom right will load the selected audio clip as a recording and automatically try to identify individual sound events. You can then readjust the elements if necessary and automatically turn them into a sample bank.

A click on *Create MIDI Clip* in the bottom-right corner of the Multi Sampler window creates a MIDI clip that plays the original recording as samples. For monophonic drum loops, this works almost perfectly and gives the arranger a quick way to turn samples into easily manipulable MIDI compositions.

For MIDI compositions, Waveform 9 offers new automatic help, which lets you create a special track with chord patterns, for example, that is automatically followed by the pattern generator for MIDI clips. If the available, chords are not enough, you can easily set up your own chords (Figure 4).

MIDI Despite Everything

What actually works for MIDI with Waveform 9 on the Rasp Pi? First of all, you can edit existing notes in tracks normally, including the new controller functions for individual notes. Transposing, quantizing, and arranging do not cause any problems, but you cannot insert new notes, not even with copy and paste. It is therefore advisable to proceed very carefully when deleting.

What you can do is edit compositions with score-enabled editors such as Rosegarden [3] or MuseScore [4], type in the notes, then export the MIDI files, and import them back into Waveform. Both Rosegarden and MuseScore are available as free software for the Rasp Pi.

Another tool that is lean enough for the Rasp Pi and has proven itself in our lab is Seq24 [5] (Figure 2). Although the software does not have a score editor, it offers far more convenience than the built-in Waveform editor. Seq24 also features an innovative pattern concept that is reminiscent of a greatly simplified version of the popular matrix sequencers from Bitwig and Ableton.

Seq24 can also play an interesting role in another application without a fully functional MIDI editor, and Waveform is the perfect candidate: use of the DAW in live mode. This technique is sometimes known as "controllerism." Controllerists operate software live using MIDI or OSC controllers and a keyboard. Waveform is very much suitable for this kind of music making.

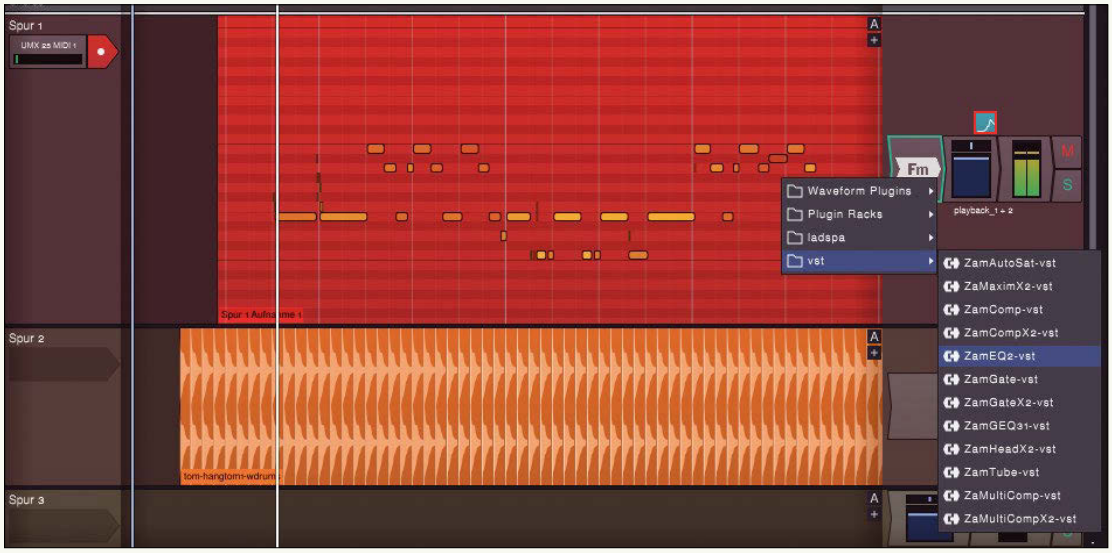


Figure 2: The Seq24 loop-based sequencer has a plain interface that generates hardly any additional load.

Figure 3: By defining zones as rectangles that you move and overlap, Multi Sampler lets you build your own instruments that react very subtly to notes and velocities played with MIDI instruments.

Faceplates

Tracktion has always pursued its own design concept. The application and the interfaces of the underlying JUCE library were designed to be lightweight, functional, and easily scalable to different resolutions. The developers do not bother imitating metal or even wood as a front panel, something you can see with many plugins from other manufacturers. Overly playful elements, like virtual handsets that you can't touch anyway, are instead implemented in the mixer and channel settings as simple bars that you drag with the mouse.

However, for musicians, plugins have the image of virtual hardware, and many love the controls to look like the controls of real Neve or Moog devices. In order to fulfill such wishes, Waveform 9 introduces the concept of faceplates: graphical interfaces with controls as pixel graphics and backgrounds

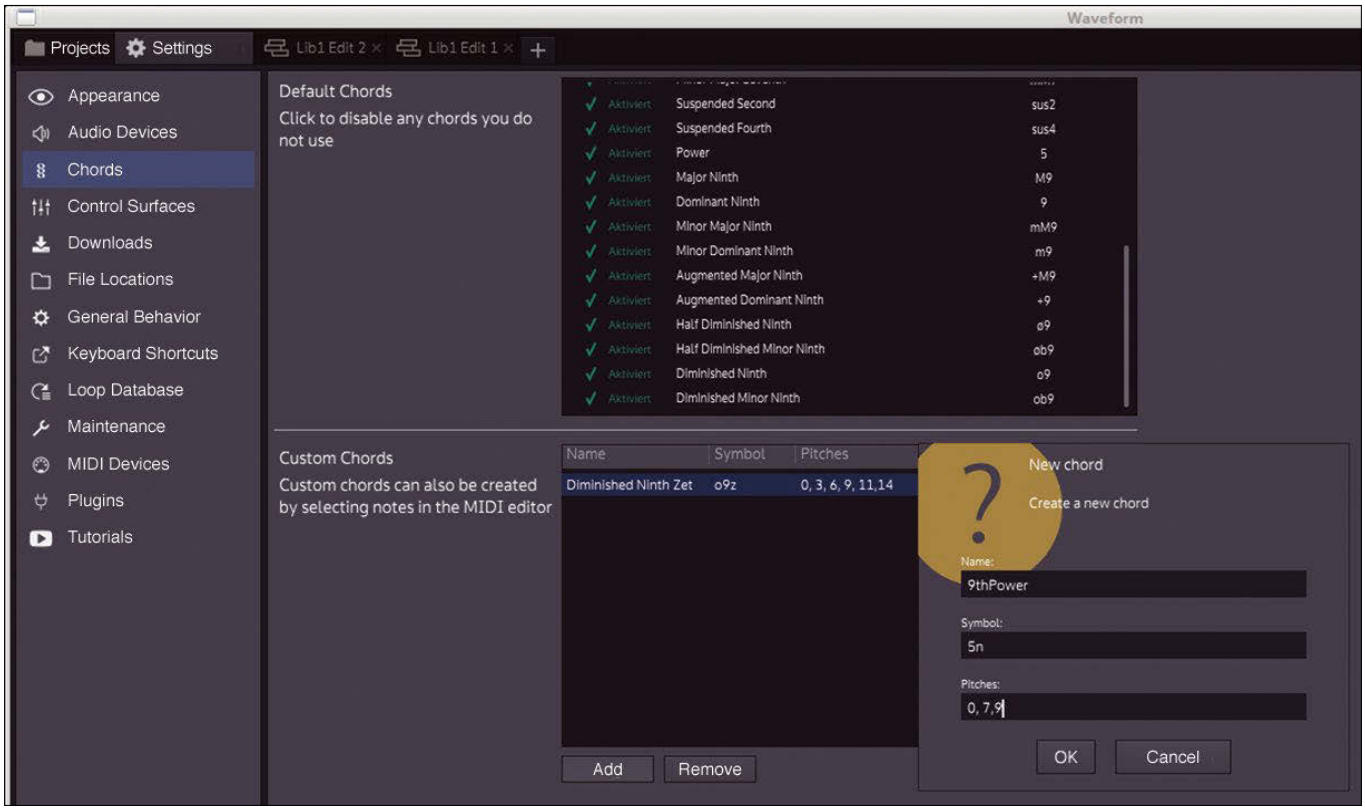


Figure 4: Waveform 9 lets you create your own custom chords.

that look like brushed die-cast aluminum or painted sheet metal (Figure 5).

You can set up these faceplates for plugins and also design and configure them yourself. Faceplates open up the possibility to assign any parameter of a plugin to the buttons and thus create something like a virtual MIDI controller.

Figure 5: A truly Frankenstein-like plugin interface with buttons in the style of Moog, Juno, or Nord.

In order to change the frequency of the second oscillator quickly in the built-in soft synth, you need to select the correct control for the parameter in the GUI of the synth. Put the control on the faceplate, and you can reach it without detours. It is also possible to operate several parameters simultaneously with a faceplate controller and thus implement operations that would simply be impossible in many plug-in interfaces because controllers are located in different views.

Macros

Waveform 9 also introduces macros. Macros let you combine several parameter controllers in one controller. In this way, it is possible to set the influence of the controller on different parameters in different ways. For example, you can couple the opening of a filter cutoff with a decrease in distortion – a technique that might be particularly interesting for friends of controllerism (see the "MIDI Despite Everything" box).

All controllers can now be controlled with wave functions known as modulators. Drag the conspicuous orange symbol in the upper-right corner of the main window onto a plugin, but not directly onto a slider. The modulator functions offer the usual curves (sine, triangle, or similar), envelopes, and a MIDI tracker.

A special new feature is hidden in the interface: MIDI clips now support MIDI Polyphonic Expression (MPE), which means you can apply tone-shaping signals such as vibrato and aftertouch, to individual notes. In the classical MIDI standard, such expression signals are only possible for all currently audible notes at the same time. Of course, this includes a MIDI instrument that sends such signals in line with the MPE standard. Linn and ROLI offer expensive instruments that demonstrate their special qualities in Waveform and Bitwig Studio, even on Linux.

Conclusions

Waveform 9 solves some small problems of the previous version and offers many new, useful extensions. The program still launches and runs as fast as an arrow, even on the meager hardware of the RPi3. With its tidy interface, which is well suited for small screens, it is also suitable for self-built devices based on the miniature PC. If you can live with the problems in the MIDI recording function, Waveform 9 is a very good complete solution for music production on the Rasp Pi at a reasonable price. ■■■

Info

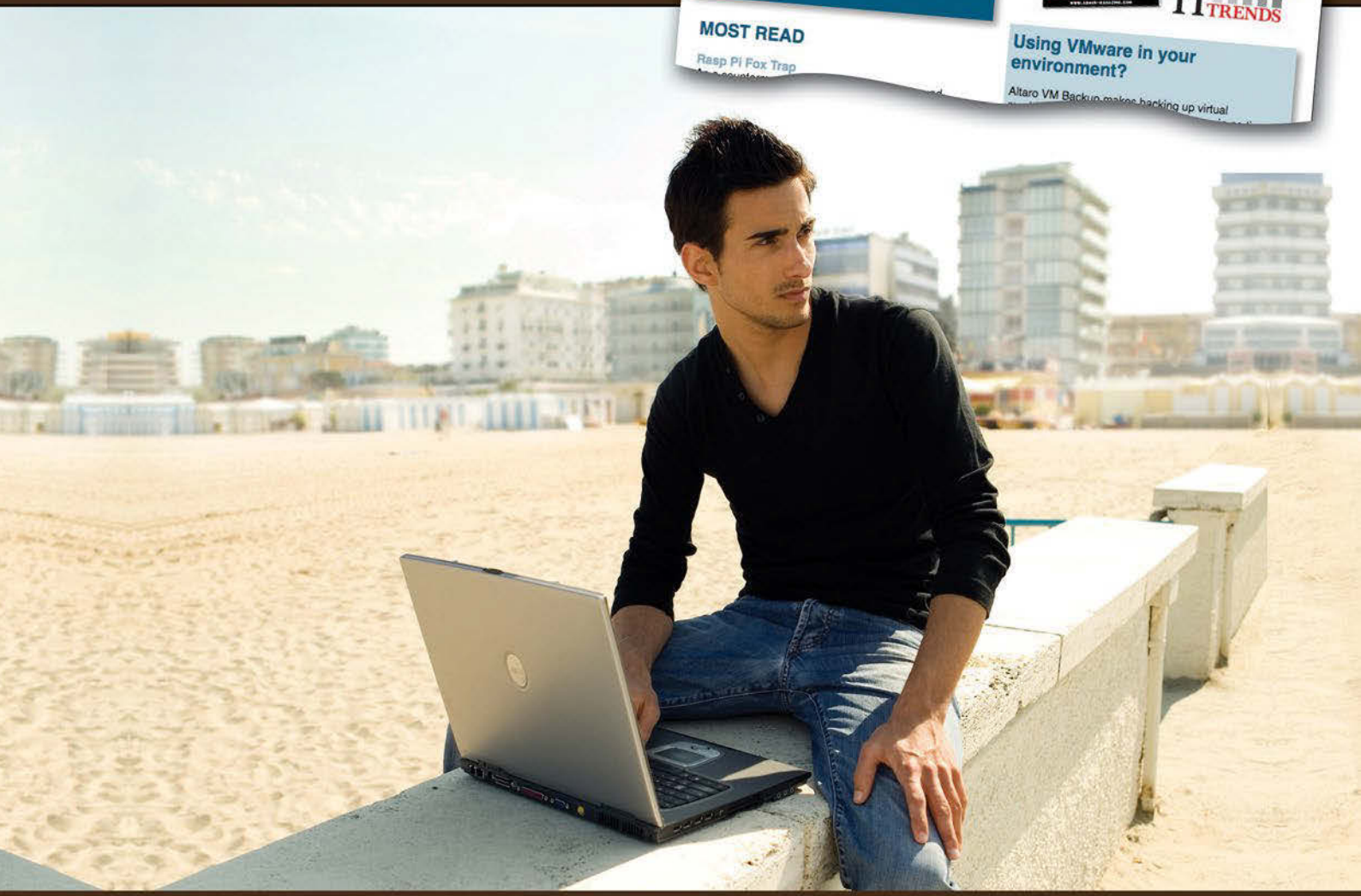
[1] Waveform: *http://tracktion.com*
[2] Ardour: *http://ardour.org*
[3] Rosegarden: *https://www.rosegardenmusic.com*
[4] MuseScore: *https://musescore.org/*
[5] Seq24: *http://www.filter24.org/seq24/*

LINUX UPDATE

Need more Linux?

Our free Linux Update newsletter delivers insightful articles and tech tips to your mailbox twice a month. You'll discover:

- **Original articles on real-world Linux**
- **Linux news**
- **Tips on Bash scripting and other advanced techniques**
- **Discounts and special offers available only to newsletter subscribers**

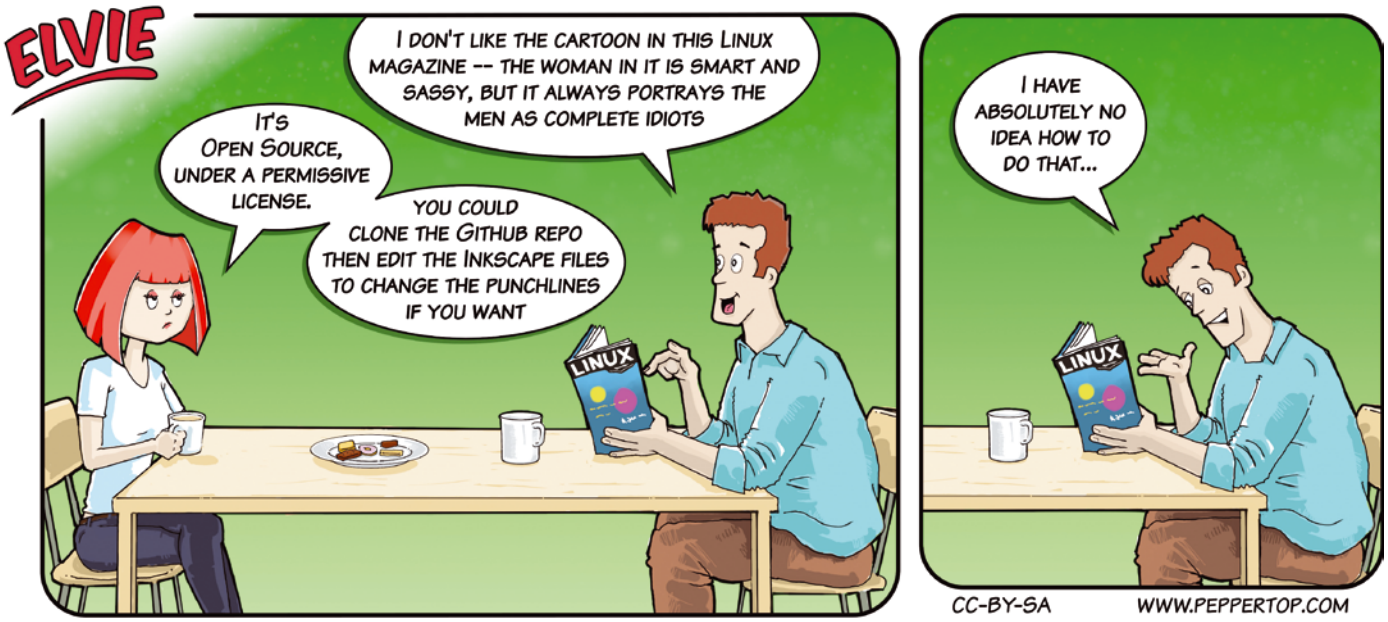


www.linux-magazine.com/newsletter

At LinuxVoice, we specialize in helping readers learn to do new things with open source software. How-to articles are what we live for, so a really original article on doing something really cool – that largely has been undocumented in the past – is especially exciting. This month, we have exactly such an article. Mark Crutch, co-creator of LinuxVoice's own Elvie comic strip (which you see below), offers an inside look at how Mark and Vince create Elvie and other cartoons using free software. Also in this month's LinuxVoice, we introduce you to Go For It!, a sensible to-do manager based on the KISS principle; maddog makes some resolutions for 2019; and Marco describes how to bring the power of Bash hashes and arrays to your shell scripts.

