FREE DVD

RabbitMQ: Multiprotocol messaging for IoT

MONITORING PERFORMANCE

Double-Sided DVD INSIDE!

LINUX PRO MAGAZINE

APRIL 2019

MONITORING PERFORMANCE

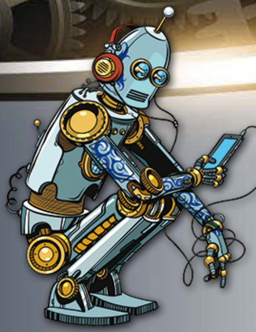
Hunting for bottlenecks that slow down your system

Silhouettes

Image processing with Go

Better Voting Machines

Is open hardware the answer?



IoT Tricks

Use an Android smartphone as a Rasp Pi display

Proton

Cross-platform gaming with Steam's Wine fork

Gnome Boxes

Easy virtualization with this handy KVM front end

LINUXVOICE

- Pixelitor: Open source image editor
- maddog takes a look back at the cost of computing
- Network configuration with the versatile ip

FOSSPicks

- MuseScore 3.0
- weborf
- MarbleMarcher

Tutorials

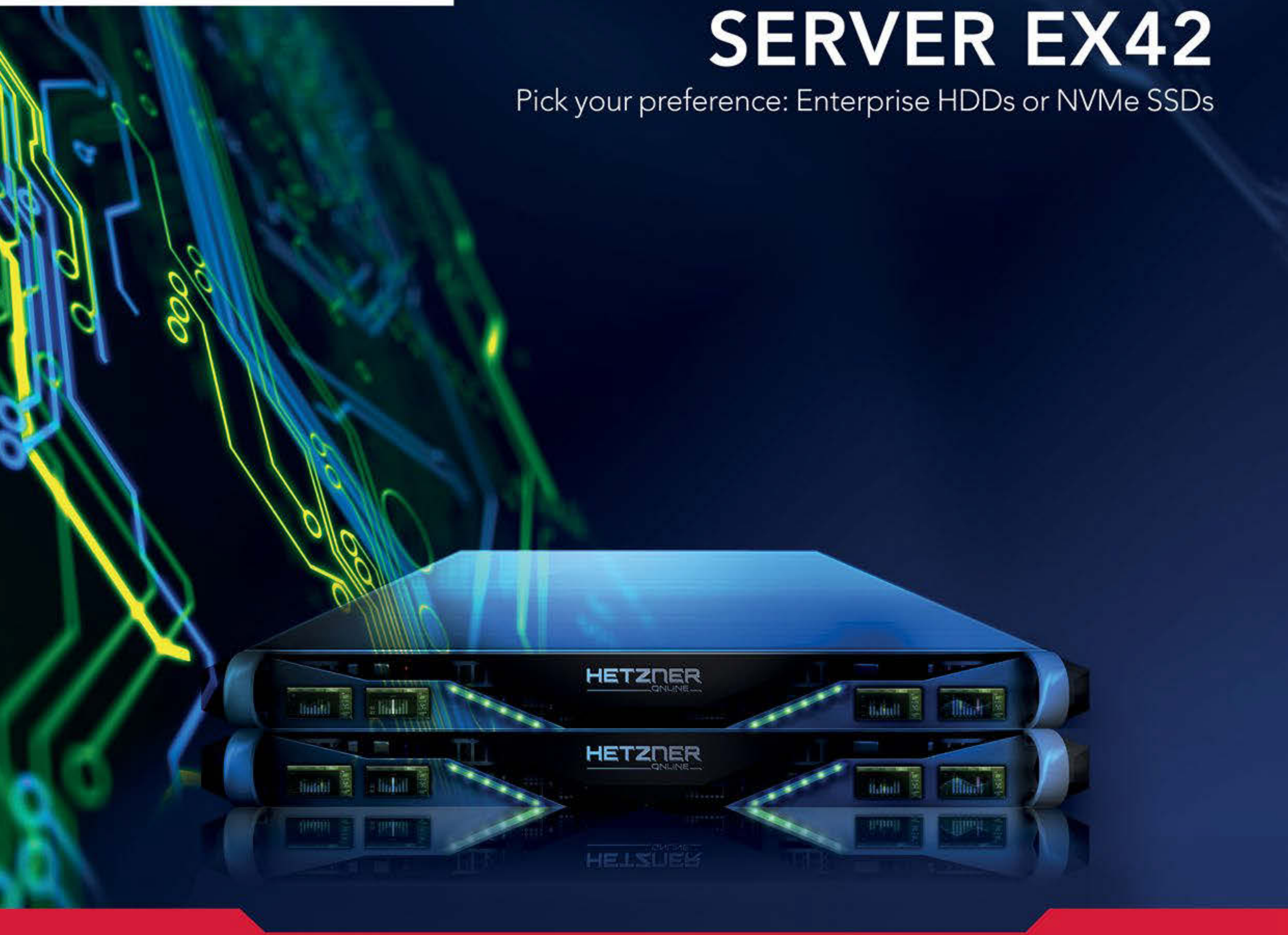
- Bash Flow Control
- Embedding Elements in Video Clips

Issue 221
Apr 2019
US$ 15.99
CAN$ 17.99

PLENTY OF POWER. SMALL PRICE.

DEDICATED ROOT SERVER EX42

Pick your preference: Enterprise HDDs or NVMe SSDs



Dedicated Root Server EX42

- ✓ Intel® Core™ i7-6700 Quad-Core incl. Hyper-Threading-Technology
- ✓ 64 GB DDR4 RAM
- ✓ 2 x 4 TB SATA Enterprise Hard Drive 7200 rpm
- ✓ 100 GB Backup Space
- ✓ Traffic unlimited
- ✓ Location Germany or Finland
- ✓ No minimum contract
- ✓ Setup Fee $44.50

monthly from $**39**

Dedicated Root Server EX42-NVMe

- ✓ Intel® Core™ i7-6700 Quad-Core incl. Hyper-Threading-Technology
- ✓ 64 GB DDR4 RAM
- ✓ 2 x 512 GB NVMe SSD
- ✓ 100 GB Backup Space
- ✓ Traffic unlimited
- ✓ Location Germany or Finland
- ✓ No minimum contract
- ✓ Setup Fee $44.50

monthly from $**39**

All prices exclude VAT and are subject to the terms and conditions of Hetzner Online GmbH. Prices are subject to change. All rights reserved by the respective manufacturers.

www.hetzner.com

BRICKS AND SPRAY PAINT

Dear Reader,

As this issue goes to print, news is circulating about a catastrophic hack on the mail provider VFEmail. According to reports, two decades of saved data for all US users is lost – totally wiped out. Email providers are accustomed to getting attacked, and most of the attacks are stopped at the front door. Attackers sometimes get through, in which case, the most common scenario is that they encrypt some data and ask for a ransom. In this case, however, the attacker didn't seem to really want anything, other than a chance to go on a rampage and destroy all the data.

No attempt was made to deliver ransom demands. The crime did not look like extortion or theft but resembled something more like ordinary vandalism. The attacker careened around the network, reformatting disks and destroying data. Mail servers, file servers, VM servers, database servers, and even backup servers were lost. Although vandalism tends to appear random, this attack seems to have been carefully planned. According to reports, the attacker needed multiple passwords to access all these servers and therefore must have been lurking and listening on the network for some time to acquire the necessary access information.