Image © Olexandr Moroz, 123RF.com

LINUXVOICE



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

MADDOG'S DOGHOUSE

With the advent of 2019, Maddog makes a wish list and some resolutions for both himself and the FOSS community.

BY JON "MADDOG" HALL

FOSS goals for 2019

I am writing this article on Christmas Eve of 2018. While (due to the workings of a print magazine) you may not see it for a while, I am going to use this as a combination of a 2019 Christmas wish list and some 2018 New Year's resolutions for myself and (hopefully) some resolutions for the free software community.

My Christmas wish list for next year is simple. I hope to have FOSS even more dominant in the world of computing than it is today.

2019 marks 50 years that I have been in the computing work force, having written my first program in 1969. Although most of that time I have been using what most people call "open source" in one way or another, I have been involved with GNU and Linux for over 25 years.

My reasons for using FOSS over the years have been many and varied. Sometimes it was because I had older computers that could no longer run "mainstream" software, sometimes because the software I needed was not commercially available. Many times it was because I could not afford the commercial software I needed, and I refused to use pirated software.

There were times that I almost regretted not using some of the commercial software available, because it was so compelling, and I had to wait to get software from the FOSS community that could let me do the things I would have liked to do.

These issues have tended to go away for much of the software I need, and FOSS software exists for many of the things I want to do. However non-free software companies keep raising the bar. While new areas of technology often have FOSS solutions, there are some new areas where closed source still seems to be the only solution.

So my first New Year's resolution is to be even more diligent in seeking out truly free software solutions and helping others to create and promote those solutions.

I have started several projects over the past 10 years, such as *CaninosLoucos.org*, a project to design and manufacture open, low-cost, single-board computers in Latin America. I have also been working on *ProjectCaua.org*, a project to make computing more efficient and computers easier to use and to create part-time jobs for university students, so they can afford the living expenses of a university student, and therefore afford university itself. Both of these projects have been moving forward steadily but slower than I desired. While they are very, very close, I need to push them harder to get them going. I resolve that 2019 will be the year for that push and to get them to a self-sustaining level so that they can go on "forever."

I have always believed in education and have tried to make it as low-cost and available as possible. Since July of 2015, I have been the board Chair of the Linux Professional Institute (LPI). It has taken time, but I feel that LPI now has the direction, staff, and funding necessary to move forward in a big way. I resolve that 2019 will be the year that happens, so I can retire from that position in the year 2020.

In the last 10 years, I have had the most magical of events happen as I go from conference to conference. At almost every conference, and sometimes two or three times, I have had people approach me and say something along these lines:

- "I listened to you 10 years ago, I followed your advice, and I moved from being a "programmer" to being a "project leader" using FOSS. Now I travel around the world, and I make three times the money I made before."
- "I listened to you 15 years ago, and now I have used FOSS to help my company. I am now the CTO of my company."
- "I listened to you 10 years ago; now I am the CEO of my own company, and I employ 60 people. Thank you."

I have told many people stories about how FOSS improved my life and how FOSS can improve their lives. My friends have encouraged me to write the stories down and to publish them. 2019 will be the year for that.

2020 will also be a magical year. I will be seventy years old. Assuming that the US government does not decide to "disentitle" me from the money I have been paying into Social Security and Medicare from the age of 15, and that my retirement account from DEC/Compaq/HP does not run out of money, I may be able to start my retirement project for a community of FOSS advocates.

Every time I talk about this community, people beg me to let them join, but they will not be able to "join" … they will have to believe. ■■■

Creating a comic strip using open source tools

Creative Cartooning

Did you know that the Elvie cartoon strip in this magazine is created entirely using open source software? With this behind-the-scenes look at the process, you might be inspired to create the next big webcomic.

BY MARK CRUTCH

It was a little over five years ago that the Elvie cartoon appeared in the first issue of *Linux Voice*, but its genesis dates back far further. Way back in 1994, two friends, Mark (your correspondent) and Vince (a talented artist), installed CorelDRAW from a magazine cover disk in order to play around with vector graphics. A shared interest in cult TV and science-fiction led to their first comic series, *The Greys*, under the name of *Peppertop Comics* [1], with Vince creating the artwork to go with Mark's words.

Life got in the way, and the cartoon fell by the wayside. By 2009, however, Mark had been experimenting with Inkscape [2] – a free software program for creating vector graphics. They decided to migrate the old comics to the new software and try launching the series as a webcomic. It was a moderate success, with a fan following that encouraged Mark and Vince to continue pushing out new cartoons every couple of weeks. Keen to encourage their readers to start playing around with FOSS, they made the source files available for download. Each strip also included additional Easter eggs, which could often only be found by downloading the files and opening them in Inkscape.

A commission for a weekly cartoon in a local newspaper followed. Again, Inkscape was used as the main program of choice. But to give the new strip a different feel, they chose to use hand-drawn graphics that were scanned, then converted to vectors in Inkscape. This approach let them build up a large library of backgrounds, characters and props that could easily be reused again and again, speeding up the creation of the cartoons to maintain the pace of weekly publication.

When the *Linux Voice* team completed their crowdfunding campaign, Mark approached the editor to ask if the magazine needed a cartoon. The response was positive, but came with a warning: The deadline for the first magazine was only two weeks away! A few frantic evenings thinking up jokes and trying different character designs resulted in a rather rushed first outing for Elvie – named after the "LV" initials of the magazine. Once again, a new approach to creating the comics was used. Inkscape still served duty for the frames and speech bubbles, but the artwork was created using a graphics tablet and the MyPaint [3] application – an arrangement that has remained largely unchanged ever since (Figure 1).

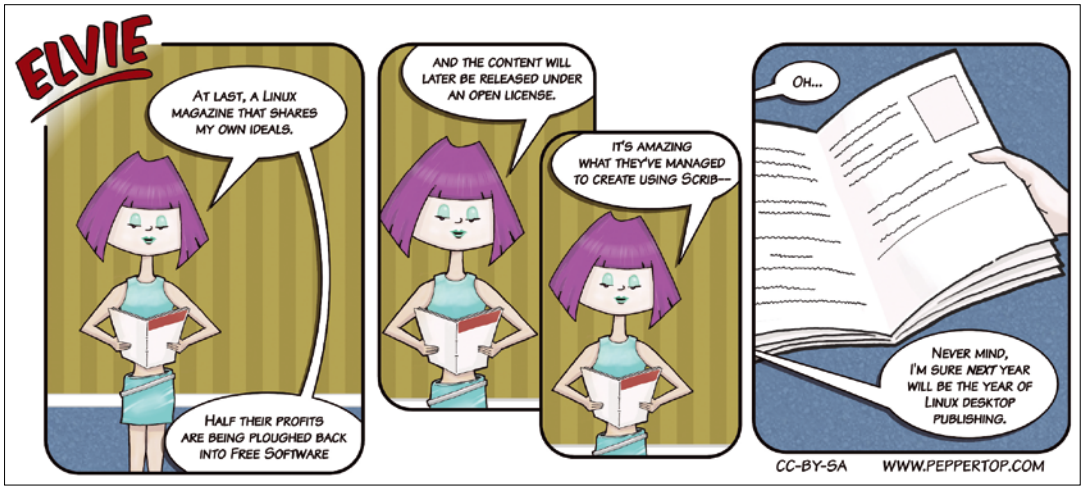


Figure 1: Elvie's first outing shows how her design was rushed.

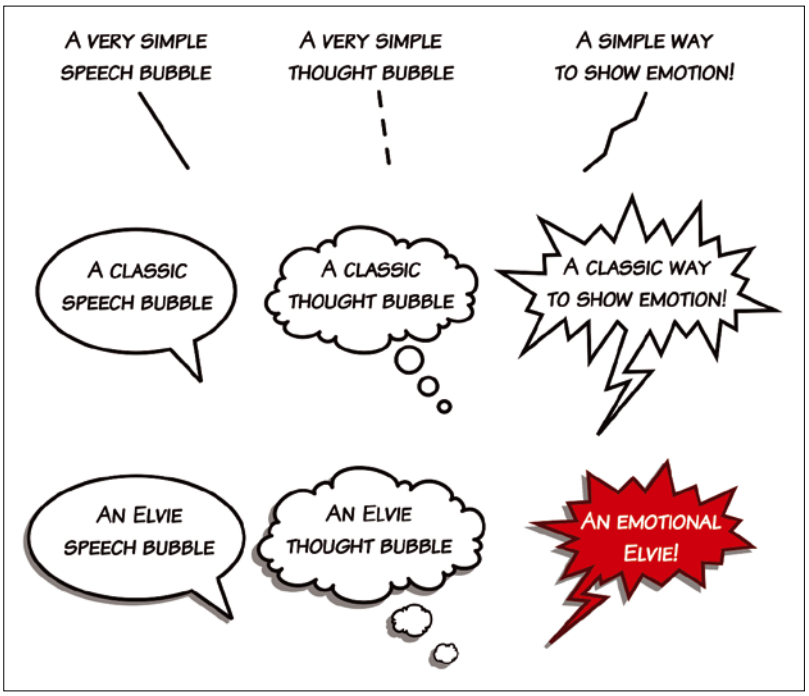


Figure 2: Examples of different types of speech bubble across three styles.

Choosing a Style

You might think that creating a comic strip or a cartoon is a simple process: you just have to draw the content for the panels, add a little text, and put the finished image online. But in most cases, your new strip will be part of a series. It may not be a "named" series, such as Elvie, but take the example of topical or political jokes in newspapers: Each artist will stick to the same style and position their signature in the same place. So before you start to draw anything, the first step is to decide on a style.

The more you look at other people's comics and cartoons, the more you realize that there are certain rules to follow. The shape and size of a humorous sketch for a greeting card is very different from the full-page image used with a graphic novel. A single panel joke might get away with no border, but once you need to show how things change over a period of time, it's typical to split the story into frames, each with its own outline. Speech could be as simple as a string of text, loosely attached to the character's mouth with a single line, or it might require a complex, colored bubble, shaped to distinguish between different types of speech or thought (Figure 2).

Inevitably, a time will come when your speech bubbles and your frame come into conflict. Usually this is a simple case of too much text for too small a frame, but, especially when designing for print, you may not have the ability to make the frame any larger. The first step is to edit the text down: Many Elvie strips start life as lengthy monologues or discussions between the characters before being pared down to just a handful of words. But editing can only get you so far before you reach a point where the impact of the joke or story is lost. Then you need to decide how to resolve the conflict between bubbles and frames. In the case of Elvie, the usual technique is to cut the bubbles off hard at the borders: this turns a curved shape into one with some straight edges, making it much easier to flow the text into the shape. Other comics let the bubbles float above the borders, breaking out of the confines of the panel to allow the extra text to fit.

If you're designing for an ongoing series, you'll probably also have to consider the standard "furniture" of the strip: its name, your own signature or contact details, and, possibly, an area to include license information. All this eats into the available space for the comic itself: The "Elvie" logo in this magazine has recently been rotated to make it slightly more horizontal in order to claw back a few precious millimeters in the first panel of each strip.

Don't forget to spend some time choosing your fonts! There are lots of comic fonts available commercially from sites such as *blambot.com*, and many sites make some fonts available at no charge. You'll want at least a standard dialog font for your characters' speech bubbles, but you might require fonts for sound effects and titles. The same text can have a radically different feel depending on the font you use, so it's worth giving this some thought.

For *The Greys*, Mark and Vince opted to use the standard Arial font for dialog. It's not a very comic-like font, but it nicely reinforced the clean vector style of the artwork. In the case of Elvie, a decision was made to use fonts that are themselves released under an open font license. Limiting the choice to open fonts greatly reduces the range of available typefaces, but it does mean that the cartoons are totally open, right down to the individual letterforms. The Open Font Library [4] is a good resource for finding fonts under open licenses, but it doesn't take long on the site to realize that there aren't many comic fonts available. For most cartoon work, therefore, you are likely to have to either pay for commercial fonts, or at least use fonts that are free of charge but not freely licensed. If there are any budding font designers out there, creating additional comic and effect fonts under an open license would be a great way to contribute.

The key thing is to pick a style that works for you. If you're not very adept with your tools, stick to something simple, with plain borders and basic speech bubbles. Elvie uses drop shadows on the speech bubbles, fades the first panel out under the logo, and has an awkward cutout in the frames to leave space for the license information. All of this is only possible because of the years of Inkscape experience Mark and Vince have acquired during their years creating *The Greys* – otherwise such graphical flourishes would just serve to slow the creative process.

Of course, rules are there to be broken – and in the case of cartoons and comics, a little anarchy can make the difference between a good comic and a great one. A strong style will register subconsciously with your readers, making any deviations from the norm even more jarring in a way that can be used creatively. In an example from The Greys, a strip in which a spaceship is under attack quickly starts to tilt and rotate the frames before finishing with the complete destruction of the regular page furniture, leaving the Creative Commons icons rolling across the screen, and the regular border literally hanging on by a few wires. But more subtle tweaks are possible, such as the usual red Elvie logo being replaced with a green "Elfie" equivalent for a cartoon about the game NetHack. That one also played with the borders, shaping them to look like the blockwork of a medieval castle at first glance, but in a way that also mirrored a thought bubble in the final panel, implying that the events of the earlier panels were all in Elvie's head.

Getting Creative

Having decided on your comic's style, it's time to start creating some content. Usually this begins with the script: a joke, a story, or even a rough plot idea. At this point, there's no need to go anywhere near any graphics software; it's just about getting the basics of the text written down. Any old text editor will do the job. Comics do sometimes use bold-italic text to emphasize key words in the speech bubbles, but that doesn't mean you need a full word processor. Simply wrapping any emphasized words *in asterisks* is sufficient. You should just hammer out your initial text as quickly as you can, to avoid forgetting that brilliant punchline or subtle plot point, with little regard to how it will finally appear in the comic. Here's an example of the initial notes for the recent "Cambridge Analytica" strip, from issue #213:

```
Cambridge Analytica joke:
James is shocked by social media
profiling so has posted warnings
to his friends -- on a social
media site!
```

With a rough idea written down, it's now time to work on the script in order to split it into individual panels. Are there obvious breaks or pauses in the conversation which warrant a change to another frame? Does the conversation bounce back and forth between characters too much to fit into a single image? Is there a clear punchline that deserves a panel of its own? The brief idea above was expanded into this:

```
[Panel 1: James is talking to Elvie
in front of a screen showing a
social media site]

James: With the Cambridge Analytica
scandal, I was shocked to hear how
extensively social media companies
can profile you, just based on your
normal activity on the site.

Elvie: So you've decided to delete
all your accounts?


[Panel 2: Close-up on James]

Friend: No... but I have liked,
shared and retweeted the news
stories, to warn all my friends.
```

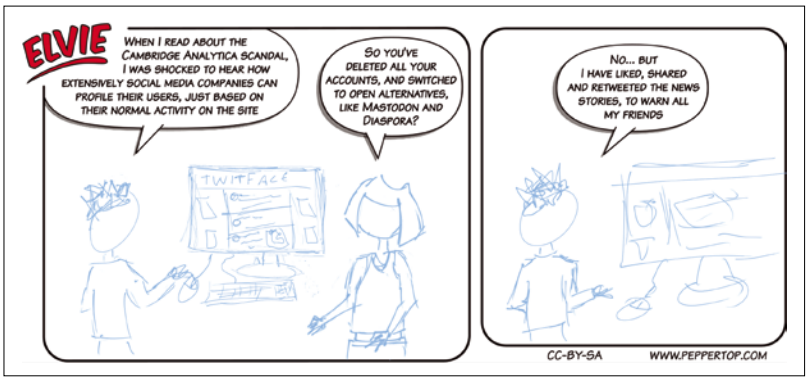


Figure 3: Mark's Elvie sketches still leave much to the imagination.

Now it's finally time to start drawing something! For an Elvie strip, this usually consists of Mark creating the overall layout in Inkscape. Panels are drawn, sizing them to approximately the right proportions based on what each one needs to contain. The text is put in place, and edited to fill the right space in the cartoon. Speech bubbles are drawn, but the tails are left as separate objects and any part that extends beyond the border is only approximately cut off or hidden. This is about getting an initial idea for how the content fits together, not trying to get it absolutely right from the start.

The Inkscape drawing is saved and will go on to form the basis of the final strip. Because the text, panels, and bubbles are all vector objects, they can be edited, tweaked, and moved around right up to the last moment – something that's much harder if you only use raster graphics for your comic. In the case of Elvie, however, the characters and sets *are* drawn as raster graphics, using MyPaint. To help get the sizes and proportions right for each panel, Inkscape's ability to export to PNG format is used to create an initial raster version of the layout, at the full resolution required for the final appearance in the magazine.

The resultant 600dpi image, measuring about 4000 pixels wide, is imported into its own layer in

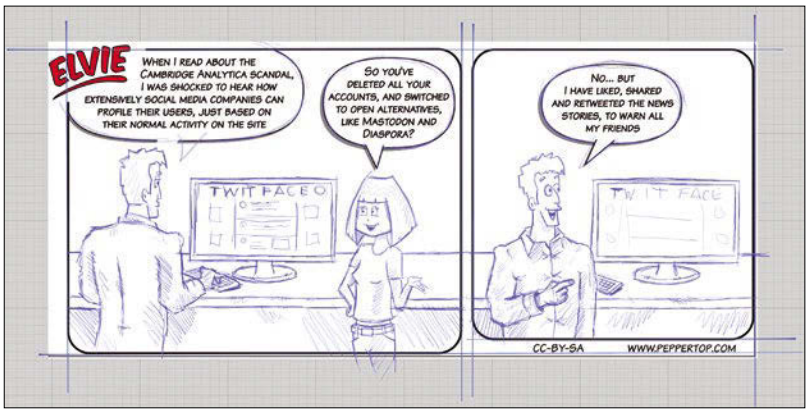


Figure 4: Vince's Elvie sketches get much closer to the final design.

MyPaint. A suitable brush is chosen in the program – typically some sort of pencil or marker pen – and then Mark starts to sketch the positions of the characters and other key elements. These are typically drawn in light blue: This is a hold-over from working on hand-drawn comics, where light pressure with a blue pencil lets the artist play with sketch ideas that can then be inked over in black. The blue pencil lines provide a clear contrast so, unlike with grey pencil lines, it's very clear where parts of the outline might still be missing. The blue lines are erased before the outline is scanned or filled and, again, the contrasting color helps to show where sketch lines have yet to be removed. You can see the result of this step in Figure 3.

It's time for a real artist to take over, so Vince sketches his own version of Mark's scribblings. As with most drawing software, MyPaint has support for multiple layers. Typically the new content is drawn on a layer above the original sketch. The panel borders and speech bubbles may be roughly reproduced in yet another layer, allowing the first sketch to be hidden completely without losing track of the areas of the cartoon that shouldn't be drawn into. The end result is still a sketch, but one that starts to get much closer to the final design (Figure 4).

This is the ideal time to make changes. It's rare for a sketch to be perfect the first time, so there's usually an iterative process in which tweaks are made and resubmitted for an additional round of discussions. This happens throughout the entire creative process, but if there are changes required, it's much easier to make them at the sketch stage rather than later on. Typically a sketch might be revised three or four times before the next step: outlining.

Look at almost any comic, cartoon, or graphic novel, and you'll notice that the characters and objects typically have a black, or at least dark, outline. This helps to define the edges of the elements, separating characters from background, or clothes from the people wearing them. In a 2D medium, it's easy for everything to look flat, as though it's at the same distance from the reader, so sharp edges help the brain to distinguish between parts of the image. For some styles, this outline is barely visible; for others – particularly comedy works that aren't trying for any semblance of reality – the outline can be quite thick. In the case of Elvie, there tends to be a thick outline around the outside of characters, and inner lines are much thinner.

Drawing outlines over the blue sketches takes place on yet another layer or, more often than not, a new layer for each part of the scene. In the case of the example strip, there were separate layers for Elvie, James, and the background. Some strips have many, many more. Splitting the parts into separate layers makes it easier to tweak the exact position of everything later on. This is also a good time to split the individual panels into separate files.

In addition to the outline layers, it's typical to also create a color or fill layer for each element.

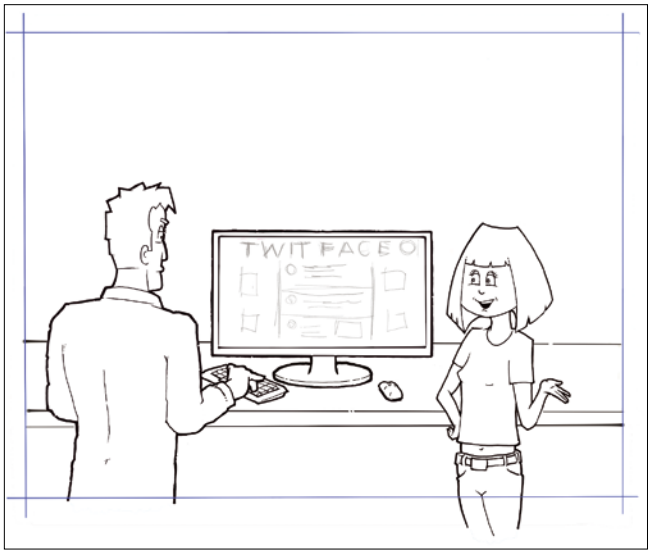


Figure 5: After outlining and filling, the final layout starts to take shape.

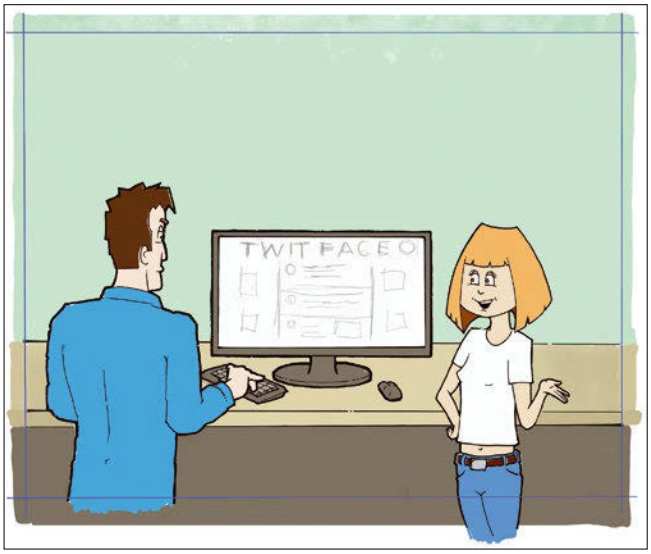


Figure 6: Flatting adds in the base colors for the final strip.

Each of these layers will eventually hold the final colors for the element, but for now they're usually filled with white simply to stop the outlines of other elements from showing through (Figure 5).

For most comics, there is a step known as *flatting*, in which individual parts of the image are filled with an appropriate flat color. This new color replaces the single white color from the previous step with an impression of how the final colors might look. Flat colors are also easy to select with the magic wand tool in graphics programs, allowing you to limit the extent of any changes you make as you further refine your image. For Elvie, the images are boldly colored, with little use of gradients or subtle color changes, so flatting usually gives something close to the final color palette for the cartoon (Figure 6).

There's one more step to perform in the raster software: the addition of highlights, shadows, and fine details. You can easily spend a lot of time on this step, trying to make your image look as good as possible. Or, if the style of your comic suits it, you can go for a quick-and-dirty approach with sharp transitions and scant regard for small excursions beyond the outline of your element. A very close look at any Elvie strip will likely expose sections of color that leave their own territory, but minor infractions are allowed to pass uncorrected as it helps to keep the loose and easy feel of the strip. In Figure 7, you can see the result of adding highlights and shadows to the image, as well as a close-up of one of the many color breaches.

After many trips back and forth between Vince and Mark (a typical Elvie panel will go through about 10-15 revisions), the raster image is completed. Nearly. Vince sends the MyPaint file to Mark to import into Gimp, where it's easier to finely adjust the positions of the individual layers. Any remaining layers with construction lines are turned off, and the image is cropped to give a clean edge. The final panels are exported as PNG files.

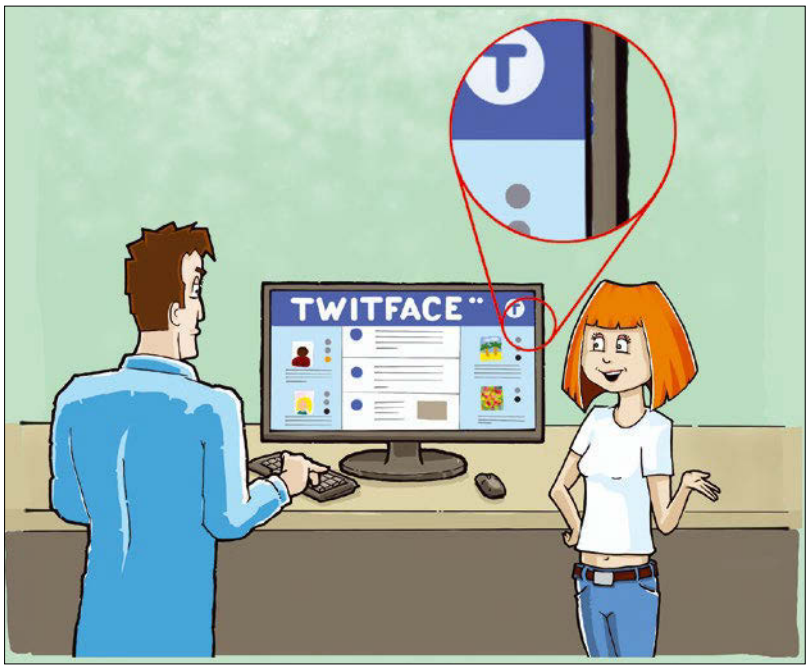


Figure 7: Highlights, shadows, and an inability to color inside the lines that most people beat in childhood.

Return to the Vector World

Back in Inkscape the PNG images are imported by dragging them into the window. Once again layers are used: This time a single layer holds all the raster images to allow them to be stacked easily behind the speech bubbles and borders. Inkscape defaults to 90dpi for rasters, resulting in the imported images being far too big for the panels. Zooming out is quickly achieved by holding the Ctrl key while spinning the mouse wheel, so it's trivial to reach the zoom level required for the edges of the raster to become visible. Now the image can be selected, and the resize handles used to scale it down to suit the panel it's being

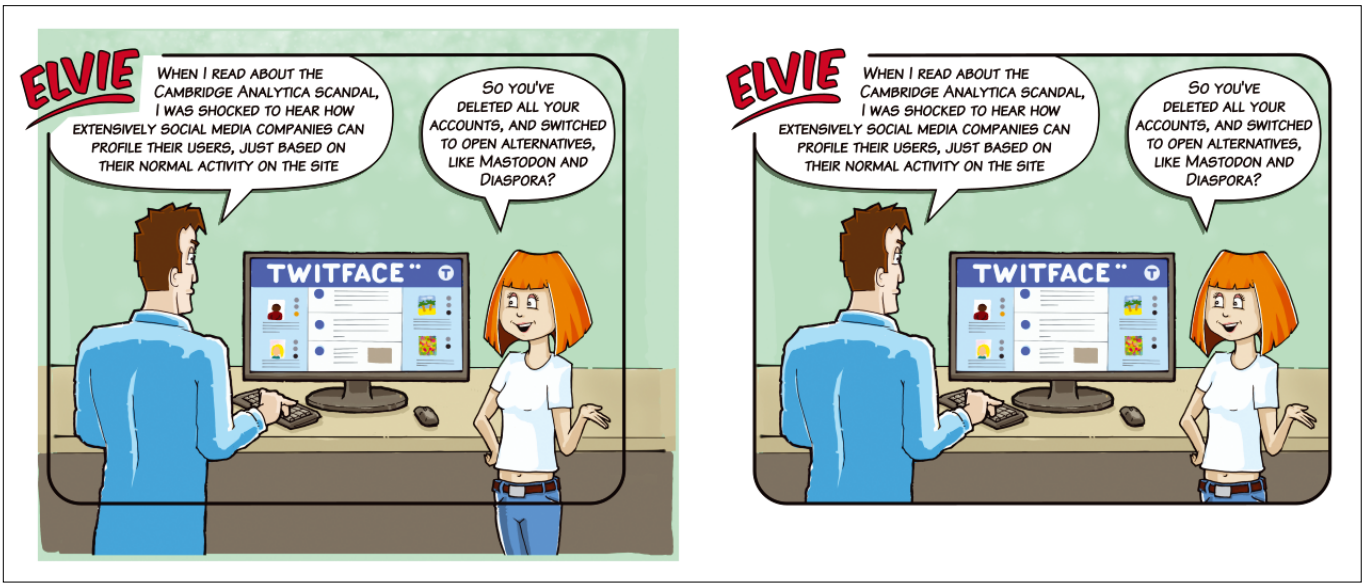


Figure 8: Before and after clipping a panel in Inkscape.

put into. Once again, the Ctrl key comes into play, this time ensuring that the image is scaled in proportion. When the size is close enough, Mark zooms back in for fine positioning and size adjustment. The aim is to get as much of the image content as possible into the panel while not clashing with the speech bubbles – and making sure there's at least a little of the image that extends beyond the panel border.

The final strip needs to have crisp, clean edges, so any part of the image that does extend beyond the border has to be clipped. This is achieved by duplicating the border shape, cutting it from the strip (Ctrl+X), and then pasting back onto the raster layer, but at exactly the same coordinates it was removed from, using the *Edit | Paste In Place* option in Inkscape (Ctrl+Alt+V). The pasted border remains selected, and the raster can be added to the selection with a Shift-click. From the menus, *Object | Clip Set* will set the duplicated border as a "clipping path" for the raster image – meaning that any part of the PNG that falls outside the border will not be drawn. Thus each panel image is constrained to its own area and doesn't leak out beyond the edges (Figure 8).

At this point, the drawing work is almost complete. There may be some final tweaking of the text and speech bubbles, but the leaders are still separate objects, sitting on top of the speech bubbles rather than joining to them. Selecting both the leader and the bubble and then using the *Path | Union* menu entry combines the two objects into a single composite path to give the final shape that an Elvie bubble requires. If the bubbles extend beyond the panel edge, you'll need another round of clipping.

Although the artistic content is finished, there's still some practical work to do before the strip is sent for publication. The SVG format that Inkscape uses may be a web standard, but there's not even full compatibility between different browsers, let alone with a professional print workflow, so sending the Inkscape file directly isn't an option. Although it should be possible to export in another vector format that's more familiar to Adobe software – such as EPS or PDF – experience has shown that even those formats can expose issues between different platforms and programs. Instead the cartoon is exported as a 600dpi PNG image. There is some loss of fidelity with this approach: Remember that the raster images for the panels are also 600dpi, but have been scaled arbitrarily so an original pixel in the raster may not fall in the same position as one in the export. Inkscape interpolates between pixels to overcome this mismatch, and the resolution is high enough that, although the export may not technically be perfect, the end result is easily good enough.

With a large PNG image in hand, it's time for one last trip to Gimp. Through their history of comics appearing in print, Mark and Vince are well aware that the colors that are displayed on screen can be significantly different from those that end up on the paper. For Elvie, the specific colors aren't terribly important – nobody's going to notice if her hair has a touch more cyan than intended – but the overall lightness of the whole strip does make a difference. Compared with Mark's PC, comics tend to print darker than expected, so a preset color adjustment curve is used to brighten the finished cartoon a little before we finally export it and send it to the magazine.

Some Time Later…

So far, our Elvie strip has taken, typically, a month to produce – but it still won't appear on shelves for several more weeks and won't get to some subscribers for months! If you've ever wondered why there are few topical Elvie jokes, and even some that are long past warranting the "topical" label, that's the reason.

Those overseas subscribers also explain why the Elvie website lags so far behind the published cartoons. From the early days of *Linux Voice*, Mark and Vince have been putting all their Elvie source files onto GitHub [5] but were requested to include a few months' delay to ensure that all the paying readers of the magazine had a chance to see the strips before they were released to the world at large.

What this means, in practice, is that some months after the strip has been sent for publication, Mark has to pull together the final versions of each source file, tidy them up, remove unnecessary layers, and fix up any spelling mistakes in file or layer names. The Inkscape file is edited to add license information and other metadata; then a PNG version of the cartoon is exported once more and compared to the version sent to the magazine to ensure that no important parts have been missed or broken during the clean up.

In Gimp, the PNG is scaled to suit the website. The colors are *not* adjusted in this version, since most people will view it online with a relatively bright screen. Apologies to anyone with a properly calibrated monitor who thinks the strips are all too dark! Finally the source files are bundled together, committed to Git, pushed to GitHub, the website updated, and a post made to Twitter and Facebook to announce the arrival of a new cartoon. More than a year after first writing the joke, Mark can finally mark the strip as finished.

Writing the process down like this makes it sound like a lot of hard work, but creating comics and cartoons can also be immense fun. If you've ever been put off the idea by your lack of a Wacom tablet or not wanting to get onto the rolling treadmill of Adobe's subscription products, hopefully you've now got an idea of how to create

the next great webcomic using only a Linux box and a mouse. There are a lot of steps to learn and technical issues to solve, but all that comes with time and experience. For now, just start writing and drawing with whatever tools are at hand, and the rest will surely follow.

Cartooning on a Budget

If there's one thing to take away from this article, it's that creating a cartoon doesn't have to cost a lot of money. Many of the available tutorials are based on Adobe Photoshop and Illustrator, or they use specialist commercial comic software such as Manga Studio. But with a little imagination, it is possible to apply similar techniques to free and open source applications. The Greys were drawn with Inkscape and a mouse, yet they have appeared in print and were even featured in a museum exhibition. A cheap scanner was used for the hand-drawn newspaper strip, but even a photograph from a mobile phone would be sufficient provided there's enough contrast for the Trace Bitmap command in Inkscape to do its thing.

Elvie required some expense but not much. The first 13 strips were produced using a Monoprice branded graphics tablet that cost less than £50. At the time, a little effort was required to get it working on Linux, but these days it is just plug-and-play. Mark still uses this same tablet for the editing work he does, although Vince has since been promoted to a £250 Wacom tablet for the proper artwork.

In practice, the differences between spending £50 and £250 aren't as great as you might think. Even the budget tablet has enough pressure sensitivity for cartoon work, but the wireless connectivity of the Wacom does make it a little easier to get the tablet into a comfortable working position. The Monoprice tablet also lacks an "eraser" on the opposite end of the stylus, meaning that switching between two tools can only be done by selecting icons or pressing keyboard shortcuts, not by flipping the stylus over.

If you're working on a budget, don't dismiss a secondhand tablet as a viable option. Linux can still support professional tablets that connect over an RS232 serial connection, so you may be able to pick up an eBay bargain if you're prepared to add a suitable port to your PC – or use an adapter. If you're handy with a soldering iron, the WaxBee project [6] can help you convert an old serial or ADB tablet to USB via a microcontroller, so your Linux box need never even know that it's talking to a legacy device.

No article about comics and FOSS would be complete without mention of David Revoy's beautiful comic strip, Pepper&Carrot [7] (Figure 9). Charting the adventures of a young witch (Pepper) and her feline familiar (Carrot), it's notable for its light-hearted plots and imaginative fantasy setting – but even more for the stunning artwork that makes up the bulk of each comic. David creates his imagery mainly using Krita, with Inkscape serving duty for the speech bubbles and panels. The lavish and detailed images take a long time to create, but David is able to work on them thanks entirely to sponsorship on Patreon and similar sites. The finished comics are released under a Creative Commons Attribution license. Pepper&Carrot is probably the finest example of a FOSS-created comic – and a great counterpoint to anyone who thinks you have to use proprietary software to create professional looking cartoons. ■■■

Figure 9: Pepper&Carrot is one of the most beautifully drawn webcomics there is – and it's all created with FOSS!

Info

[1] Mark & Vince's back catalog of cartoons: *http://www.peppertop.com*

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

[3] MyPaint: *http://mypaint.org/*

[4] The Open Font Library: *https://fontlibrary.org*

[5] Elvie GitHub repositories: *https://github.com/Peppertop*

[6] WaxBee project: *https://github.com/popbee/waxbee*

[7] Pepper&Carrot: *https://www.peppercarrot.com/*

Plan and perform your daily tasks

Focus on Your Work

Is your to-do list filling up faster and faster? Go For It! helps you work your way through your task list.