I won't solve the mystery in the time it takes to write this column. Too much is unknown at this time. Was the attack from a disturbed loner who just wanted to destroy something? Was it a disgruntled customer or a former employee out for revenge? Was it an inside job? Another possible scenario is that the attacker was a customer with a secret who decided to destroy the evidence by destroying *every* account, rather than just deleting personal emails and risking leaving a trail.

The VFEmail attack caught the imagination of the high tech press because it was just so weird. Nefarious as ransomware attacks might be, we are at least able to classify them as being somehow related to the quest for money (which we all secretly understand). A wanton attack of vengeance or vandalism scares us the way we are scared by a tornado or a madman with a knife. This attack underscores the dark reality that the Internet really is an unsafe place. Criminals and sociopaths from all over the world can ride a magic carpet to your front door, and the onus is on you to find the right kind of lock – and to continually change the lock as new techniques render old locks ineffective. It is actually profoundly strange that our whole economy and trillions of dollars in business interests are based on this model.

Still, VFEmail deserves some heat for the failure of their disaster recovery plan. If you read down through the comments under the news stories on the attack, you'll find lots of notes from sys admins who are unimpressed that such a thing could happen. Without the details, it is difficult to see exactly what went wrong. At least so far, there doesn't seem to be an obvious gotcha-type mistake, such as an unpatched server or sloppy password policy. Two issues *are* clear at this time and should serve as a cautionary tale for other admins as they prepare for what we hope *will not* be a new era of stone-age-style, destroy everything attacks:

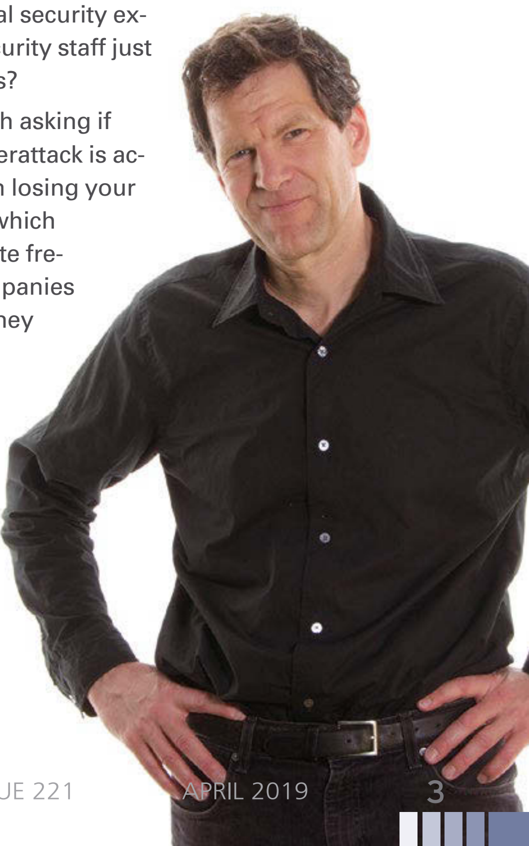
- Although the company did provide regular backups to backup servers located on the network, their backup process apparently did not include an offline storage component, which is often (though not universally) recommended by security experts.
- Unless the attack was an inside job, the intruder spent some time hanging out on the network snooping passwords. (Note that this is not a disaster recovery problem but is more of an intrusion prevention problem.)

Another question that no one seems to be asking is to what degree this story reflects a growing trend in our IT industry, which favors huge providers over small to mid-size businesses. Based on comments and responses that have appeared in the press, it appears to be a fairly small-time operation. Fifteen years ago, there were thousands of small businesses operated by the owner and a small team providing services on the Internet. Are we now approaching a world in which every company needs to be big enough to employ a professional security expert and full-time security staff just to watch for intruders?

Then again, it is worth asking if losing email to a cyberattack is actually any worse than losing your credit card number, which seems to happen quite frequently with big companies who have lots of money for security experts.

Joe

Joe Casad,
Editor in Chief



LINUX MAGAZINE

WHAT'S INSIDE

As computers get more powerful and computer applications get bigger and more ungainly, users end up with the same question no matter how many times they upgrade: How do I make the most of my system resources by tuning up system performance? This month we explore some tools that offer a window into the performance and health of your Linux system: Netdata and perf.

Other highlights in this issue include:

- **Steam Proton** – this Wine fork running within the Steam Play gaming system provides a seamless way to run Windows games on Linux (page 24).
- **Gnome Boxes** – an easy interface for configuring and managing KVM virtualization (page 32).

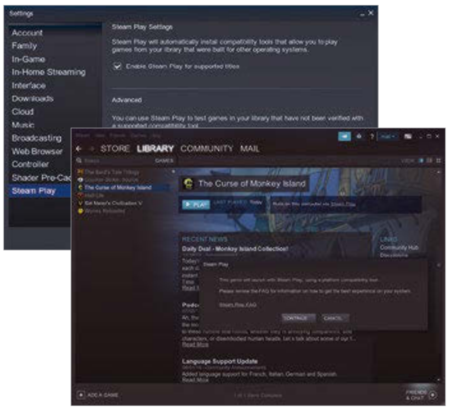
Check out LinuxVoice for a look at the Pixelitor graphics editor and another installment in our series on Bash scripting – this time with the emphasis on flow control.

SERVICE

NEWS

COVER STORIES

REVIEWS



Steam Play Settings

Steam Play will automatically install compatibility tools that allow you to play games from your library that were built for other operating systems.

☑ Enable Steam Play for supported titles

Advanced

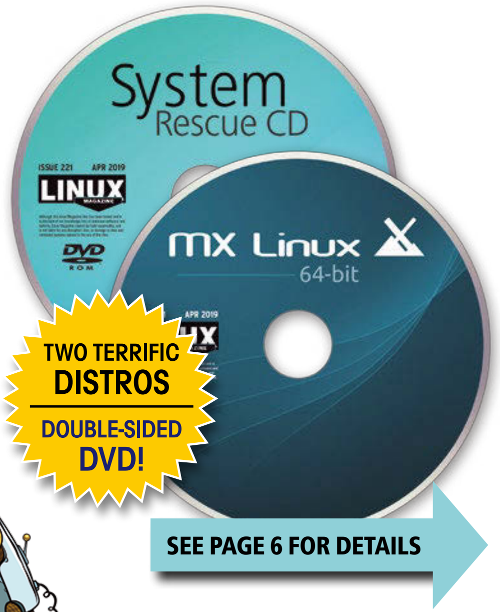
You can use Steam Play to test games in your library that have not been verified with a supported compatibility tool.

RECENT NEWS

Daily Deal - Monkey Island Collection!

Language Support Update

Added language support for French, Italian, German and Spanish.