BY ERIK BÄRWALDT

Linux supports many heavyweight applications that help you plan and implement projects [1, 2]. But if you are looking for a small tool that simply helps you to complete your upcoming tasks quickly, Go For It! [3] is the right choice. How easy is Go For It? The refreshingly sparse user interface contains only a timer and two lists: unfinished and completed tasks.

Installation

The software has only made its way into a few distributions so far. For Ubuntu and its derivatives, you can add a separate software repository to the system and update your package lists with the commands found in Listing 1, lines 1 and 2. Next, install the program with the command from line 3. The package manager automatically creates a matching entry in the desktop menu structure.

This completes the installation. Arch Linux and its derivatives have a ready-made package in their repositories, and Elementary OS lists the program in its AppCenter. There are no ready-made packages available for RPM-based systems yet.

If you run a Linux derivative that supports Flatpak, however, the good news is that there are ready-made containers. The project's GitHub page [4] provides installation instructions.

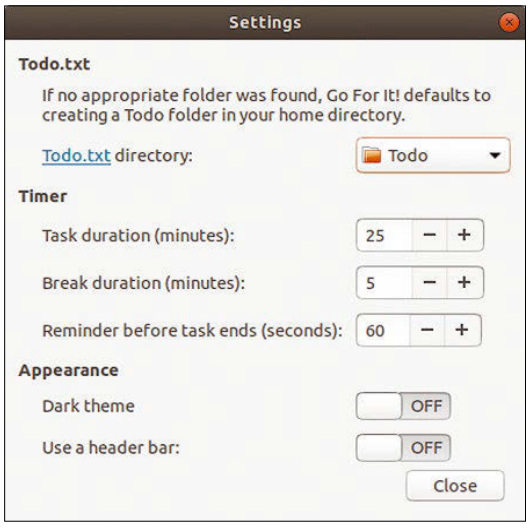


Figure 1: The Settings dialog lets you change the appearance or configure paths for the lists.

Listing 1: Installing Go For It! on Ubuntu

```
01 sudo add-apt-repository ppa:go-for-it-team/go-for-it-daily
02 sudo apt-get update
03 sudo apt-get install go-for-it
```

Intuitive

After starting the application, a small window opens on the desktop minus a menubar and with three buttons (*To-Do*, *Timer*, and *Done*) arranged side by side at the top edge. Above these buttons to the right is a gear icon, which you can use to open the Settings and Help windows. At the bottom, you'll find the *Add new task* field.

In the Settings dialog, you can define the appearance and the To-Do lists' path, as well as modify the Timer configuration if necessary (Figure 1).

The Timer offers three displays: In addition to the scheduled time to complete a task, you can also define breaks and reminders to notify you when a deadline is approaching. The dialog does not contain any further options.

Here We Go

Once you have completed the configuration, enter a task in the *Add new task* field in the main window. The program then transfers this text to the To-Do window, displaying the individual tasks in a vertical list (Figure 2).

To define the time assigned for working on a task, click the *Timer* button while the task is highlighted. In the Timer display that now appears, use the plus and minus buttons to set the expected duration for the current task. Then click *Start* at the bottom of the window to enable the timer (Figure 3).

You can edit the task details while the clock is running. If you have completed a task earlier than expected, click on the *Done* button in the bottom left corner of the Timer window. The tool then adds the entry to the list of completed tasks in the *Done* window and removes it from the *To-Do* window.

Coffee Break

If you want to take a break, just press the *Pause* button in the lower right corner of the Timer window. You can restart at any time with the same timer status.

If you need to interrupt a task unexpectedly, you can exit the current task by pressing *Skip*. The

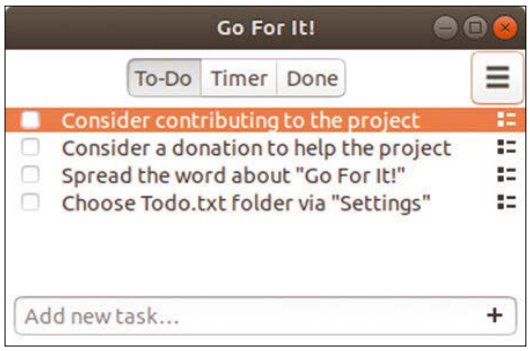


Figure 2: Avoiding unneeded gimmicks, you can find your tasks easily in the *To-Do* window.

software now pauses the timer and simultaneously displays a message from the working environment's system tray letting you know that you can now take a break.

To end the break, click on *Skip* again. The software now displays a message confirming the end of the break in the system tray for a few seconds. You are then returned to the Timer display with the current task and the allocated time. If you press *Start* instead of *Skip*, the timer resets and counts down the full time defined in the settings.

Done!

The application collects the completed tasks as a list in the *Done* window. The program displays these tasks in strikethrough text and with a check mark in the box to the left of the task. If you click to uncheck, the entry is reactivated and moved to the *To-Do* window.

This function is particularly useful if you need to complete the same task more than once. In the list of completed tasks, however, a task that has been completed several times is only shown once (Figure 4).

Since the list of completed tasks can become very long, the tool offers the option to empty the list completely at the push of a button. The *Clear Done List* option is available in the settings for this purpose.

The software usually saves the completed tasks after closing the program so that the old data can be seen again when the program is reopened. This is useful for repetitive daily tasks that you can transfer from one list to the other as shown. Only the timer is reset, as it always starts with the value defined in the settings.

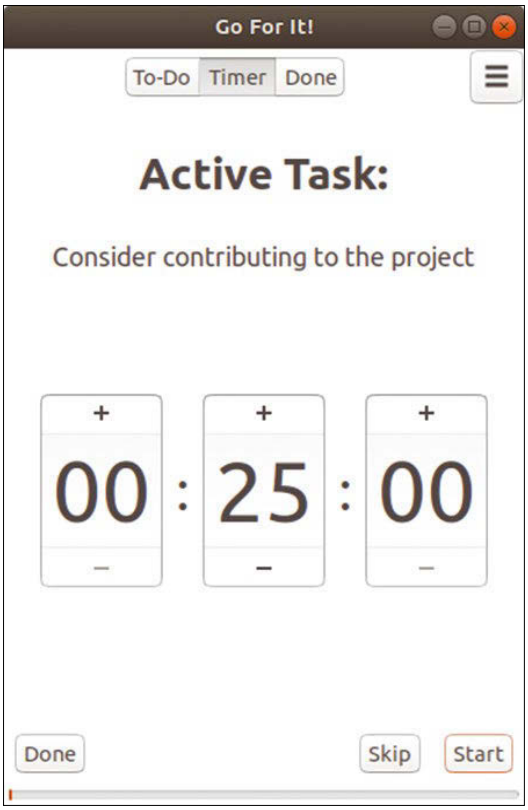


Figure 3: When the timer is running, it's time to focus on your work.

Mobile

Go For It! is available for different platforms: There is a smartphone app and PC versions. If you use Go For It! on more than one platform, you do not need to enter your tasks each time. The software stores completed and pending tasks in simple text files, which it keeps in a defined subdirectory.

The text files are easily transferred when switching between devices, which gives you easy access to the lists regardless of the platform.

Conclusions

Go For It! helps you bring order to your daily workload. The strictly time-oriented application makes it easier to concentrate on the tasks at hand without confusing the user with superfluous functions. ■■■

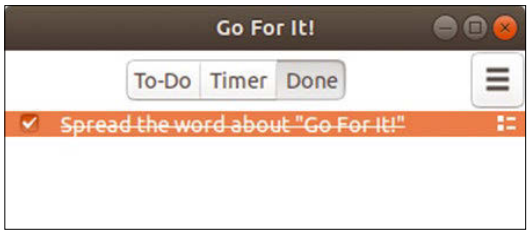


Figure 4: Go For It! displays all the completed tasks in strikethrough text.

Info

[1] Task Coach: *https://www.taskcoach.org*

[2] Nitro 1.5.1: *http://nitrotasks.com/legacy.html*

[3] Go For It!: *http://manuel-kehl.de/projects/go-for-it/*

[4] Go For It! GitHub page: *https://github.com/mank319/Go-For-It*

FOSSPicks

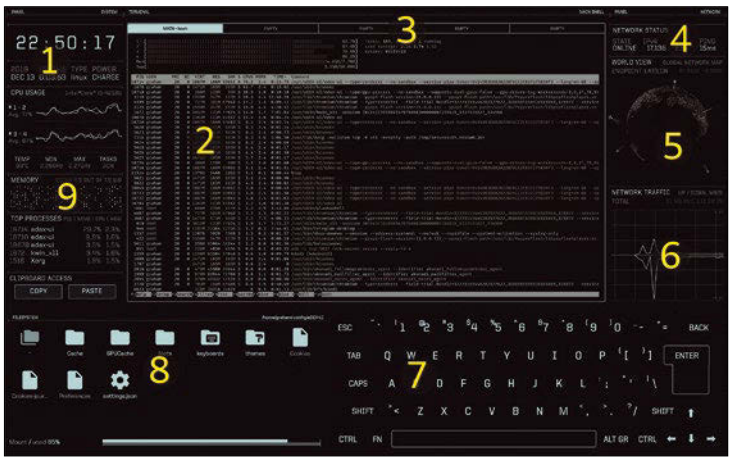
Sparkling gems and new releases from the world of Free and Open Source Software

This month, Graham discovered the spontaneous knotting of an agitated string while trying to untangle the mess of wires behind his PC and quickly gave up. BY GRAHAM MORRISON

Hacker desktop

eDEX-UI

We admit this choice is mostly for the eye candy rather than the functionality. But this has to be one of the best looking and most original desktop replacements you can use on your Linux box. Its appearance is surely what we all imagined computer interfaces would look in the same year that *Blade Runner* was set – a combination of 1980s retro vectors with a touch interface, and waveforms for no other reason than they just look good. eDEX-UI is all of this and more. It calls itself a "desktop application resembling a sci-fi computer interface," and it has taken more than naming inspiration from the DEX-UI interface in the *Tron: Legacy* film sequel to the original *Tron*. If you've not seen the film, the user interface (UI) feels a lot like something from the XCOM: Enemy Unknown game or the famous touch interface in *Minority Report*. Back on Linux, the launch animation is a faux-retro boot sequence that ironically looks just like a booting Linux box. You then get a personalized welcome message, also uncanny, before the main application appears in its high-DPI vector glory.



1 Local details: Time, date, and charge state, above CPU usage charts. **2 Terminal:** Your default terminal appears here. **3 Terminal instances:** Use these buttons to switch between different terminal tabs. **4 Network:** Reports your status, connection speed, IP address, and ping. **5 Geolocation:** See your location, and the physical location of your connections, on a rotating vector globe. **6 Network traffic:** Tracks both the data coming in and the data leaving your computer. **7 Touch keyboard:** Use a touch screen or mouse to type without a physical keyboard. **8 File manager:** Browse your local filesystem. **9 Task monitor:** See which processes are taking the most resources, and visualize how much memory is being used.

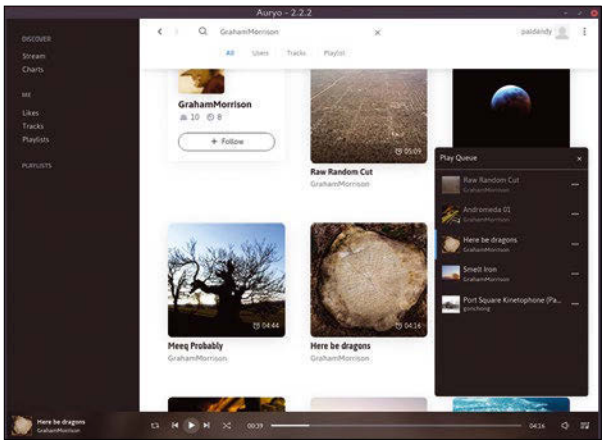
The functional part of the UI is a terminal placed in top center of the full-screen display. This is your default terminal, and you can switch between various tabs using the buttons above. As this really is your default terminal, it's perfectly possible to do real work here and forget that the terminal is really embedded within a selection of constantly moving and updating panes. On first glance, these panes look like graphical frippery, but they turn out to be both useful and interesting. In the top left, for example, is the time and details about your uptime, date, and CPU usage. Beneath these is a graphical memory map showing which sections of your computer's memory are being used. Over on the right is a vectorized rotating 3D image of Earth with small pins placed in specific geographical locations; these turn out to be the endpoints of your current Internet connections. We were able to connect to servers where we know their geographical location and confirm the accuracy. It's surprisingly useful to see to where your computer is currently connected. Something like this would be useful outside of eDEX-UI, especially if you're concerned about what servers are being accessed by your various services.

The lower third of the display contains a graphical file manager and a keyboard. The keys on the keyboard light up as you type, but you can also use the mouse to click on the keyboard to generate input. The keyboard is comprehensive enough to include the cursors and function keys and works perfectly if you're using a touch screen or VNC to Android. The file browser does exactly what you'd expect and lets you click around your local files and folders just as you might with Dolphin. It currently lacks integration with the command line, but it's useful for viewing your current working directory's contents. The whole application actually feels very effective. While it's currently a CPU hog on a high-resolution display, we'd love to see a less resource-hungry version that could potentially replace the desktop completely.

Project Website
https://github.com/GitSquared/edex-ui

Online music player

Auryo



Auryo is a desktop version of the SoundCloud web UI, written in the JavaScript-inspired (and Microsoft-maintained) TypeScript.

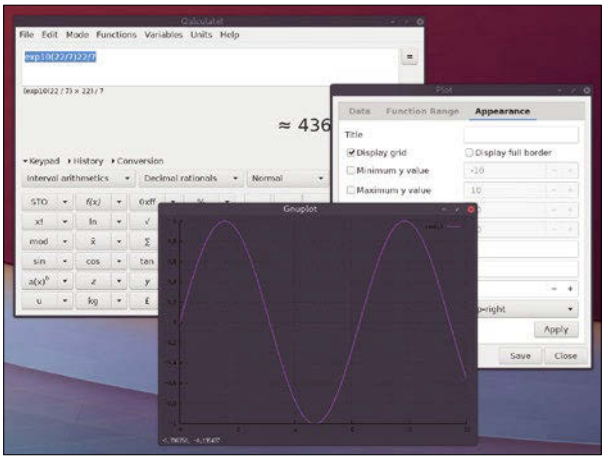
Even though it's a commercial service, and the Internet is swamped with services attempting to turn themselves into a social network, SoundCloud is still a useful place to share your music, at least until we can reinvent the idea with a federated and open source version. But it's impressive that, even with its initial focus on people sharing their own music creations, SoundCloud has become a streaming service to compete with the likes of Spotify and Tidal thanks to some big names dropping rare tracks into the site. What still makes SoundCloud slightly different, and worth exploring, is that it retains an experimental edge to the music. That means if you're into more left-field or home-grown music, you're more likely to find it on SoundCloud than anywhere else.

Which is where the beautiful Auryo can help. This is an application that removes browser requirements from listening to SoundCloud, letting you enjoy music through a native desktop application. Unfortunately, you will need a SoundCloud account, which isn't required if you're simply browsing and listening through a web browser, but it does mean any playlists you save, or tracks you star, will be available both in the application and via the web interface. You can then take advantage of the desktop audio integration and keyboard control available within Auryo. Its simpler UI design also makes it easier to discover new music, as well as tracking anyone following your own creations. There are no options for the application itself, which says a lot about how intuitive the design is – Auryo doesn't need any configuration options, because it's easy to use and already looks great. If you're bored with the mainstream music applications and looking, or listening, for something a little different, it's definitely worth a download.

Project Website
http://auryo.com/

Desktop calculator

Qalculate!



Augment your computing potential with a calculator that has more options than the New York Stock Exchange.

Despite its terrible name, Qalculate! has more mathematical functionality than almost any other desktop calculator has in their advanced views. It's so crammed full of features, it's difficult to get a mental grasp of its entire capabilities, especially on first glance. It looks like a standard calculator application that can easily replace your desktop's default choice. For that reason, switching over shouldn't be too much of a mental burden. In this mode, it operates just like a regular calculator on your desk, albeit a decent scientific calculator with more than eight digits of precision. Tap out the numbers, and you see your answer. It's only when you start exploring the myriad of drop-down menus that it starts to become complicated.

Every menu seems to have 20 options, and every button seems to hide another menu. Even the Edit menu includes scary headings like *Manage Variables* and *Insert Vectors*. The Functions menu contains shortcuts to common functions split into many different categories, from microeconomics to trigonometry, and there's a plot window that will render the output into a Gnuplot window. Many of the button-driven functions of the main calculator have drop-down menus to extend their functionality, too. The *sin* button, for example, lets you choose between *Hyperbolic Sine*, *Inverse Sine*, and *Inverse Hyperbolic Sine*. There are also file operations, such as loading or saving a list of comma-separated values, updating exchange rates, and a very useful periodic table. If you're willing to put some effort into learning where everything is, this is a calculator that seems to have almost everything the desktop mathematician might need. Qalculate!, as its name almost implies, is the Gimp of mathematical applications.

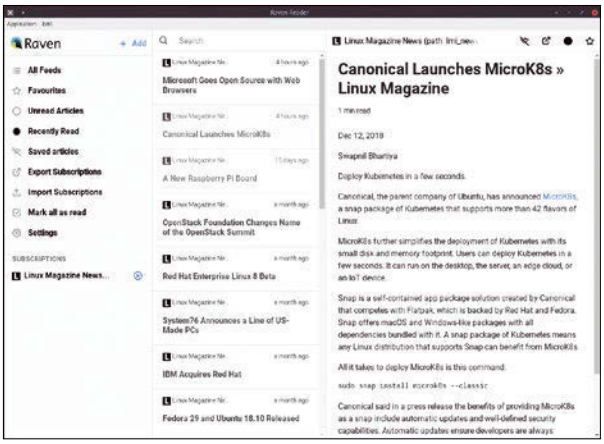
Project Website
http://qalculate.github.io

RSS reader

Raven

Each time we cover an RSS reader, we briefly explain what RSS is and why it's so important. This is because it always feels like we're looking at the last RSS reader to be made and that RSS is on the cusp of disappearing forever. But if the constant stream of new RSS applications is anything to go by, we don't have to worry. And this application, Raven, is further evidence that RSS is going through something of a renaissance. It may be Electron-based, and cross-platform, but its simple design and function makes this a perfect tool if you're new to RSS readers, especially if you're new and need a solution for other operating systems, too.

The main view is very similar to the old Google Reader. It's split into three horizontal columns; the first allows you to add and manage your various RSS subscriptions. For example, you can mark a source as a favorite, save stories for later, and import or export a complete set of feeds via an OPML file. This is very useful if you're still hanging on to those old collections exported from Google Reader. The middle column is the headline of the story from the RSS itself, while the final column is the downloaded or cached story linked to from the selected title. It's impressively clean and easy to navigate. The options to save, favorite, or open in a browser are perfect for most usage, and there's support for offline reading. The dark mode is also very useful for late night reading or if your desktop theme uses dark colors. This great UI design is going to be a real advantage for users new to RSS, as the RSS itself remains almost transparent with only the downloaded content being shown in the main window, rather than the sometimes too brief sentences contained within the RSS itself.



If you've been looking for a modern, well-designed application for making sense of the web, Raven is a great choice.

Project Website

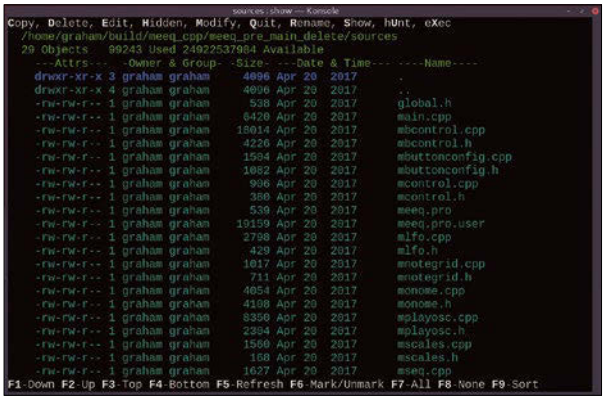
https://github.com/mrgodhani/raven-reader

CLI file manager

DF-SHOW

There are many different file managers that run from the command line, but the one most of us think of, and the one we're most likely to use, is Midnight Commander. Midnight Commander was created by Miguel de Icaza way back in 1994, before he started the Gnome project, before Ximian, Mono, and Microsoft. Midnight Commander was itself inspired by Norton Commander, which was an essential tool for anyone working in the pre-Windows world of Microsoft DOS. And so too is DF-SHOW, which is a new command-line file manager inspired by another DOS classic, Larry Kroeker's DF-EDIT. This becomes apparent as soon as you launch the application via its `show` command. The files are listed in that warmly familiar teal color that seems, even now, the color of the early 90s DOS era.

Unlike Midnight Commander, the default view isn't split between source and destination directories. Instead, it's a flat list of the files and directories in the launch location. It looks a lot like the output from `ls`, but you navigate through the files using the cursor keys or the function keys. A top "menu" of commands display the initials you can use to navigate into directories, or to mark files and folders to copy, delete, rename, show, and edit. The latter will launch your default text editor, allowing you to edit files in-place, just like in the 90s. You can even "hUnt," which means pressing *U* to search through the contents of the files using a case-sensitive search term. The `show` function is implemented as a separate command, `sf`, which can be useful if you want to quickly see the contents of a file without worrying about what its specific type might be, all of which can be configured if necessary.



View, edit, and copy files, as well as change their permissions and execute them like it's 1994.

Project Website

https://github.com/roberthawdon/dfshow

Synthesizers for ADLMIDI and OPNMIDI

ADLplug

Whether you enjoy creating chiptune music, or you're looking for a new sound source, these two classic chip emulators are worthy additions to your sound library.

Here are two software synthesizers that are part of a much larger project, although you're unlikely to have heard of it unless you enjoy listening to a certain era of video game music. The "upstream" project consists of two command-line music players that emulate two specific frequency modulation (FM) audio chips from a bygone era, the Yamaha YMF262, also known as the OPL3, and the Yamaha YM2612, or OPN2. Both were used in early computers and games consoles, with the latter being in Sega Genesis (also known as Mega Drive) and the former commonly found in even the earliest Sound Blasters, where FM chips could be found for years. If you played PC games in the mid-to-late 90s, there's a good chance the music came from one of these chips. As with the SID chip from the Commodore 64, these chips offer plenty of audio nostalgia for people who played the games that used them. Consequently, there's a large audience of devotees playing those old tracks on their modern machines with the command-line players.

These two painstakingly engineered emulations have been used to create two software synthesizers, turning those old sounds into fully fledged, MIDI-capable plugins that you can use within any VST-compatible Linux host. It's just like taking one of those early chips and putting them at the heart of a real hardware synthesizer, and they sound wonderful. They're both FM, but the brilliant thing about FM synthesis is that it sounds very different from classic oscillators. They're great for plucky strings and warm brass sounds, for example, and sound quite different from most other synths. Both have a similar architecture and almost identical controls, but they keep their individual character and are an essential installation if you're into any kind of chiptune music creation.

Project Website
https://github.com/jpcima/ADLplug

Software synthesizer

synister

If you can hunt down the binaries, installing this synth is as easy as learning how to play it.

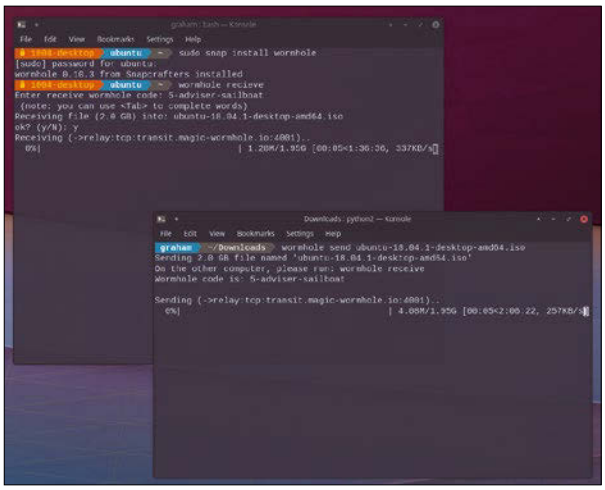
The state of Linux audio has improved hugely over the last few years, thanks to the many applications that now help us make music. But we still need more tools to help us make sounds; in particular, we need more software synthesizers. There are a few amazing ones (see above), also notably Helm and VCV Rack, but we don't get any of the mainstream releases you'll find on macOS and Windows, even when they're open source. Which is why we're trying to shine some attention on synister, a wonderful open source VST plugin that will work on any VST-compatible Linux host (we used Ardour). It hasn't been updated in some time and doesn't even officially support Linux, but it's worth the trouble of either building it yourself or tracking down a pre-built binary that you can simply drop into your VST system path.

The synth is built around three oscillators and can generate a beautifully rich set of sounds. The layout design is also easy to understand, which makes it ideal for beginners as a first synth – especially on Linux where there is less choice. In particular, modifying the sound of one element with the output of another, a process known as modulation, is brilliantly handled with a simple destination drop-down menu next to each source or destination. Use one of the oscillator source modulation menus to select an low-frequency oscillation (LFO) to modulate the pitch, for example, or control the amount of filter with the aftertouch value from your MIDI keyboard. Each section of the synth is easily separated by colorful horizontal sections, including the oscillators, envelopes, LFOs, filter, and effects, just as you'd find on a modern hardware synth, as is the sound, which is like a 70s-era analog polyphonic classic.

Project Website
http://the-synister.github.io/

File transfer

Magic Wormhole



Use a simple command-line tool to securely transfer files between two machine.

Wormhole really is one of those tools that, once you've used it, you'll wonder how on earth you managed to get by without it. This is because Wormhole solves one of those ancient problems that have been around on Linux since we first started to connect computers together with null modem connectors and a couple of lengths of wire. Back in those olden days, you might have used ZMODEM to transfer a file across the serial connection. If you then upgraded to a connection capable of TCP/IP, you could use FTP to transfer files, and then came HTTP. In the modern age, `scp` is often the best choice if you're accessing an SSH server and want to securely transfer files, or `rsync` if you want to copy folders or perform incremental backups. But all of these solutions suffered and suffer from the same problem. The remote machine needs to be running a server of some kind, and you need to know the remote address of the machine you're wanting to access.

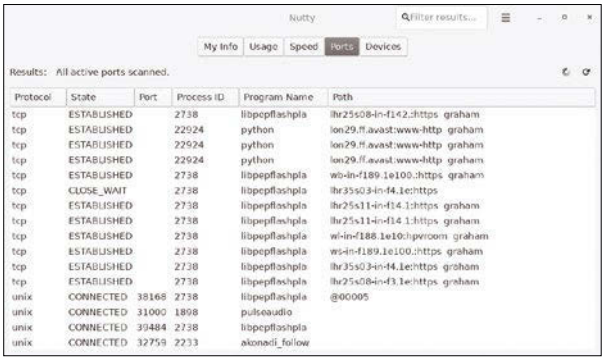
This is why Wormhole is so brilliant. If you want to send a file, simply type `wormhole send` followed by the filename. In the output, you'll see a secret phrase that you'll need to share with whoever you want to receive the file. The phrase is constructed from a few words, so it's easy to say or copy without mistakes. Your recipient then simply types `wormhole receive` followed by the same phrase, and the transfer will start immediately downloading the file from one machine to the other, no server required. It's perfect for transferring between virtual machines or people sitting next to each other, when every other solution requires more than a set of Python scripts.

Project Website

https://github.com/warner/magic-wormhole

Network monitor

Nutty



Easily monitor which processes are using your network without resorting to the command line.

Network monitoring isn't easy for everyday users. If you've ever taken a look at the packets captured by Wireshark, they're incredibly complex and difficult to understand. And so too are the associated command-line tools. It's easier for distros to keep users at arm's length, letting them worry about wireless network strength and streaming quality rather than presenting users with a stream of data that's difficult to interpret. But much like with a task manager or memory monitor, there are many good reasons for ordinary users to be better informed about their network consumption. We just need a decent application to turn that burden of knowledge into something easy to use, yet powerful.

Nutty isn't quite that application, but it gets close. It's a network monitoring tool that will attach itself to one of your interfaces, such as your wireless connection, and then perform a series of monitoring processes or tests. The main view is a tabbed interface with the first pane showing general details about your hardware, such as your hostname, network driver, IP address, and firmware. The second tab, *Usage*, attempts to detect which processes are using your bandwidth. This can be very revealing, especially if you've forgotten about that Nextcloud daemon quietly syncing your files to the cloud in the background. The third tab uses `speedtest-cli` to test the speed of your connection, while the fourth lists all your local ports being used on the network and the processes attached to those ports. This is likely the most useful if you want to see which processes are accessing your network. The final tab will perform a network scan, much like `netstat -sP` on the command line. And that's really what this great little tool is all about, encapsulating some of the most useful output from disparate and sometimes difficult to use command-line tools to help you monitor your network.

Project Website

https://github.com/babluboy/nutty

Roguelike

Shattered Pixel Dungeon

This is a roguelike RPG with beautiful 8-bit style graphics, fabulous retro sound, and a brilliantly addictive explore-and-combat style gameplay. It's actually a fork of Pixel Dungeon, created initially to improve on the original game's "quirks," but it's blossomed into a fully fledged alternative title with the same style of play. And unlike many games that name-drop "Rogue," Shattered Pixel Dungeon actually plays like the original Rogue, complete with fog of war, dungeon levels, and items and magic. But the first thing you notice about it, at least on the Linux desktop, is the aspect ratio of the default main window as soon as the application is launched. It's best described as an elongated portrait, and there's a good reason for this. Shattered Pixel Dungeon is primarily a game to run on your phone.

But because Shattered Pixel Dungeon is open source, it's not just available on F-Droid for your phone. You can also install it on your Linux desktop. On the Linux desktop, the aspect ratio is only temporary. Just drag the corners of the window to change the play area to any aspect ratio or size. Equally modern is the internal scaling that sizes up or down those chunky pixels to best suit the density of your screen. This is a common theme in modern games that take their inspiration from the procedurally generated nature of Rogue games, and it also makes the game both difficult and addictive. This is what Shattered Pixel Dungeon does,

Playing either on your phone or desktop, the retro graphics in Shattered Pixel Dungeon are exquisite.

because you will die. But you'll also want to play again. There are four different character classes, eight different subclasses, five different areas of each dungeon to explore, and over 150 items. The style of click-and-move, click-and-hit gameplay is addictive enough to pull you through the random design. And like the original Rogue, it's completely free.

Project Website
http://www.shatteredpixel.com/

Even more Roguelike

KeeperRL

KeeperRL is another Roguelike, but this time, rather than being the adventurer exploring the dungeon, you play the role of an evil wizard who creates a dungeon to trap naive adventurers. This part is somewhat similar to Peter Molyneux's classic Dungeon Keeper, and there's no doubt that the main game mode is intended to be this dungeon building. The creation mode is a little like using the old drawing package, Deluxe Paint. You have a palette of possible symbols on the left, including structures, monsters, and items, and you paint and populate them into the four-way scrolling main display. But there's also an explore mode, too, where you can play the game as if you were one of those adventurers encountering the dungeon for the first time. Or you could create a dungeon and share it with other players, who will then use their adventurers to explore your creation.

KeeperRL is a fascinating project, both in the way the game is played and in the way it's being designed and developed. The best way of playing the game, for example, is to pay for it. This is because, despite it being open source, the commercial version includes the awesome artwork and music. But this shouldn't put you off downloading and building the open source version either. Its graphics are built from high-resolution ASCII symbols, but they're colorful and effective, and it's much more in style with the original Rogue. Apart from the graphics and sound, it's exactly the same game, so you

KeeperRL is the inverse of Shattered Pixel Dungeon (above), because the game is about creating a dungeon for adventurers to explore.

can play it to see if it's worth paying for the commercial version, or even whether you want to help with development. The game is very much in-progress, and much of the design has yet to be finalized. But there's enough here to play with, and there's already a sizable community built around exploring new features and each other's dungeons.

Project Website
https://github.com/miki151/keeperrl

Complex Containers

Discover the data structures that make your shell scripts really powerful.

BY MARCO FIORETTI

In the first installment of this tutorial [1], I described the main properties of shell variables, which are the basic data structures of the Bash shell. This time, I am going to show you how to handle more complex containers for your data, which make much harder (and cooler!) things possible: Bash arrays.

Note: Unlike almost anything else in the other installments of this tutorial, most of what you will learn here only works with Bash v4 or later. If you are using any reasonably modern Linux distribution, this should not be a problem. However, just to be sure, check your version by typing

```
echo $BASH_VERSION
```

at the prompt.

Strictly speaking, a Bash array is still a variable, meaning a data container with a unique name, at least in the script or programming scope in which it is called. The Advanced Bash-Scripting Guide [2] contains more shell array usage examples than you may ever use or want to know, but you do not need to learn them all to make productive use of these structures.

A Bash array's defining property is that each array can contain multiple values, each with its own distinct identifier. Arrays are the tools that Bash puts at your disposal to aggregate multiple objects and treat them as one entity, while preserving the ability to distinguish among them. Basically, you can use arrays to keep all the values of any imaginable "set" or "group" together.

In practice, the first thing to know about Bash arrays is that there are two types: plain arrays (which I will simply call arrays) and associative arrays (hashes). Each array or hash can contain values of different types, without built-in limits to their size.

The difference between arrays and hashes is the way their single elements are referenced. Arrays address each element with a numeric identifier, starting from zero. Inside hashes, on the other hand, each value is associated with a key, which may be any text string.

That difference corresponds to two, potentially very different use cases. Use arrays when all that matters is ordering (i.e., whenever all you need to know about each component of a set is its position in the sequence). When what matters is the relationship between pairs of elements, use hashes instead. To understand the difference, compare Table 1 and Table 2. Although Table 1 is ordered alphabetically for our reading convenience, its purpose is to associate each city with one of its own intrinsic properties, regardless of other cities' properties. Table 2, on the other hand, is all about ranking. If Madrid had only three residents and Rome had two, only the first table would change. Therefore, you use a hash to store Table 1 and an array to store Table 2.

Now, I'll show you how to process arrays and hashes, before dealing with a more complex, real-world script that uses them. First of all, you have to declare each structure with the proper flag:

```
declare -a eu_capitals_by_population # ↵
  -a => array
declare -A population_of_eu_capitals # -A => hash
```

Then, you fill arrays or hashes with this syntax:

Table 1: Population of EU Capitals (2012)

Berlin	3.5M
London	7.4M
Madrid	3.2M
Rome	2.6M

Table 2: EU Capitals, Ranked by Population (2012)

1) London
2) Berlin
3) Madrid
4) Rome

```
eu_capitals_by_population=↵
  ('London' 'Berlin' 'Madrid' 'Rome')
population_of_eu_capitals=( [Berlin]="3.5M" ↵
  [London]="7.4M" [Madrid]="3.2M" [Rome]="2.6M" )
```

You may also populate arrays or hashes dynamically, with the output of other scripts, Linux commands, or text files, as long as the data is in the necessary format. For example, use

```
eu_capitals_by_population=$( ↵
  ./some_other_script.sh )
```

to initialize the array of EU capitals with the output of some `otherscript.sh`. The simplest way to load the current line of a plain text file you are reading in your script into `$myarray` is as follows:

```
read -a myarray <<< $line
```

(See the first installment of this series [1] and then references [3] and [4].)

Another, perhaps faster, way to load values from files or scripts into a plain array is the built-in Bash command, `mapfile` [5]. I will not cover `mapfile` here, partly because it is not very portable and partly because, very frankly, I have never found myself compelled to use it in actual work. However, those reasons shouldn't stop you from checking out `mapfile`.

To use the current element of an array or hash, write them as follows:

```
#> echo ${eu_capitals_by_population[3]}
Rome
#> CAPITAL='Berlin'
#> echo ${population_of_eu_capitals[$CAPITAL]}
3.5M
```

Instead of `$CAPITAL`, I could have used the string `Berlin`, with or without quotes, to achieve the same effect. However, for both arrays and hashes, do *not* forget the curly braces; otherwise Bash will not interpret what is inside the square braces properly!