System Rescue CD
ISSUE 221 APR 2019
LINUX
DVD ROM
MX Linux
64-bit
TWO TERRIFIC DISTROS
DOUBLE-SIDED DVD!
SEE PAGE 6 FOR DETAILS

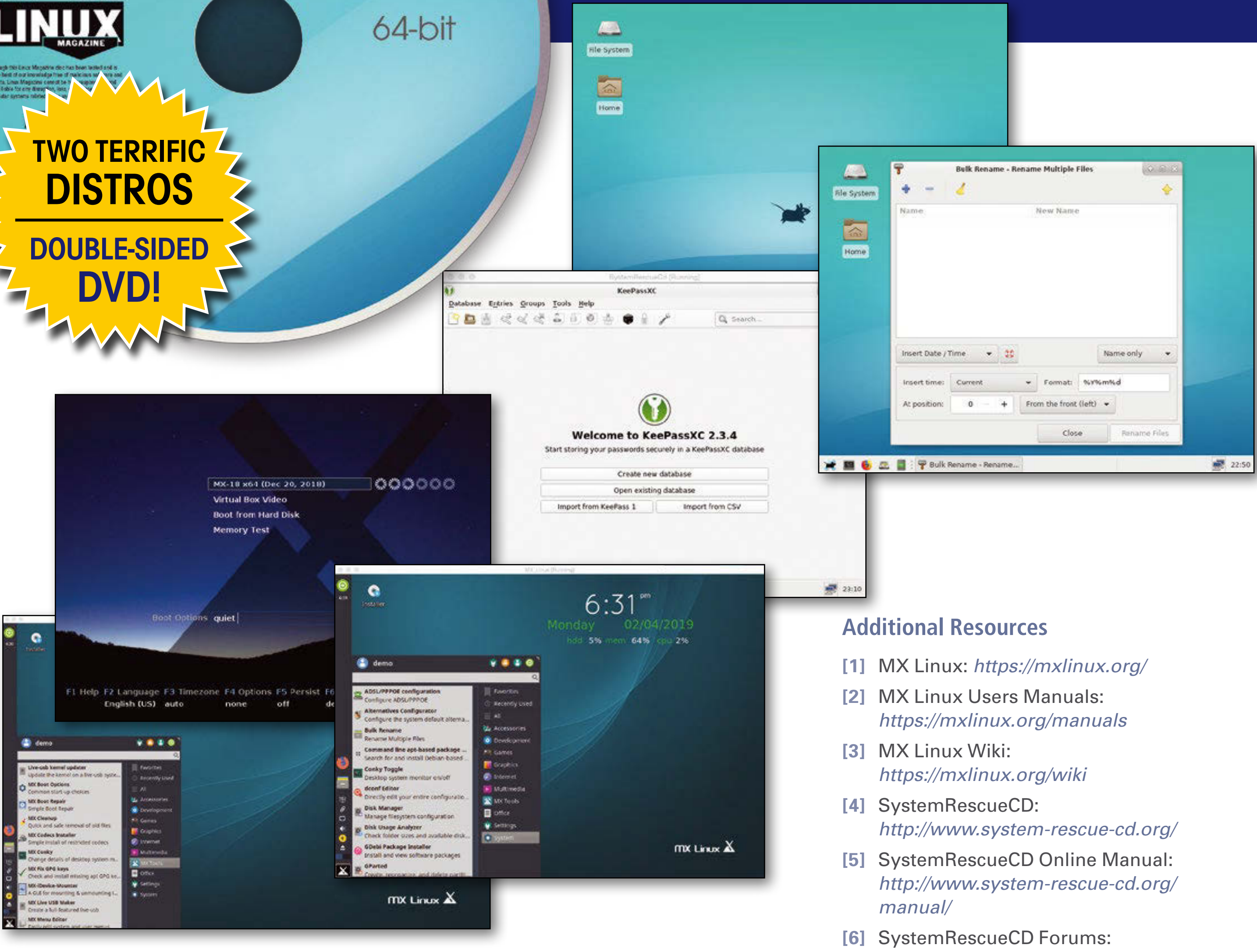
On the DVD

MX Linux

The MX Linux website describes this elegant distro as "a cooperative venture between the antiX and former MEPIS communities, using the best tools and talents from each distro." MX Linux is based on Debian and comes with the Xfce4 desktop. The latest edition includes updated firmware and Meltdown/Spectre mitigation.

SystemRescueCD

This system on a disc is designed for resurrecting downed Linux and Windows computers. Onboard you'll find a gallery of diagnostic and troubleshooting tools, along with filesystem utilities and other rescue aids.



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

Additional Resources

[1] MX Linux: *https://mxlinux.org/*

[2] MX Linux Users Manuals: *https://mxlinux.org/manuals*

[3] MX Linux Wiki: *https://mxlinux.org/wiki*

[4] SystemRescueCD: *http://www.system-rescue-cd.org/*

[5] SystemRescueCD Online Manual: *http://www.system-rescue-cd.org/manual/*

[6] SystemRescueCD Forums: *http://forums.system-rescue-cd.org/*

Nuremberg, Germany

openSUSE®

Conference

Open Source Software

♦ Embedded Systems

♦ Cloud ♦ Containers ♦ More

May 24 - 26, 2019

events.opensuse.org

NEWS

Updates on tech s

THIS MONTH'S NEWS

Linux Kernel Continues to Offer Mitigation for Spectre Mitigation

Usually, you want to mitigate all possible vulnerabilities unless you are talking about Meltdown and Spectre which are a class or family of dozens of vulnerabilities. But what sys admins hate more than these vulnerabilities are mitigations offered to these vulnerabilities (*https://www.zdnet.com/article/linux-kernel-gets-another-option-to-disable-spectre-mitigations/*). Some of these mitigations have a massive impact on performance, while not offering any significant protection.

Gauging the pros and cons, sys admins have gone as far as asking the Linux kernel community to give them an option to disable these mitigations (*https://www.phoronix.com/scan.php?page=news_item&px=Global-Switch-Skip-Spectre-Melt*). The Linux kernel community always listens.

Linux Kernel 4.15 added the ability for sys admins to disable the kernel's built-in mitigations for the Spectre v2 vulnerability, then Linux Kernel 4.17 offered the option to disable all mitigations for Spectre v4, and now Linux Kernel 4.19 allows admins to disable mitigations for Spectre v1.

You may or may not trust the NSA, but they have a very decent guide on GitHub (*https://github.com/nsacyber/Hardware-and-Firmware-Security-Guidance*) to help keep up with all Spectre-related vulnerabilities.

SpeakUp Trojan Targets Linux Servers

Researchers at Check Point have found a new Trojan called SpeakUp that's infecting Linux servers. SpeakUp exploits known vulnerabilities in Linux and is targeting servers in China.

According to Check Point, "SpeakUp acts to propagate internally within the infected subnet, and beyond to new IP ranges, exploiting remote code execution vulnerabilities. In addition, SpeakUp presented the ability to infect Mac devices with the undetected backdoor" (*https://research.checkpoint.com/speakup-a-new-undetected-backdoor-linux-trojan/*).

The Trojan has spread beyond China and is fast spreading across East Asia and Latin America. It's not sparing even AWS-hosted Linux servers. Check Point said six Linux distributions and macOS are vulnerable, but they didn't name exactly which six Linux distributions.

SpeakUp's initial infection vector targets a known vulnerability in ThinkPHP and then uses command injection techniques for uploading a PHP shell that serves and executes a Perl backdoor. After executing the script to install the backdoor, it deletes the file to remove any evidence.

Check Point warns that while the initial payload of SpeakUp is mining, it poses a much bigger threat. "The threat actor behind this campaign can at any given time deploy additional payloads, potentially more intrusive and offensive. It has the ability to scan the surrounding network of an infected server and distribute the malware."

KDE Plasma 5.15 Beta Arrives

The KDE community has announced its first release of 2019 – Plasma 5.15 Beta. One of the major highlights of the beta release is increased focus on usability and productivity.