To delete an element, just pass its identifier, formatted as above, to the `unset` command. Don't forget the index, or the key if it is a hash! Calling `unset` with just an array or hash name will erase that whole data structure. Instead, to quickly remove all and only the elements that match a certain pattern, use regular expressions:

```
#> echo ${eu_capitals_by_population[@]}
London Berlin Madrid Rome
#> declare -a ifbrexit=( ↵
  ${eu_capitals_by_population[@]/London/} )
#> echo ${ifbrexit[@]}
Berlin Madrid Rome
```

The `$ifbrexit` array is created by copying into it all the elements of `$eu_capitals_by_population` that do *not* contain the string `London`.

Once arrays or hashes have been created, you can add elements both singularly or in bulk. Here is one example for hashes and two for plain arrays:

```
#> population_of_eu_capitals[Lisbon]='1M'
#> eu_capitals_by_population+=↵
  ("Paris" "Bucharest")
#> eu_capitals_by_population[9]='Sofia'
```

As you can see, you can append to an array multiple elements (or just one element) with one command, but with whatever index you desire. An alternative way to append elements to an array is as follows:

```
eu_capitals_by_population=(↵
  "${eu_capitals_by_population[@]}" ↵
  "Paris" "Bucharest")
```

This is more flexible than the previous example, because it also may be used to prepend elements to an array or to merge multiple arrays into one.

To list all of an array's elements, in order, or its existing indexes (adding `!` in the right place), use the `*` or `@` wildcards. If you are only interested in one slice of the array, specify the starting element and how many elements to fetch:

```
#> echo ${eu_capitals_by_population[*]}
London Berlin Madrid Rome Paris Bucharest Sofia
#> echo ${!eu_capitals_by_population[*]}
0 1 2 3 4 5 9
#> echo ${eu_capitals_by_population[@]:2:3}
Madrid Rome Paris
```

Using the same trick with a hash would return all its values and, keys, in whatever order Bash stored them internally.

As far as arrays are concerned, please have another look at the second-to-last result, which is a direct consequence of what I did to the `$eucapitals_by_population` array: First, I dumped four cities in the array, then I appended two more with one command, and finally I deliberately created a seventh one (Sofia) with a non-consecutive index. This brought the total number of elements of that array to seven, but without defining any indexes between `6` and `8`. In programming courses, arrays that do not need to have consecutive arrays are called sparse; it is up to you to be aware of this Bash array feature.

In general, omitting the index of an array makes Bash use only its first element. The first two com-

mands of this sequence would both load into `$UK_capital` the string `'London'`

```
#> UK_capital=eu_capitals_by_population
#> UK_capital=eu_capitals_by_population[0]
#> echo ${#eu_capitals_by_population}
6
#> echo ${#eu_capitals_by_population[@]}
7
```

while the last two commands return the number of characters in `'London'` and, the number of elements in the whole array.

Processing Whole Arrays or Hashes

Complex data structures would be almost useless if one could not quickly loop through them and process one element at a time, with as little code as possible. In fact, the syntax to do just that is relatively simple and basically the same for both arrays and hashes. The only meaningful difference is in the output: Arrays are by default sorted, and consequently processed, from (numerically) lower to higher indexes. Hashes, instead, are processed in whatever order the shell stored each key-value pair internally. For example, this loop over the `$population_of_eu_capitals` hash

```
for city in "${!population_of_eu_capitals[@]}"
do
  echo "Population of $city is ${
    population_of_eu_capitals[$city]}"
done
```

returns just what one would expect, because the exclamation mark in the first line means "use all the keys of this hash." However, the lines are not sorted in any way:

```
Population of Berlin is 3.5M
Population of Madrid is 3.2M
Population of Rome   is 2.6M
Population of London is 7.4M
```

I am going to show you how to get around this hash limitation in a moment, but first I want to show you one last trick I found online [6], which can be useful in many different situations.

Can you easily generate a hash that is the "reverse" of an existing hash (i.e, another hash with keys used as values and vice versa)? Of course you can! Just adapt the following code

```
declare -A eu_capital_with_population
for city in "${!population_of_eu_capitals[@]}"; 
  do
    eu_capital_with_population[${
      population_of_eu_capitals[$city]}]=$city
done

for ppl in "${!eu_capital_with_population[@]}"; 
  do printf "%s people live in %s\n" "$ppl" 
  "${eu_capital_with_population[$ppl]}"; done
```

which, in our case, returns

```
7.4M people live in London
2.6M people live in Rome
3.2M people live in Madrid
3.5M people live in Berlin
```

Of course, this specific example is not especially helpful, because I could have generated the same output from the original hash. However, swapping keys with values can be useful in some situations. I'll now move on to more complex stuff and a complete, useful script.

Sorting and Multidimensional Arrays

Bash is very powerful, but when it comes to sorting arrays and hashes, especially in non-basic ways, it is no match for Perl (probably other languages, too). That said, I hope to prove that Bash is more than adequate for basic and not-so-basic data structure processing.

When you begin using arrays or hashes, two questions surely arise: How do you create (and process) multidimensional ones (e.g., hashes of hashes)? And how do you sort the keys of hashes when you loop through them?

The easiest (but not the only [7]) answer to both questions, is to cheat.

You can emulate hashes of hashes by using ordinary hashes with "composite" keys and sort the result of looping over a hash after it is finished.

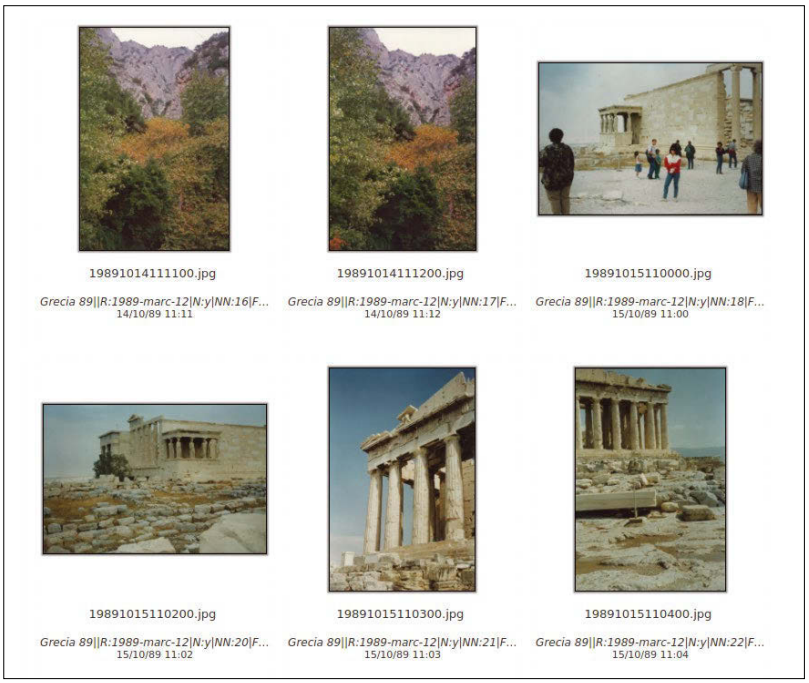


Figure 1: Digital photography is great, but it brings so many files, with so much metadata, into your computer that GUI applications alone have a hard time aggregating and presenting the information in useful ways.

Both tricks are shown in a script that follows.

A Photo Archive Statistics Generator

Years ago, I became the de-facto maintainer of the complete, unified archive of digital photographs (Figure 1) ever taken by my entire extended family (this is what you get when relatives discover you are a Linux geek!). In order to make sense of some tens of thousands of files, I needed summaries of how many photographs were already geotagged, which places had been already included in the archive, and of which places we had the most photographs. Listing 1 shows a partial output of the script I put together to scratch this itch. The script is shown in Listing 2.

Listing 1: Partial Output of Listing 2

```
Distribution of photos By year:

Total photos taken in 1989:  125

  Grecia, Atene, Acropoli   :   39
  Grecia, Capo Sunio        :   20
...

Total photos taken in 1990:  256

  Sardegna, Isola Rossa     :   88
  Roccacalascio             :   65
...

Total photos taken in 1998:   24

  San Francisco             :   24

Total photos taken in 1999:   40

  home, sweet home          :   23
  in-laws home              :   17

Total photos taken in 2000:  396

  in-laws home              :  111
  home, sweet home          :  106

...

Total photos taken in 2001:  184

  home, sweet home          :   65
  Sardegna, Cannigione      :   57
  parents home              :   31
  in-laws home              :   19
  kids daycare              :   12

10 most photographed places from 1989 to 2001 :

home, sweet home            :  197
in-laws home                :  147
Sardegna, Isola Rossa       :   88
...

By place, alphabetic order:

Abruzzo National Park       :    5
Calcata Vecchia             :    2
Campo Ceraso, Volubro       :   37
....

Photos not geotagged yet: 1
```

Before explaining the code, please take note of the hidden, but crucial message in Listing 1: Using arrays and other Bash script features allows you to quickly extract quantitative and qualitative data from very large, messy files of all kinds, in formats that greatly facilitate further processing. It would be easy, for example, to rearrange the numbers above as one `.csv` file that could be used to generate charts. And you could extract with the same techniques any number or text, from stock market quotes to book titles.

The code for the photo archive statistics generator script in Listing 2 is simple to understand – once you know how digital pictures are geotagged and how I catalog them on my computer (Figure 2). Here is the bare minimum you need to know to keep going.

GPS coordinates are stored inside JPG files in the following format, which can be read with the `exiftool` program:

```
GPS Position : 37 deg 58' 18.02" N, ➥
  23 deg 43' 33.76" E'
```

In any folder I have already geotagged, I keep a plain text index called `.tags.summary.` that lists each picture's filename, author, and place together with other data in pipe-separated fields:

```
19980110110000.jpg | = ➥
  | Marco|San Francisco| other fields here...
```

In other words, the files only contain coordinates, which I need to map to human-readable place names that are only found in the indexes. Moving on to the script in Listing 2, the first 10 lines declare all the hashes and arrays that are needed to store data. The loop in lines 15 to 44

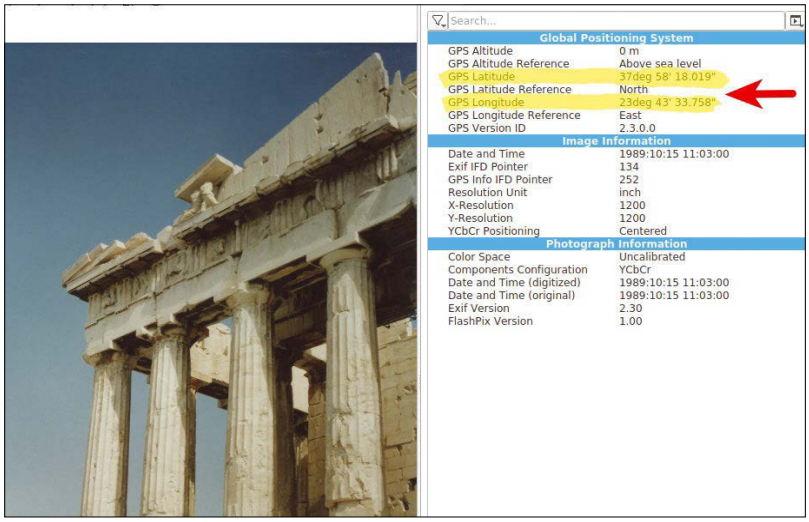


Figure 2: digiKam knows that each geotagged digital file includes its own GPS coordinates, in plain text format. However, it takes a shell script like Listing 2 to use that data in any meaningful way.

finds all the folders that contain indexes (i.e., those folders already geotagged) and does three things.

First (lines 16 to 19), it saves into a temporary file named after the folder subject (`$TMPDIR/$n`) only the index fields that correspond to the name and place of each picture, in this format:

```
19891010110000.jpg | Grecia, Atene, Acropoli
```

This is what happens in line 19, while line 17 extracts the year value from the folder name. The purpose of lines 21 to 28, which read the newly created temporary file, is to load into the complementary hashes, `$coordinates_of` and `$name_of`, each place's GPS coordinates and name. To get the `$photo_place`, I split the current `$line` of the file using a pipe as a separator (`cut '-d|'`); to get the GPS coordinates (line 25), I use `grep`

Listing 2: Photo Archive Statistics Generator

```
01 #! /bin/bash
02
03 declare -A coordinates_of
04 declare -A name_of
05 declare -a photos_per_year
06 declare -A photos_per_place
07 declare -A photos_per_year_in_place
08
09 PHOTODIR=/home/z/photo/marco
10 TMPDIR=/tmp/photostats
11
12 cd $PHOTODIR
13
14 for summary in  `find . -type f -name ".tags.summary*" |
   cut -c3- | sort  `
15 do
16   d=`dirname $summary`
17   y=`echo $d | cut -d/ -f1`
18   n=`basename $summary | cut -c2-`
19   cut '-d|' -f1,4  $summary | sed -e 's/ *$//g'| grep -v
     '|$' | grep -v ^# | sort | uniq > $TMPDIR/$n
20
21   while IFS= read -r  line
22   do
23   photoname=`echo $line | cut '-d|' -f1 | sed -e 's/ //g'`
24   photoplace=`echo $line | cut '-d|' -f2 | sed -e 's/^ *//g'`
25   gps=`exiftool $d/$photoname | grep '^GPS Position' |
     sed -e 's/^.*: //'`
26   coordinates_of[$photoplace]=$gps
27   name_of[$gps]=$photoplace
28   done < $TMPDIR/$n
29
30   for photo in `find $PHOTODIR/$d -type f -iname "*jpg"`
31   do
32   let "total_photos++"
33   echo "Processing photo n. $total_photos"
34   current_position=`exiftool $photo | grep '^GPS Position'
     | cut -d: -f2 | cut -c2-`
35   if [[ ! -z "${current_position// }" && ! -z "${name_
     of[$current_position]}"  ]]
36   then
37     let "photos_per_year[$y]++"
38     let "photos_per_place[${name_of[$current_position]}]++"
39     let "photos_per_year_in_place[$y ${name_of[$current_
   position]}]++"
40   else
41       let "not_geotagged_photos++"
42   fi
43   done
44 done
45
46 printf "\n\nDistribution of photos\n\nBy year:\n\n"
47
48 for year in "${!photos_per_year[@]}"
49 do
50   printf "\nTotal photos taken in $year: %4.4s\n\n"
     ${photos_per_year[$year]}
51   for place in "${!photos_per_place[@]}"
52   do
53   printf "\t\t%-45.45s : %4.4s\n" "$place" ${photos_per_
     year_in_place[$year $place]}
54   done | sed -e 's/: *$//' | grep ':' | sort -t ':' -k2,2rn
55 done
56
57 printf "\n\n10 most photographed places from %s to %s
   :\n\n" \
58     `echo ${!photos_per_year[*]} | sed 's/\s.*$//'`     \
59     `echo ${!photos_per_year[*]} | sed 's/^.*\s//'`
60
61 for place in "${!photos_per_place[@]}"
62 do
63   printf "%-45.45s : %4.4s\n" "$place" ${photos_per_
     place[$place]}
64 done | sort -t ':' -k2,2rn | head -10
65
66 printf "\n\nBy place, alphabetic order:\n\n"
67
68 for place in "${!photos_per_place[@]}"
69 do
70   printf "%-45.45s: %4.4s\n" "$place"
     ${photos_per_place[$place]}
71 done | sort
72
73 printf "\nPhotos not geotagged yet: $not_geotagged_photos\
   n\n"
74 exit
```

to filter the line starting with `GPS Position` (with `grep`) from the output of `exiftool` and remove the initial string with `sed`. In this way, `$name_of` and `$coordinates_of` will be filled with values like these (using bogus coordinates for brevity!):

```
coordinates_of['San Francisco']=
  '37 deg N, 75 deg W'
name_of['37 deg N, 75 deg W']='San Francisco'
```

The loop in lines 30 to 43 completes the gathering of raw data by looking again at all pictures in the current folder (line 30). Line 33 increases the `$totalphotos` counter. Line 34 saves into `$currentposition` the GPS coordinates of the current pictures, in the same format used to fill the `$name_of` hash.

Line 35 checks that `$currentposition` is not an empty string (i.e., that the current photo did contain a `GPS Position` field), and that such a position already exists in `$nameof`. If that is not the case, the counter of photos not yet geotagged is incremented (line 41). Otherwise (lines 37-39), I increase the counters of photos taken in the current `$year` and in the current place (line 38) and then resort to the first dirty trick I mentioned above. I need a hash of hashes as follows:

```
YEAR: 1989:
    Pictures in
      Greece: 49
      home:   25
YEAR: 1990:
    Pictures in
      Sardegna: 81
      home: 54
etc...
```

To emulate it in Bash, I create a standard hash, but with composite keys like `"1989 home"` (line 39). As dirty as it is, it does the job. Once all folders have been scanned, all that happens from line 46 to the end is pretty printing of the totals previously stored in the several hashes. Only six lines need a thorough explanation. Line 54 is where, looping over both the indexes of `$photos_per_year` and the keys of `$photos_per_place`, I use the "composite key" trick to retrieve, from `$photos_per_year_in_place`, the total number of pictures taken during `$year` in each `$place`. Lines 54, 64, and 71 all show how to sort the output of a loop over hash keys: You feed it to the `sort` command purging and filtering with `sed` and `grep` if needed. Please see `sort`'s man page [8] to see how the `-k` option does the numeric sorting.

Last, but not least, the arguments of the print function in lines 58 and 59: The problem here is to find the smallest and biggest indexes of the sparse array `$photos_per_year`. You already know by now that writing `'${!photos_per_year[*]}'` returns *one* string like `"1989 1990 2001"`, containing ordered but *not* consecutive numbers. Feeding that string to `sed` removes everything after the first space (line 58) or everything before the last space (line 59), and that is how I generate the "from 1989 to 2001" part of the script output (Listing 1). Wasn't that fun? Happy hacking! ■■■

The Author

Marco Fioretti (*http://mfioretti.com*) is a freelance author, trainer, and researcher based in Rome, Italy. He has been working with free/open source software since 1995 and on open digital standards since 2005. Marco also is a Board Member of the Free Knowledge Institute (*http://freeknowledge.eu*).