The KDE community has teamed up with the Visual Design Group (VDG) contributors to get feedback on all the paper cut bugs in the software to create an intuitive and consistent workflow for users.

The release has enhanced integration with third-party technologies like GTK and Firefox, so users can choose the apps they like without worrying about a subpar experience.

Massive improvements have been made to the Discover software management tool, something similar to App Store. Users can now perform the distribution upgrade within Discover. It also offers fine-grained control over which packages users want to update.

Discover also now supports app extensions offered with Flatpak packages and lets you choose which ones to install.

If you are a KDE Plasma user, you can test the beta and provide the community with feedback. Some of the best beta distributions to try are openSUSE Tumbleweed, Arch Linux, and KDE neon.

Canonical Announces Latest Ubuntu Core for IoT

Canonical has announced Ubuntu Core 18, their open source platform for IoT devices. Ubuntu Core 18 is based on the Ubuntu 18.04 LTS codebase and will be supported for 10 years.

At 260MB, Ubuntu Core is one of the smallest IoT platforms. They achieved this size by stripping unnecessary components from the core. However, the overall size of the OS will grow depending on the IoT device.

Reduction in size also improves security. "The attack surface of Ubuntu Core has been minimized, with very few packages installed in the base OS, reducing the size and frequency of security updates and providing more storage for applications and data," Canonical said in a blog post.

Thanks to the popularity of Ubuntu, Canonical's IoT platform has a wider range of applications at its disposal. "Ubuntu Core enables a new class of app-centric things, which can inherit apps from the broader Ubuntu and Snapcraft ecosystems or build unique and exclusive applications that are specific to a brand or model,"continued the post.

Smaller size, combined with a refined app delivery mechanism (Snap), enable Canonical to enhance its security further.

"All snaps distributed to devices are scanned regularly for known weaknesses and devices, enabling enterprises and manufacturers to learn quickly about potential risks in their ecosystem," Canonical said.

Vulnerabilities Found in Cisco Routers

The German security firm, RedTeam Pentesting has found two vulnerabilities in Cisco routers.

The vulnerabilities are found in the web-based management interface of Cisco Small Business RV320 and RV325 Dual Gigabit WAN VPN Routers.

The flaw allows an authenticated, remote attacker with administrative privileges on an affected device to execute arbitrary commands.

MORE ONLINE

Linux Magazine
www.linux-magazine.com

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

Scanning with Zenmap • Ken Hess
Discover your network with the user-friendly Zenmap network scanner.

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

OpenACC Directives for Data Movement
Jeff Layton
The OpenACC data directive and its associated clauses allow you to control the movement of data between the host CPU and the accelerator GPU.

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

Assess USB Performance While Exploring Storage Caching • Federico Lucifredi
Use dd fluently to compare disk performance.

Workflow-Based Data Analysis with KNIME
Martin Horn
They say data is "the new oil," but all that data you collect is only valuable if it leads to new insights. An open source analysis tool called KNIME lets you analyze data through graphical workflows – without the need for programming or complex spreadsheet manipulation.

Debugging Bash Scripts Automatically
Tam Hanna
We look at various extension frameworks that make the life of developers and administrators easier when debugging a script.

According to Cisco, the cause of the vulnerability is due to improper validation of user-supplied input. "An attacker could exploit this vulnerability by sending malicious HTTP POST requests to the web-based management interface of an affected device. A successful exploit could allow the attacker to execute arbitrary commands on the underlying Linux shell as root," the company said in an advisory.

Cisco has already released a patch to fix these vulnerabilities. If you use either of these two routers, you need to update now.

Two New Malware Campaigns Found

Security researchers at Carbon Black have found two malware campaigns related to the Ursnif malware.

"This attack originally came in via phishing emails that contained an attached Word document with embedded macros; Carbon Black located roughly 180 variants in the wild. The macro would call an encoded PowerShell script and then use a series of techniques to download and execute both a Ursnif and GandCrab variant," wrote Carbon Black in a blog report. Carbon Black has released a detailed overview of the campaigns.

Researchers at Talos have released a list of indicators of compromise (IOCs) to help users detect and mitigate the spread of the malware.

Image © wamsler, 123RF.com

US Government Shutdown Ties Up $139.2 Million in Grant Funding

The American Society of Biochemistry and Micro Biology (ASBMB) has posted a study of the effects of the US government shutdown on the National Science Foundation (NSF) grant process. According to the tally, which is available at the ASBMB website, the zero grants totaling $0.0 during the shutdown period between December 22, 2018 and January 25, 2019 contrasts with 465 grants totaling $139.2 Million awarded during a similar period a year ago.

The ASBMB page does not discuss the effects of the funding delay. The money has already been allocated by the Congress, so grant agencies could theoretically "catch up" by increasing their rate of funding for the next few months. However, even if the total funding for the year is equal, the delay could still disrupt the continuity of ongoing projects, especially at universities and other institutions tied to the academic calendar, as administrators assess their space and funding needs and postdocs scramble to lock in employment for the next academic year.

A notice at the NSF site entitled "Resumption of Operations at the National Science Foundation" offers thanks to the scientific community for "forbearance during this challenging time." NSF Director France A. Córdova vows to begin "… processing the backlog of awards to universities and small businesses, rescheduling merit review panels that were cancelled, funding facilities and renewing oversight of those facilities, and funding graduate student and postdoctoral fellowships."

Photo by Thought Catalog on Unsplash

The other question is whether this shutdown era is actually over yet. The continuing resolution that reopened the government expires on February 15, which will result in another closure unless a solution is in place.

Zack's Kernel News

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

Considering Plan 9 Extensions for Type Conversion

For a long time, the Linux kernel would only compile with the GNU C Compiler (GCC). Now, several compilers can do it, but each compiler has its own way of doing things, offering various extensions to the C language and optimizing code in different ways. The question of which compiler features to depend on can have an effect on whether other compilers can keep supporting Linux.

Recently, Matthew Wilcox suggested using the `-fplan9-extensions` GCC option to handle some implicit type conversions. This way, a particular cyclic memory allocation could be made to embed a needed reference instead of requiring it to be passed explicitly to the function. If the code used the Plan 9 extensions, the functions would not need to be tweaked to accept the additional input.

However, as Nick Desaulniers pointed out, other compilers might not support that particular extension. Even if Matthew successfully argued in favor of it, Nick suggested making it optional, so that other compilers could continue to support the kernel.

Linus Torvalds had even stronger reservations. He said:

The full Plan 9 extensions are nasty and make it much too easy to write "convenient" code that is really hard to read as an outsider because of how the types are silently converted. And I think what you want is explicitly that silent conversion. So no. Don't do it. Use a macro or inline function that makes the conversion explicit so that it's shown when grepping.

Linus also added, "We've used various GCC extensions since day #1 ('inline' being perhaps the biggest one that took *forever* to become standard C), but these things need to have very strong arguments."

So that was that. The kernel will not use the Plan 9 extensions. On the other hand, Matthew probably won't suffer too much implementing his cyclic memory allocations, because modifying the necessary functions will only be a slight inconvenience. The interesting question is when Linus would say it's legit to use a particular extension. Judging from this conversation, it seems like the main justification for an extension is if the feature truly belongs in the standard C language. I'd love to see that sort of debate play out at some point.