Info

[1] "Tutorial – Shell Scripting" by Marco Fioretti, *Linux Magazine*, issue 219, February 2019, p. 84

[2] Advanced Bash-Scripting Guide: *https://www.tldp.org/LDP/abs/html/arrays.html*

[3] Splitting with IFS: *https://stackoverflow.com/questions/10586153/split-string-into-an-array-in-bash*

[4] Reading a file into an associative array in Bash: *https://stackoverflow.com/questions/40387649/reading-a-file-into-an-associative-array-in-bash/40388130*

[5] mapfile: *https://www.computerhope.com/unix/bash/mapfile.htm*

[6] Reversing hashes: *https://stackoverflow.com/questions/26856070/bash-populate-an-associative-array-using-a-loop?rq=1*

[7] Multidimensional arrays: *https://unix.stackexchange.com/questions/78624/bash-multidimensional-arrays-and-extracting-variables-from-output*

[8] sort man page: *http://man7.org/linux/man-pages/man1/sort.1.html*

Track Record

Natron allows you to create eye-catching effects and combine different video clips in surprising ways, letting you build up your clips like a pro.

BY PAUL BROWN

In last month's issue [1], we had a look at Natron [2] and saw some basic things we could do with it. But Natron does much more than basic. In fact, doing complex effects and composites involving several tracks and techniques is what Natron does best.

To illustrate what you can achieve with Natron, read on for a couple of practical examples.

Transition

This effect lets you use an animation to transition from one clip to another. The animation in this case is a running person on a green background. The effect you want to achieve is having the character run from right to left, dragging the second clip in over the first one [3].

If you are a regular reader of *Linux Magazine*, this should be familiar to you. We did this effect using *FFmpeg* [4] and again using Kdenlive [5]. I'll quickly show you how to prepare the transition animation and then cut to the business of building the transition in Natron.

So, first things first, you'll need to get your animation ready. Step one is to split the animation clip into its individual frames:

```
ffmpeg -i runner.mp4 -ss 00:02:09 ➥
  -t 00:00:07 runner_frame%04d.png
```

Broken down, this instruction is made up of the following components:

- `runner.mp4` is the video from which you are going to extract the frames.
- `-ss 00:02:09` is the starting point of the sequence of frames you want to extract – in this case, two minutes and nine seconds into the clip.
- `-t 00:00:07` is the length of the sequence in frames – in this case, seven seconds.
- `runner_frame%04d.png` gives each frame a sequential name, like `runner_frame0000.png`, `runner_frame0001.png`, `runner_frame0002.png`, `runner_frame0003.png`, etc.

Next you need to make the green background transparent and cut out the running man. The ImageMagick command

```
convert runner_frame0000.png -fuzz 40% ➥
  -fill none -opaque "#13ff09" ➥
  -trim alpha_cropped_runner_frame0000.png
```

will do both on your first frame, so that Listing 1 will do it on all the frames.

Get rid of (or move) the original clip and the original `runner_frame*.png` frames, leaving only the cut out frames with transparent backgrounds in their own directory. Next, you can use the *customtransitions.sh* script you saw in [5] to finish off the preparation of the animation frames. To save time, you can download the script [6].

Figure 1 shows the desired result.

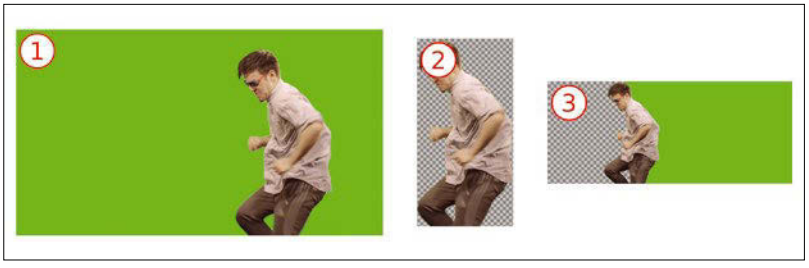


Figure 1: You want to process the frames from the original animation (1) so that they end up looking like (3).

Natronizing

Time to fire up Natron.

A new blank Natron project contains only a *Viewer* node. The first thing you're going to do is right-click in an empty area in the *Node Graph* (lower left-hand pane) and go to *Image*. Pick a *Read* node. This will open a file explorer. Navigate to where you have saved your first clip in the transition sequence and pick that. The *Read* node will automatically connect to the viewer, as shown in Figure 2. The first frame from your clip will show up in the preview window.

Listing 1: Transparency and Cut Out

```
for i in runner_frame0*.png;\
do convert $i -fuzz 40% -fill none -opaque
   "#13ff09" -trim alpha_cropped_${i};\
done
```

Next bring in the animation frames you created with *customtransitions.sh*. To do that, create a new *Read* node and, when the file explorer appears, choose *Sequence* from the dialog's bottom left-hand corner. Navigate to where `customtransitions.sh` stored your processed frames and choose the `chroma###.png` series of frames.

If Natron connects the sequence to the *Viewer*, disconnect it by clicking and dragging on the arrow connecting the two nodes.

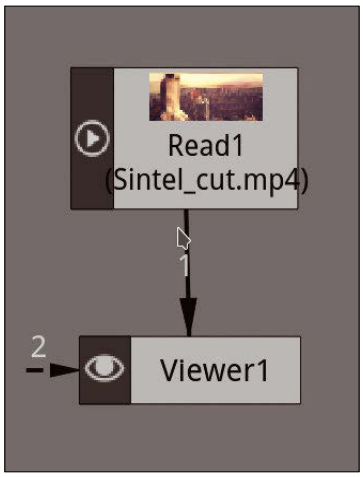


Figure 2: Your clip connected to the ***Viewer*** node.

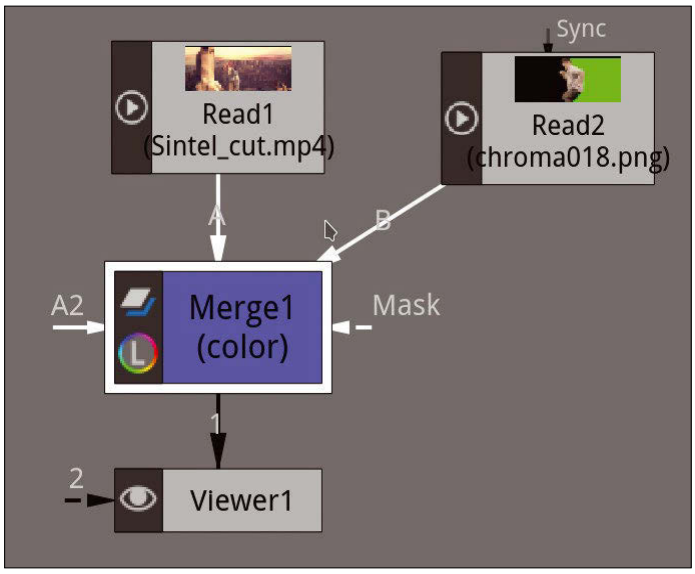


Figure 3: Merging the animation sequence onto clip 1.

To composite the two inputs, clip 1 and your animation sequence, you need a *Merge* node, so right-click in the *Node Graph* and choose *Merge* | *Merge* from the pop-up menu. Select the new *Merge* node and, in its property box (column on the left of Natron's window), change the *Operation* value to *color* (selecting this option from the drop-down list).

Connect *Merge*'s *A* arrow to your video clip and its *B* arrow to the animation sequence, as shown in Figure 3.

The preview pane will show something like what you see in Figure 4, which is a bit more than halfway through the transition.

It is time to bring in clip 2. In the same way as before, create a *Read* node and navigate to the where you have stored the second clip to which you want to transition. If Natron connects it to your tree of nodes, disconnect it and bring in a *ChromaKeyer* node by right-clicking in the *Node Graph* and choosing *Keyer* | *ChromaKeyer* from the menu.

Connect the output from your *Merge* node into the *ChromaKeyer* node and your clip 2 node to the *ChromaKeyer*'s *Bg* arrow, as shown in Figure 5.

The final step is to tell the *ChromaKeyer* node what color to turn transparent. In the node's properties box, you can either adjust the *Key color* RGB values by hand, or, if you click in the color box (by default set to black), a little color picker will appear. Move it over to the preview pane, hold down Ctrl, and click on the green section of your animation clip. Presto! Clip 2 now shines through where before was garish green (Figure 6).

Time Offsetting

Your transition is all but done, but so far you have only seen how to apply it to the beginning of the first clip. As you are engineering a transition from clip 1 to clip 2, this is not what you would usually want to do. On the contrary, what you really want to do is place the animation at the end of clip 1. You also want to push the beginning of clip 2 towards the end of clip 1, to when the transition starts.

Although this is bordering on video-editing territory (which is not Natron's purpose), in this case at

Figure 4: Animation and clip 1 merged and a bit more than halfway through the transition.

Figure 5: Connecting your second clip to finish off the transition.

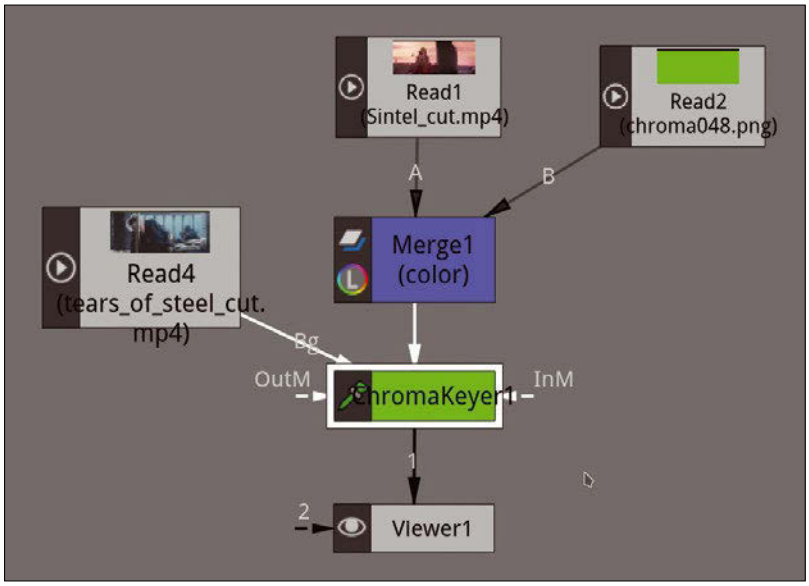


Figure 6: The finished product shows part of clip 1, the animated transition dude, and part of clip 2.

least, it is legitimate thing to do. To move the beginning of an effect or a clip along the timeline, you can use a *TimeOffset* node (right-click, and then *Time* | *TimeOffset*). Hook it up as shown in Figure 7 and, in the *TimeOffset* node's property box, change the *Time Offset (Frames)* value to, say, *100* to make the animation transition start on frame 100 of the overall project.

You can now select and right-click on the *TimeOffset* node and clone the node by choosing *Edit* | *Clone Nodes* from the pop-up menu. Use the cloned node (a pinkish red in Figure 7) to delay the start of clip 2 as well, so it coincides with the beginning of the animation. By cloning the node (instead of just creating a new one), you can guarantee that both clips will start at the same point in the project.

Although we have made sure all our clips are the same size for this experiment, you can also resize a clip on the fly with the *Resize* node (right-click, and then *Transform* | *Resize*). However, if you dragged in a large clip, the view port may still show a great big empty area (the size of the biggest clip) even after you have resized it. This is normal: Natron by default resizes your project to the size of your largest clip's frame. To fix this, you need to readjust the project's settings. Mouse over an empty space within the node working area and press the *S* key on your keyboard. The project settings will show up in the *Properties* column on the right. Change the size of the project in the *Output Format* field.

Figure 7: The ***TimeOffset*** node allows you to push effects back in the timeline.

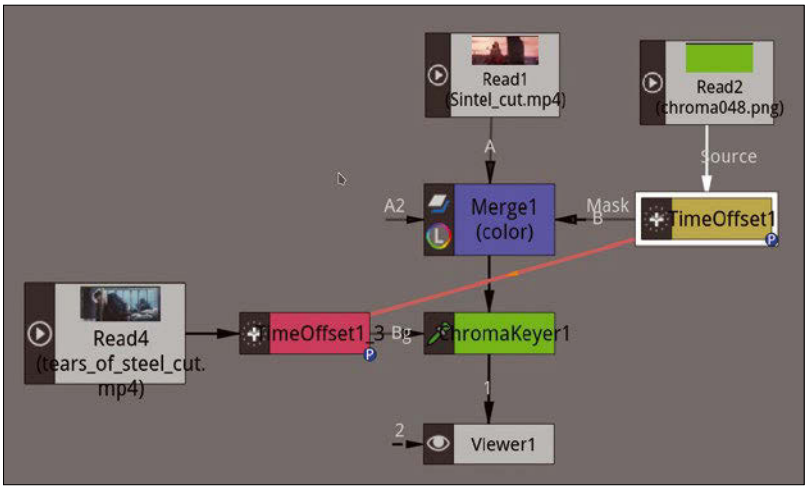
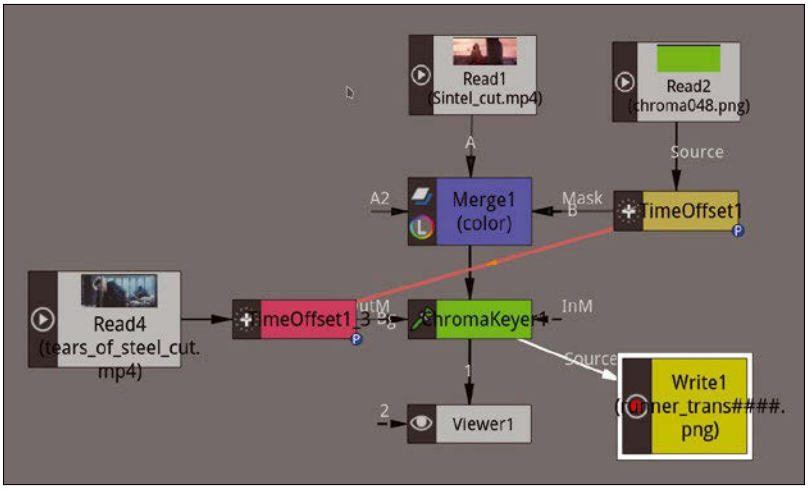


Figure 8: Use a ***Write*** node to render your clip.



Rendering

To render your project, add a *Write* node (right-click, and then *Image* | *Write*) and plug the *ChromaKeyer* node in to it, as shown in Figure 8.

Create a new directory and pick a name for your frames. Something like *transition####.jpg* will work. Thanks to the #### part of the name, Natron can generate names for your files, such as `transition0000.jpg`, `transition0001.jpg`, `transition0002.jpg`, etc.

Although most of the *Write* node's values are okay, you may want to change the *Frame Range* to *Manual* and then set the first and last frame to render. You should also change the *Input Premult* value to *Opaque*, or the chroma part of your animation sequence will show up white or black instead of transparent, depending on the output format you choose.

Tracking

Tracking is one of the more interesting and useful things Natron does quite well. A tracker automatically follows a portion of an image from frame to frame in a video clip.

Say you place a tracker on your main character's motorbike. If the tracking is done correctly, you can have Natron follow the motorbike all through a scene. You can use tracking data to stabilize a shot around the object: Although the motorbike is moving in the original clip, in the modified clip using trackers, you could make the whole environment move around the motorbike, giving your clip a dreamy, giddy feel.

How successfully Natron manages to track something will depend on the clip's quality. A high number of frames per second, a high definition, and a high contrast of the object you want to track with regard to the background is what is going to make tracking easier.

Tracking works on the principle that two consecutive frames within the same sequence (i.e., with no cuts between them) will be similar to each other. If you have someone riding the motorbike from left to right in frame 1, frame 2 will show something very similar, with that same someone, on the same motorbike, riding in the same direction. The only difference will be that the rider and their bike will have moved ever so slightly to the right of the frame. A program can track those slight changes, comparing frame 1 to frame 2, and keep a register of what has changed by inputting data into an array. If you wanted to track the position of the motorbike's back wheel, for example, Natron can create an array with the relative *x* and *y* position of said wheel throughout the whole sequence.

That, at least, is the theory. In real life, videos are wobbly, making frames blurry, and changes in light often throw the tracker … well, off track. This means the tracker needs some human assistance.

Load the clip you want to track. I used some stock footage of a motorcycle stunt [7] I found on Pixabay. Bring out a *Tracker* node (right-click, and then *Transform* | *Tracker*), and place it between your clip's *Read* node and the *Viewer* node.

Move to the first frame in your clip and double-click on the *Tracker* node to bring up its properties box to the top of the stack on the right. In your *Tracker* node's properties box, make sure you are on the *Tracking* tab and locate the largish text box at the bottom. Add a new tracker by clicking on the + at the bottom of text box (Figure 9). A new tracker will appear in the viewport showing the preview. It looks like a square bullseye and appears in the center of the viewport.

By clicking and dragging on the tracker's central point, you can drag the tracker onto the element you want to track. In the motorbike stunt video, I picked the nearest bike's back wheel. I placed the tracker's central point over the wheel's axle. You can push and pull on the tracker's handles so that it roughly covers the shape of the element you want to track (Figure 10).

With the tracker in place, theoretically you could just press the button with the right-pointing blue arrow in the upper left-hand corner of the preview pane and hope for the best. But the tracker is guaranteed to get lost. The correct thing to do is to place keyframes along the timeline so that the tracker only has to track for short stretches of frames.

The clip of the stunt is only 112 frames long, so you can place a keyframe every 10 frames. To do that, select the track you are going to work with by clicking on it in the property box (in Figure 10, the *track1* track is selected), and then make sure you are on the first frame of the segment you want to track and that the tracker is covering the element you want to track. Move to frame 10 and move the tracker by hand (click on the tracker's central point and drag it) to the new position of the element. In the case of the motorbike stunt video, you move the tracker to every new position of the motorcycle's wheel.