Supporting Heterogeneous Systems

As hardware systems become more complex, operating systems need to be able

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**.

to accommodate a much wider variety of configurations. It used to be that system memory was all the same type. Then we encountered systems like cell phones, which were built fast, and slow RAM. Now a system can host a wide range of different types of memory, and many different buses for sending data between them.

Among the many people working on supporting all this, Jérôme Glisse wanted to take the bull by the horns and come up with an overarching memory and bus management system to enable the kernel to operate at maximum efficiency on any hardware configuration.

One of his main goals was first to find a way even to express the configuration of a given system. Only once it could be referred to within the kernel could the strange mix of different RAMs and different bus topologies begin to be supported in a methodical way. Even the CPU had to come into play, given that now graphics processing units (GPUs) and field-programmable gate arrays (FPGAs) were taking on more and more computational tasks.

To express all these things, he named "targets," which were any memory on the system; "initiators," which were CPUs; "links," which were fast connections between targets and initiators; and "bridges," which were remote connections to other groups of initiators and targets.

With these identifiers, the entire graph of a given system could be exposed to user space via the sysfs directory. This was Jérôme's first goal.

But his patches went beyond identifying the hardware graph to expressing actual memory policy for the running system. This would be used to decide which parts of memory should host which processes. In general, one would presumably always prefer to use fast memory when it's available, and slower memory when no fast memory is free.

One of the major problems that Jérôme encountered was that the kernel's existing memory policy is handled on a per-CPU basis. He felt this was not fine-grained enough to deal with modern systems properly. A really robust set of memory policies, he felt, would require an entirely new API.

But this was easier said than done. Revising the existing memory policy API would necessarily break vast tracts of existing user code, all of which would need to be ported to the new API. That type of change, unless absolutely necessary, was unlikely to find sympathetic ears among the top kernel people.

Jérôme's solution was to write an entirely new API that could sit alongside the original. Any user code could use the new hotness, while existing code could stick with the old reliable code.

Yet this solution also carried its own drawback. As Aneesh Kumar put it in response to Jérôme's proposal, "we now have multiple entities tracking CPU and memory."

Aneesh also drew out of Jérôme the confession that even once the new API was in place, it might not result in granting all initiators optimal access to all targets. Mostly Jérôme seemed to feel that the situation was still too unknown to have real clarity. He wanted first of all to expose the topology to user space and then, once that was accomplished, see what could be seen. He wanted to climb the mountain first and only then look out across the landscape.

Dave Hansen had the sternest objections to Jérôme's work. He said that since the kernel already had infrastructure to deal with various types of RAM on the system, it made more sense to enhance those existing features rather than write something new. In particular, he said, the Heterogeneous Memory Attribute Table (HMAT) was already present in firmware for this very purpose – to express the system's hardware topology to the operating system. He also pointed out that non-uniform memory access (NUMA) existed in the kernel already to deal with multiple types of RAM. In fact, he said, NUMA had already been embraced by Advanced Configuration and Power Interface (ACPI) specification. So, Dave said, there was quite a lot of work underway to address this whole issue.

However, Jérôme was very conciliatory. He said his code was not intended to replace any of the stuff Dave had mentioned. His own work was essentially separate. He said he couldn't see any way to extend NUMA to support device memory, because that memory was not cache coherent – it didn't have the same characteristics as other memory on the system. In some cases, the memory couldn't be seen at all by the CPU. NUMA, he felt, just wasn't able to handle that sort of case. Jérôme intended his own interfaces to take up some of that slack.

And Dave essentially agreed with all that and affirmed that NUMA really was intended to handle memory that was visible universally and could be allocated normally.

So the two agreed that there didn't seem to be any conflict there. But Dave still had some technical objections, or issues, to address. For one thing, exposing the system topology on sysfs could potentially lead to a metric ton of files on systems with large numbers of CPUs and RAM resources, each with their own links and bridges to the others.

Additionally there was the question of time. Even given the non-overlapping nature of their work, it was possible that enhancing NUMA would still give faster solutions than writing an entirely new system from the ground up. Maybe it would still be better to focus on NUMA, rather than wait potentially years for Jérôme's approach to bear fruit.

These issues and others formed the rest of the technical discussion, with various other folks pitching in with suggestions. It seems that for the moment, at least, Jérôme's idea has made it past the breakers.

The issue of trying to support a widening array of increasingly complex systems is an interesting one. It's possible that in the not-so-distant future, Linux might migrate processes over WiFi, sharing the resources of every device in the house or even a whole city. It's easy to see the value in managing a vast range of unequal resources.

Optimizing CPU Idle States

Rafael J. Wysocki wanted to improve the kernel's menu governor, which is used to put a CPU into a power-saving state when the CPU is inactive. A number of such power-saving states could be available from which to choose: some that save less power but can be awakened quickly, and others that save more power but take more time to awaken. If there's good reason to know how long the CPU will remain idle, the menu governor puts the CPU into the appropriate state directly. There's also something called the ladder governor, which walks a CPU into deeper and deeper power-saving states on the basis of simple heuristics, like how long the CPU has already been in its current state.

Rafael felt that the existing logic used by the menu governor to pick CPU states was a horrifying violation of all natural law, which could only be fixed by a thorough rewrite. He offered a lot of reasons. For one thing, the menu governor used pattern matching to identify timer data, but also data from other sources, and mixed them together. This, he said, could cause the menu governor to perceive a time-based wake-up call at a point where no time-based wake-up calls were possible in the code.

The current menu governor, he said, also relied on data about processes that might not be running on the target CPU at all, so it couldn't be relevant to when to wake up the target CPU. Rafael also pointed out that some of the menu governor's heuristics had to do with whether a process was waiting for input, which he said was not actually related to the problem and seemed just random. And lastly, sometimes he found that the menu governor would start analyzing time frames that were just way too large and therefore completely irrelevant and a waste of resources even to run that code at all.

However, Rafael did acknowledge that a wholesale replacement, while good overall, might make some workloads perform worse. Specifically, any workload that had been highly tuned to work well with the current menu governor might not work so well with the replacement.

And, since those highly tuned workloads were most likely to be the ones that needed peak performance, Rafael suggested keeping both menu governors, at least for awhile, and let people choose their favorite.

He called his new one the timer events-oriented (TEO) governor. Like the menu governor, it always attempts to find the deepest (most power-saving) state in which to put the CPU, but it had a cleaner new strategy for identifying that state. He explained:

First, it doesn't use "correction factors" for the time till the closest timer, but instead it tries to correlate the measured idle duration values with the available idle states and use that information to pick up the idle state that is most likely to "match" the upcoming CPU idle interval.

Second, it doesn't take the number of "I/O waiters" into account at all, and the pattern detection code in it avoids taking timer wake-ups into account. It also only uses idle duration values less than the current time till the closest timer (with the tick excluded) for that purpose.

Doug Smythies replied with some test results comparing several different workloads under the plain kernel vs. Rafael's patched kernel. In some cases he found no significant performance difference between them, and in some cases, he found a 1.4 percent speed-up under Rafael's patch. Doug also looked over Rafael's code and posted some bug fixes, which Rafael accepted for the next iteration.

Giovanni Gherdovich also replied with his own benchmarks, saying that Rafael's patch was much better than an earlier version he'd tested and also better than the current menu governor. He also posted a set of tests he performed, which came back with no significant difference between the two governors.

Rafael looked over Giovanni's results and felt that actually the tremendous speed improvements might mean that the patch was being too aggressive. He remarked that other tests even showed a slowdown in some cases. He said he would soon put out a new patch that was a little more energy efficient, but he said that if this resulted in too much speed degradation, he'd return to the current version of his patch.

There was a bit more back-and-forth, between test results and patch tweaks, before the thread ended. There seems to be no controversy whatsoever with this patch, and inclusion in the main tree may just be a question of a few more tweaks to the code. It does seem as though there is enough variation between Rafael's TEO governor and the menu governor to warrant keeping the menu governor around for a while longer. But eventually I'd expect most user code that's optimized for the menu governor to start optimizing for the better organized and more usable TEO governor instead. ■■■

18 YEARS OF LINUX! Issues 1-214
Searchable Linux Archive: 19,000 Pages of Practical Know-How
All the Tricks
All the Hacks
All the Apps
ISSUE 215
OCT 2018
LINUX MAGAZINE
THE COMPLETE LINUX MAGAZINE ARCHIVE
DVD ROM

Real-time performance monitoring with Netdata

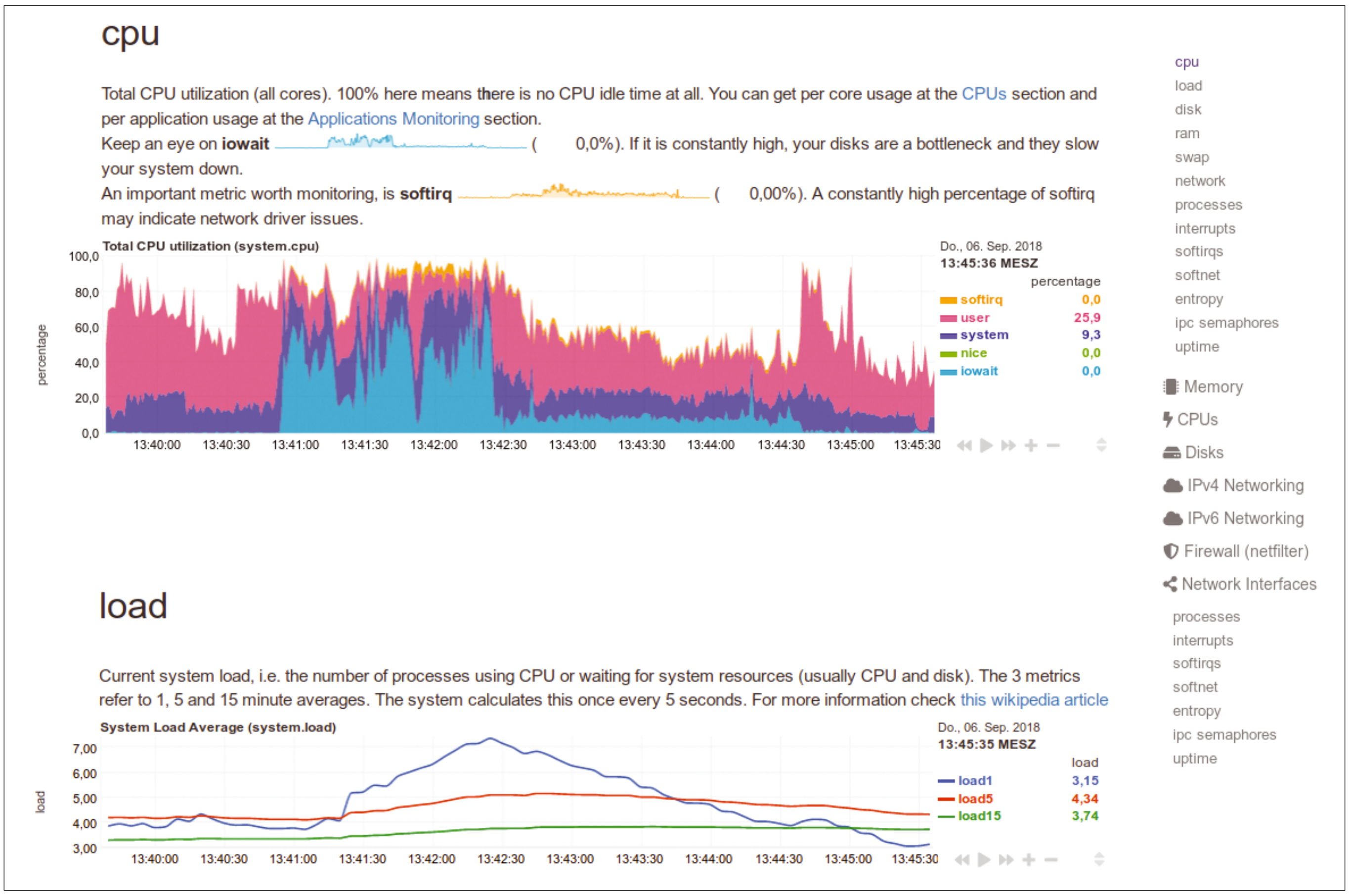
Multimeter

What cannot be measured cannot be improved. Netdata lets you measure almost anything – at least as long as it's about the performance and health of a Linux computer.

By Jens-Christoph Brendel

Netdata is a real-time monitoring tool for Linux systems. You can't use Netdata to log a long history of monitored data, but if you're looking for a tool that will let you explore a snapshot of the system state from thousands of different angles, Netdata is a powerful alternative.

You can't ask Netdata for the values for yesterday or even the last hour. In fact, Netdata usually only shows you the last five minutes. However, it displays all measured values from a fast round robin memory in RAM with a resolution of one second. Netdata strives to draw as complete a picture as possible of the performance and health of a computer at the current point in time with minimal expense of computing power and I/O.


cpu

Total CPU utilization (all cores). 100% here means there is no CPU idle time at all. You can get per core usage at the CPUs section and per application usage at the Applications Monitoring section.
Keep an eye on **iowait** (0,0%). If it is constantly high, your disks are a bottleneck and they slow your system down.
An important metric worth monitoring, is **softirq** (0,00%). A constantly high percentage of softirq may indicate network driver issues.

Total CPU utilization (system.cpu)

Do., 06. Sep. 2018
13:45:36 MESZ

	percentage
softirq	0,0
user	25,9
system	9,3
nice	0,0
iowait	0,0

load

Current system load, i.e. the number of processes using CPU or waiting for system resources (usually CPU and disk). The 3 metrics refer to 1, 5 and 15 minute averages. The system calculates this once every 5 seconds. For more information check this wikipedia article

System Load Average (system.load)

Do., 06. Sep. 2018
13:45:35 MESZ

	load
load1	3,15
load5	4,34
load15	3,74

Figure 1: Any browser displays the Netdata dashboard, which visualizes many measured values.

you can download the source code from GitHub [1]; an installer script included with the GitHub files makes the installation easy.

Netdata runs as a systemd-managed daemon on the monitored system. By default, you can reach the Netdata browser GUI via *http://localhost:1999*. An SSH tunnel is useful if you want to access Netdata from another computer. Alternatively, you can run Netdata behind a proxy [2]. You can view the dashboard, which visualizes the measured values, on any other computer that can establish a network connection (Figure 1).

Netdata was originally designed to handle both data acquisition and dashboard representation; however, this approach did not integrate well with cloud environments, where virtual machines are constantly added and removed. Therefore, recent versions of Netdata can also work as a data collector only and delegate display tasks to a central Netdata instance.