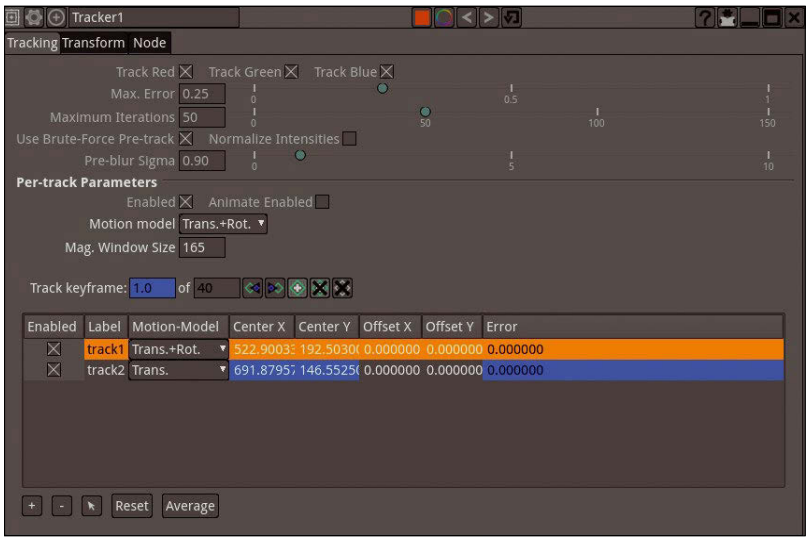


Figure 9: The ***Tracker*** node property box with two trackers.

Proceed through the clip, relocating the tracker every 10 frames. Each time you change the tracker's position, Natron will insert a new keyframe automatically into your timeline. Keyframes show up like little hourglasses in the timeline, and the *Track keyframe* value in the *Tracking* property box will have a dark blue background for keyframes and a light blue background for normal frames.

Once you have inserted all the keyframes, go back to the first frame and click on the right-pointing blue arrow button and *hopefully* the tracker will run from keyframe to keyframe, inserting all intermediate tracking data.

This is unlikely to happen, however. If the tracker gets confused and loses the thing it is tracking, it will stop on the frame that is causing problems. You then have to reposition the tracker by hand (which will insert a new keyframe) and start tracking from that point onward again.

To locate problematic stretches on a track, you may also want to toggle the *showError* button lo-

Figure 10: A tracker following a stunt bike's back wheel.

cated over the preview. This button is located third from the right on the tracker toolbar, and its icon looks like a triangle with cut off vertices. Once toggled, Natron will color each point on the track in a different color: Green means the tracker had no problem at that point of the track, Orange indicates that there was some confusion, and Red shows that the tracker is probably completely lost. It is a good idea to scroll back to the red points, readjust the tracker by hand, and start it off again from that point onward.

Stable Genius

Once Natron has tracked until the end of the clip, you can stabilize the video. In the *Tracking* properties box, click on the *Transform* tab and choose *Stabilize* from the *Motion Type* drop-down list. And that is all there is to it: When you run through the video, the object you were tracking will remain in the same position relative to the sides of the viewing port, while the rest of the frame moves around it.

Things get wild when you track two points, like both wheels on the motorbike. Then the frame doesn't only move up and down but also revolves around the tracking point (Figure 11), as one tracking point revolves around the other. The effect is quite striking, if not a little dizzying.

You can see an example of tracking and stabilization, revolving frames and all, online [8].

Next Time

Another interesting use of tracking is to use it to seamlessly insert elements into footage that were not there originally, like posters on walls or app-screen mockups onto phones. This requires careful tracking, though, and a few more techniques. Have fun! ■■■

Info

[1] "Natron Nodes" by Paul Brown, *Linux Magazine*, issue 219, February 2019, p. 90, *http://www.linux-magazine.com/Issues/2019/219/Tutorials-Natron*

[2] Natron: *https://natrongithub.github.io/*

[3] Custom animation example: *https://peertube.mastodon.host/videos/watch/58620c73-6e63-4578-8455-dfce092dc6a9*

[4] "Video Editing" by Paul Brown, *Linux Magazine*, issue 209, April 2018, *http://www.linux-magazine.com/Issues/2018/209/Gobbling-Up*

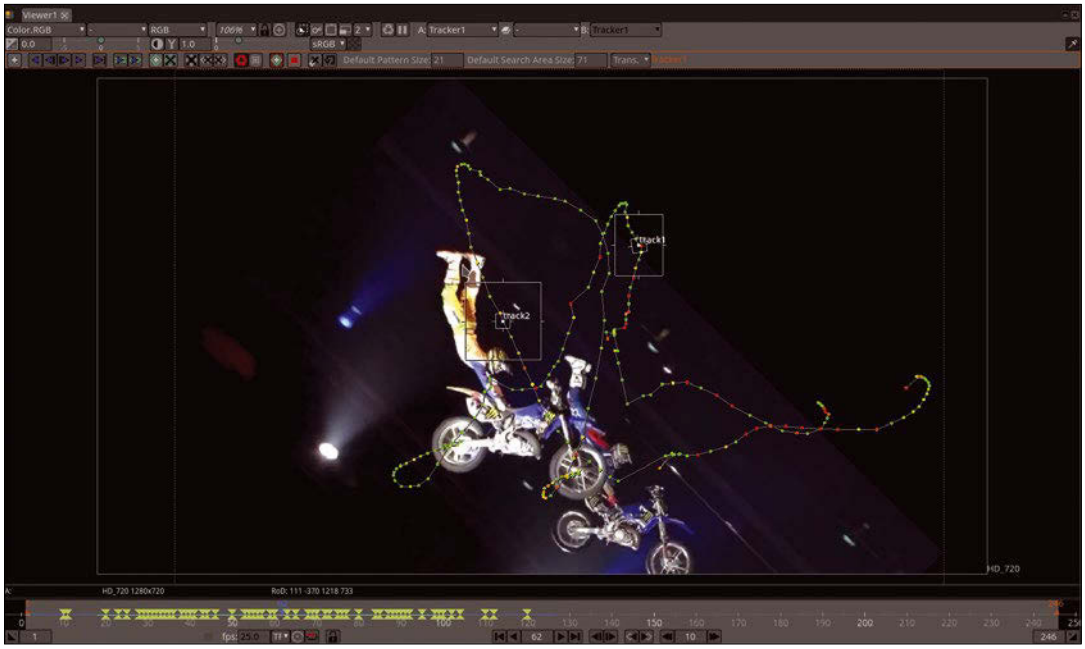
[5] "Kdenlive and ImageMagick" by Paul Brown, *Linux Magazine*, issue 213, August 2018, *http://www.linux-magazine.com/Issues/2018/213/Tutorials-Kdenlive-and-ImageMagick*

[6] customtransitions.sh: *https://gitlab.com/customcommandlinevideoediting/customtransitions/tree/master*

[7] Motorcycle stunt footage: *https://pixabay.com/en/videos/stunt-motorbikes-synchronous-1083/*

[8] Tracking two points with Natron: *https://peertube.mastodon.host/videos/watch/b0338676-f71d-41ed-90be-44ec9c68867d*

Figure 11: The stuntmen in this frame are actually flying upside down through the air, but, by tracking the wheels on one of the bikes, it seems like they are static, and it is the rest of the world that is doing somersaults.



IT Highlights at a Glance

Too busy to wade through press releases and chatty tech news sites? Let us deliver the most relevant news, technical articles, and tool tips – straight to your Inbox. Subscribe today for our excellent newsletters:

ADMIN HPC • ADMIN Update • Linux Update

and keep your finger on the pulse of the IT industry.

Admin and HPC: **www.admin-magazine.com/newsletter**

Linux Update: **www.linux-magazine.com/newsletter**

Photo by Parker Byrd on Unsplash

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 *http://linux-magazine.com/events.*

If you know of another Linux event you would like us to add to our calendar, please send a message with all the details to *events@linux-magazine.com*.

CloudFest 2019

Date: March 23-29, 2019

Location: Europa Park, Germany

Website: *https://www.cloudfest.com/*

CloudFest 2019 – everything you loved about WHD.global, only bigger, bolder, and reflecting the entire cloud ecosystem. Join more than 7,000 cloud-industry decision-makers to take over an amusement park for an immersive networking and deal-making experience. **Use code CF19LUR to register for free!**

SUSECON 2019

Date: April 1-5, 2019

Location: Nashville, Tennessee

Website: *https://www.susecon.com/*

At SUSECON 2019, you will learn the latest developments in enterprise-class Linux, OpenStack, Ceph storage, Kubernetes, Cloud Foundry, and other open source projects from technical experts, ecosystem partners, and your peers from around the world.

DrupalCon

Date: April 8-12, 2019

Location: Seattle, Washington

Website: *https://events.drupal.org/*

The Drupal community is one of the largest open source communities in the world. At DrupalCon, you'll learn to make, think about, and do things differently with Drupal. Each technical track includes sessions for beginners, experts, and everyone in between. You'll leave DrupalCon inspired and empowered to create amazing web experiences.

Events

SCaLE 17x	March 7-10, 2019	Pasadena, California	https://www.socallinuxexpo.org/scale/17x
SMART IOT 2019	March 12-13, 2019	London, England	https://www.smartiotlondon.com/
Open Source Leadership Summit	March 12-14, 2019	Half Moon Bay, California	https://events.linuxfoundation.org/events/open-source-leadership-summit-2019/
Icinga Camp Berlin	March 14, 2019	Berlin, Germany	https://www.icinga.com/events/icinga-camp-berlin/
Chemnitzer Linux-Tage	March 16-17, 2019	Chemnitz, Germany	https://chemnitzer.linux-tage.de/2019/en/
CloudFest 2019	March 23-29, 2019	Europa Park, Germany	https://www.cloudfest.com/
SREcon 19 Americas	March 25-27, 2019	Brooklyn, New York	https://www.usenix.org/conference/srecon19americas
SUSECON 2019	April 1-5, 2019	Nashville, Tennessee	https://www.susecon.com/
Cloud Foundry North America 2019	April 2-4, 2019	Philadelphia, Pennsylvania	https://www.cloudfoundry.org/event/nasummit2019/#
Open Networking Summit (ONS)	April 3-5, 2019	San Jose, California	https://events.linuxfoundation.org/events/open-networking-summit-north-america-2019/
DrupalCon	April 8-12, 2019	Seattle, Washington	https://events.drupal.org/
ASPLOS 2019	April 13-17, 2019	Providence, Rhode Island	https://asplos-conference.org/
Red Hat Summit 2019	May 7-9, 2019	Boston, Massachusetts	https://www.redhat.com/en/summit/2019
Open Source Data Center Conference (OSDC)	May 14-15, 2019	Berlin, Germany	https://osdc.de/
Secure Linux Administration Conference (SLAC)	May 27-29	Berlin, Germany	https://www.heinlein-support.de/secure-linux-administration-conference
OSCamp Ansible	May 16, 2019	Berlin, Germany	https://opensourcecamp.de/

CALL FOR PAPERS

We are always looking for good articles on Linux and the tools of the Linux environment. Although we will consider any topic, the following themes are of special interest:

- System administration
- Useful tips and tools
- Security, both news and techniques
- Product reviews, especially from real-world experience
- Community news and projects

If you have an idea, send a proposal with an outline, an estimate of the length, a description of your background, and contact information to *edit@linux-magazine.com*.

The technical level of the article should be consistent with what you normally read in *Linux Magazine*. Remember that *Linux Magazine* is read in many countries, and your article may be translated into one of our sister publications. Therefore, it is best to avoid using slang and idioms that might not be understood by all readers.

Be careful when referring to dates or events in the future. Many weeks could pass between your manuscript submission and the final copy reaching the reader's hands. When submitting proposals or manuscripts, please use a subject line in your email message that helps us identify your message as an article proposal. Screenshots and other supporting materials are always welcome.

Additional information is available at:
http://www.linux-magazine.com/contact/write_for_us.

NOW PRINTED ON recycled paper from 100% post-consumer waste; no chlorine bleach is used in the production process.

Authors

Contact Info

Editor in Chief
Joe Casad, jcasad@linux-magazine.com

Copy Editor
Amy Pettle

News Editor
Swapnil Bhartiya

Editor Emerita Nomadica
Rita L Sooby

Localization & Translation
Ian Travis

Layout
Dena Friesen, Lori White

Cover Design
Lori White

Cover Image
© balein, 123RF.com

Advertising
Brian Osborn, bosborn@linuxnewmedia.com
phone +49 89 3090 5128

Marketing Communications
Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
2721 W 6th St, Ste D
Lawrence, KS 66049 USA

Publisher
Brian Osborn

Customer Service / Subscription
For USA and Canada:
Email: cs@linuxpromagazine.com
Phone: 1-866-247-2802
(Toll Free from the US and Canada)

For all other countries:
Email: subs@linux-magazine.com

www.linuxpromagazine.com – North America
www.linux-magazine.com – Worldwide

Printed in Nuremberg, Germany by hofmann infocom GmbH on recycled paper from 100% post-consumer waste; no chlorine bleach is used in the production process.

Distributed by Seymour Distribution Ltd, United Kingdom

LINUX PRO MAGAZINE (ISSN 1752-9050) is published monthly by Linux New Media USA, LLC, 2721 W 6th St, Ste D, Lawrence, KS, 66049, USA. Periodicals Postage paid at Lawrence, KS and additional mailing offices. Ride-Along Enclosed. POSTMASTER: Please send address changes to Linux Pro Magazine, 2721 W 6th St, Ste D, Lawrence, KS 66049, USA.

Published monthly in Europe as Linux Magazine (ISSN 1471-5678) by: Sparkhaus Media GmbH, Zieblandstr. 1, 80799 Munich, Germany.

Approximate
UK / Europe Mar 02
USA / Canada Mar 29
Australia Apr 29
On Sale Date

Issue 221 / April 2019

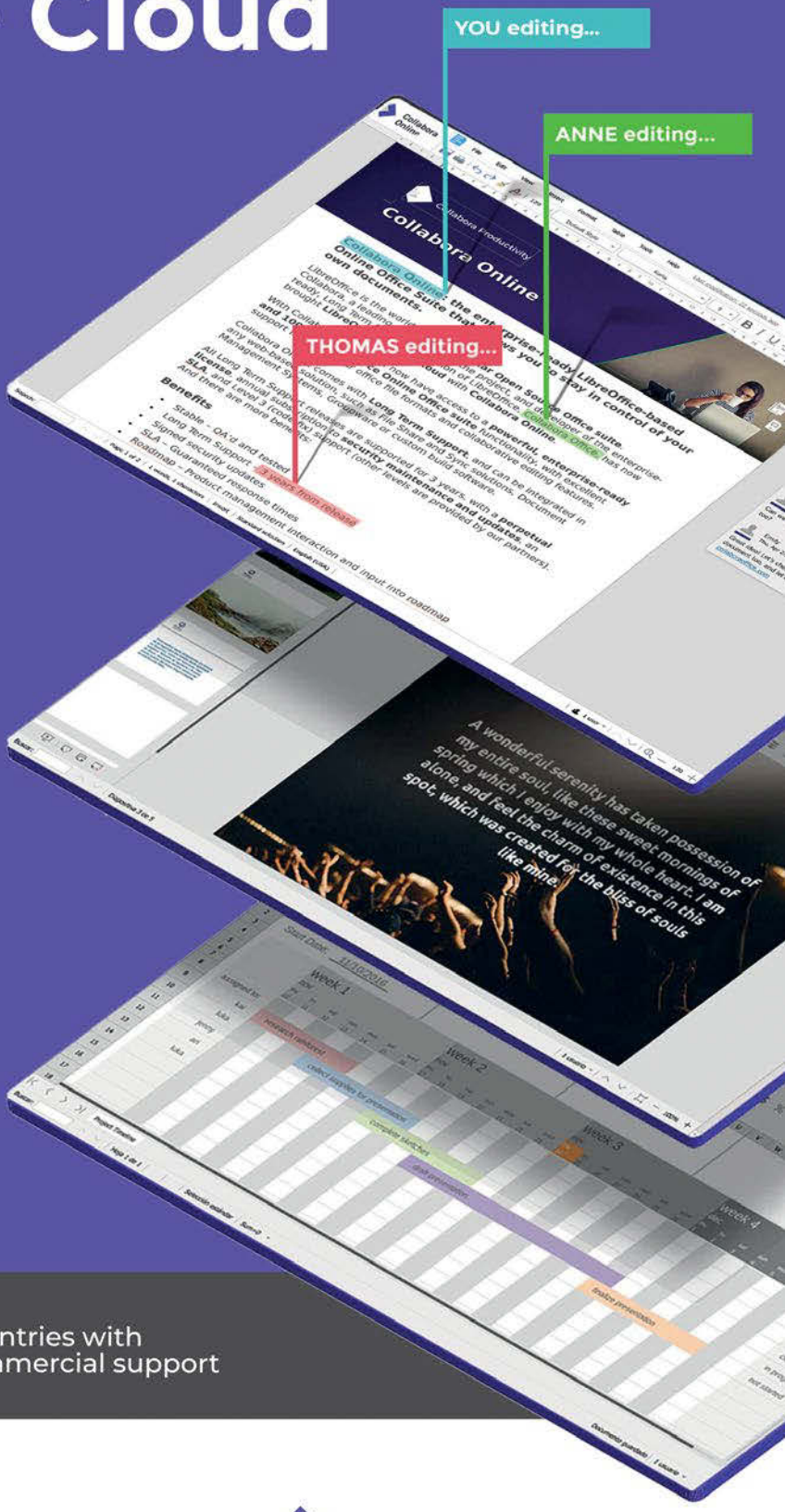
Performance Monitoring

Why spend the money to upgrade to a new system if you can still make your old system faster? Performance tuning is all about getting more from your hardware, and the first step is to watch and listen. Next month, we explore the art of performance monitoring.

The powerful on-premise office suite in the Cloud

Protects users privacy and allows them to keep full control of sensitive data

- Powerful interoperability
- Live collaborative editing
- Service Level Agreement
- On-premise privacy protection
- Collaborative roadmap planning
- Host your own on-line Office Suite



130 Partners worldwide

55 Countries with commercial support

Try it out at
collaboraoffice.com

Collabora Online

SUPERMICRO

Save Money.
Save Mother Earth.

RESOURCE-SAVING SERVERS

Build your data center with technology that is
BIG on **SAVINGS** and **LOW** on environmental **IMPACT**
with Intel® Xeon® Scalable Processors

60% Space Savings | 50% Energy Savings | e-Waste Savings

intel XEON PLATINUM inside

Learn more at
www.supermicro.com/WeKeepITGreen

SUPERMICRO