Getting Started with Netdata

Netdata is available for Linux, macOS, and FreeBSD. Linux installation is very easy. Several distributions, including Ubuntu, Debian, and openSUSE, make Netdata available in their repositories. If Netdata isn't available through your distro's repositories,

Listing 1: Creating a Netdata User

```
mysql> create user 'netdata'@'localhost';
mysql> grant replication client on *.* to
'netdata'@'localhost';
mysql> flush privileges;
```

Accessing MySQL Data

Netdata will need a MySQL user account to monitor MySQL. Enter the commands in Listing 1 to create the account. Then edit `/etc/netdata/python.d/mysql.conf` and enter *netdata* as the username without password into one of the existing login templates. Finally, in `/etc/netdata/charts.d.conf`, remove the comment character from the start of the following line:

```
mysql=force
```

After you restart Netdata, numerous new charts will appear for the local MySQL instance (Figure 2).

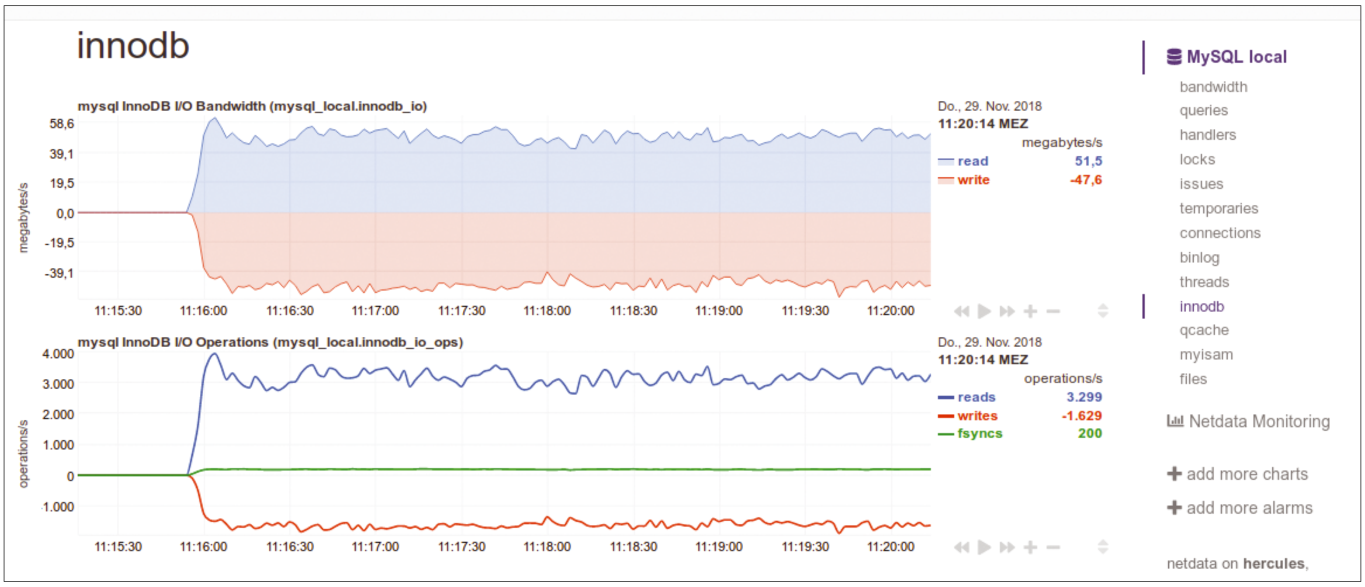


Figure 2: The bandwidth of the InnoDB storage engine within MySQL, measured in megabytes and operations per second.

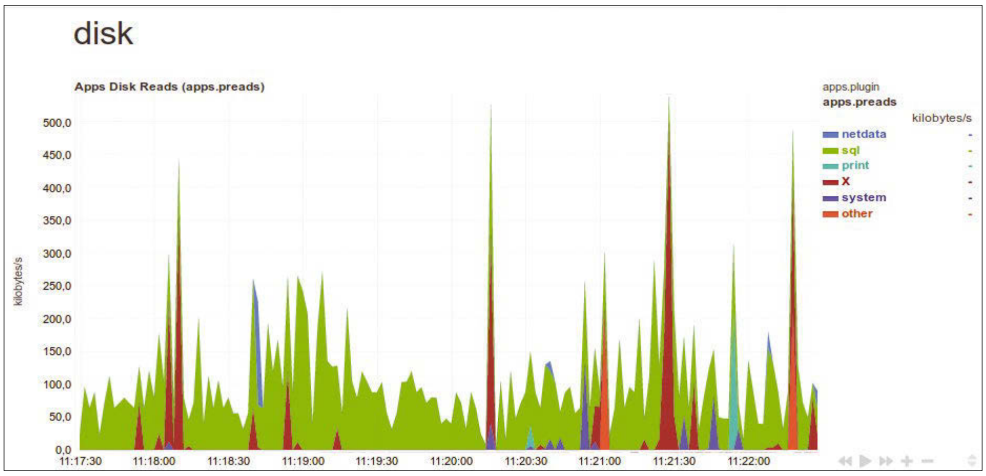


Figure 3: Disk I/O grouped by application. Green represents a SQL database on which a benchmark was running.

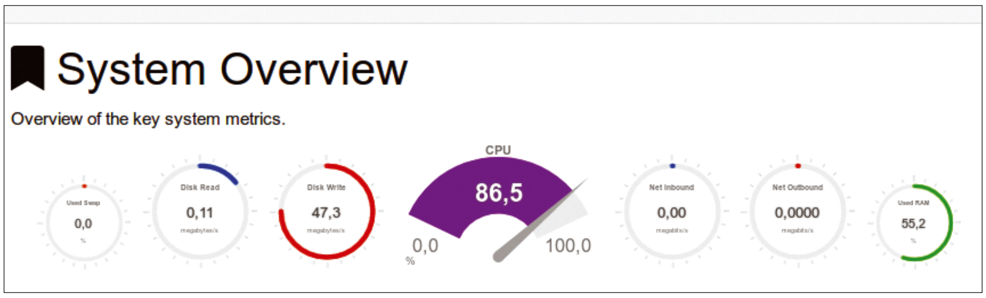


Figure 4: Basic key figures are presented on startup.

Little Configuration

Most Netdata features work without configuration out of the box. The tool supports more than 5,000 metrics without any action on the part of the user, and it comes with some preset limits and preconfigured alarms. In addition to system metrics, Netdata can gather performance data on applications. For instance, Netdata supports a number of database systems, from MySQL to Oracle, plus NoSQL databases such as MongoDB, Redis, CouchDB, and Memcached, and even time-series databases such as Prometheus or Graphite.

Netdata also supports various web servers (Apache, NGINX, lighttpd, and others), mail servers (such as Dovecot, imapd, Amavis, and pop3d), name servers, print servers, time servers, VPN servers, load balancers, and storage systems, such as Ceph, Samba, or NFS. The tool recognizes container hosts (Docker, LXC), backup servers (Rsync, Bacula), and application servers such as Kafka and RabbitMQ. Netdata also supports Java applications.

Administrators can add their own applications under `/etc/netdata/apps_groups.conf`. You can also define your own charts. In the configuration directory (`/etc/netdata`) and its subdirectories, you will find several other documented config files that are needed to store user names and passwords for monitored applications.

In some cases, you might need to configure Netdata manually to access a specific application. For example, if you wish to monitor MySQL, you need to create a suitable user account with limited rights in the database (see the "Accessing MySQL Data" box).

Metrics

Netdata collects performance data for the classic resource groups CPU, disk, network, and main memory. The information available for this purpose is similar to what you get with a tool like `sar` from the *sysstat* package, but with four major differences:

1. Values are always instantaneous, although snapshots can be saved so that the period of interest does not scroll off the screen during your analysis.
2. All data is visualized in real time.
3. In addition to absolute values, Netdata provides useful views grouped by application (Figure 3).

4 The graphics are interactive. If the user moves the cursor along the time axis, the measured values are displayed in numbers to the right of the chart.

Each chart is preceded by an overview of the basic parameters in the form of level meters displaying swap memory usage, disk read and write operations, and the network, CPU, and RAM usage (Figure 4). Time series diagrams break down these values further. Also included are statistics on processes, interrupts, soft IRQs, or IPC semaphores.

Memory statistics contain charts relating to page faults or the size of the kernel memory structures. The disk statistics show the number of megabytes written and read, as well as the I/O operations per medium, and provide information on the size of the backlog (the I/O operations still outstanding).

The CPU load per core is followed by graphs for the number of interrupts and soft IRQs, again for each virtual CPU. Network statistics show information for packets and errors, individual protocols (TCP, UDP, ICMP), broadcasts, multicasts, and fragmentation (broken down by IPv4 and IPv6). Data is also available for the Netfilter firewall.

Conclusion

If you want to collect data for a baseline, or if you want to trace a problem from way back when, Netdata is the wrong choice, because it only tells you what is happening right now. You can preserve this state in a snapshot and then analyze it when you have the time, but you can't freeze the snapshot in the program for more than five minutes. However, if you need to investigate the state of the system at a moment in time, Netdata lets you study the data from hundreds of angles. ■■■

Info

[1] Netdata Git repository: *https://github.com/firehol/netdata.git*
[2] Netdata behind a proxy: *https://docs.netdata.cloud/streaming/#proxies*

OPEN SOURCE
DATA CENTER
CONFERENCE

MAY 14 – 15 2019 | BERLIN

FIRST SPEAKERS SET.

STILL NO TICKET?

JUST GO FOR IT!

OSDC.DE

Measuring performance with the perf kernel tool

Inside Job

The kernel supports performance analysis with built-in tools via the Linux performance counters subsystem. perf is easy to use and offers a detailed view of performance data. *By Paul Menzel*

Making systems faster is a core part of business. Optimizing resources in data centers can save energy, space, and costs, and, out in the real world, a faster start-up time can improve the user experience for an entertainment system in the living room or even a car in the driveway.

In complex systems with many components, performance problems can stem from many different causes. On the hardware side, the CPU, RAM, the bus load, the memory system (block I/O), or the network can contribute to performance issues. On the software side, the operating system, the execution environment of the programming language, libraries, or the application itself can create bottlenecks. The fact that these components also often influence each other makes things more difficult. Although the systems work without any problems on their own, their interaction leads to friction.

To identify the root cause in the interaction of all these components requires the observer to take a bird's eye view and also have detailed knowledge of all the components. Not many IT professionals combine these skills, thus making performance optimization an exciting but challenging activity.

Fortunately, several useful tools are available for monitoring performance issues, including well-known utilities such as `ps`, `stat`, `top`, `htop`, OProfile [1], SystemTap [2], and so on.

One of the most useful performance monitoring tools is `perf` [3], which has been part of the kernel since version 2.6.31. `perf` uses the kernel's performance counter subsystem and supports both profiling and tracing.

During profiling, Perf evaluates data from hardware registers that count specific hardware events. Thus, it generates statistics at certain times in order to gain an impression of the system. The accuracy depends on the frequency of measurement.

In tracing, however, `perf` logs certain events using trace points. These include software events such as context changes, but also hardware events such as executed instructions, cache requests, and so on. System messages are also included.

Admins and developers should keep in mind that tracing can have a major influence on the system. In general, you need to take into account that the measurements themselves can influence the measurement results.

Installation

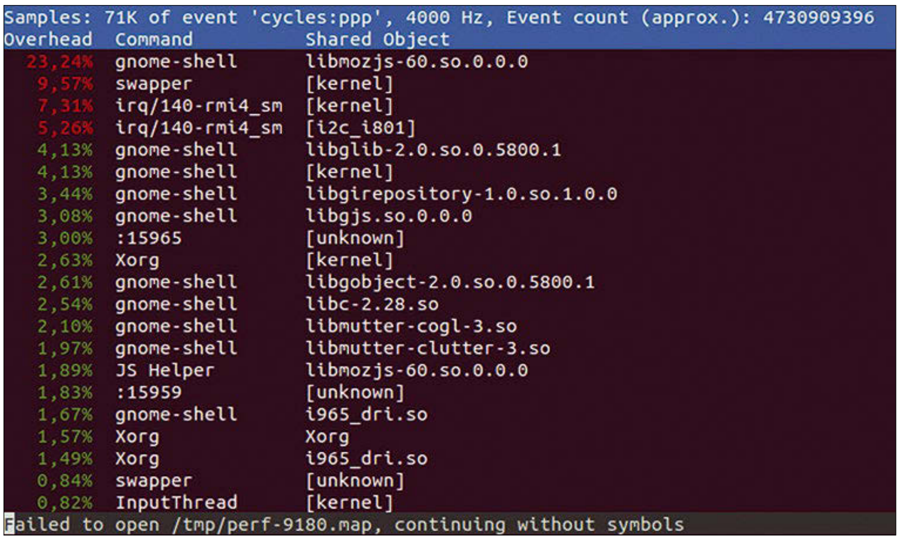
The `perf` program is part of the Linux kernel and located in the `tools/perf` directory. As the output from `make tools/help` shows, you can install with `make -C perf_install`. You can also simply change the directory to `/usr/src/<Linux_version>/tools/perf` and execute `make` there.

Because certain `perf` functions depend on the Linux application binary interface (ABI), distributions often ship one package per kernel version. Under Ubuntu 18.10, the `linux-tools-generic` package always installs the appropriate `perf` version.

If you want meaningful output, you also need to install the debug symbols for the programs you wish to example – just as you would with a debugger. The debug symbols are usually stored in separate archives. The packages automatically generated on Debian and Ubuntu 18.10 have an extension of `-dbgsym`, and the manually generated packages have an extension of `-dbg`.

Workflow

`perf` supports more than two dozen subcommands. For an overview, type the command without any parameters or check out the `perf` wiki [4]. People who use `perf` usually start with `perf list`. This command displays the available events (Listing 1), which you can then count with `perf stat`. The `record` command writes data to the hard disk; `report` displays the data for searching. `script` supports further evaluation of the data at the end.



```
Samples: 71K of event 'cycles:ppp', 4000 Hz, Event count (approx.): 4730909396
Overhead  Command          Shared Object
  23,24%  gnome-shell      libmozjs-60.so.0.0.0
   9,57%  swapper          [kernel]
   7,31%  irq/140-rmi4_sm  [kernel]
   5,26%  irq/140-rmi4_sm  [i2c_i801]
   4,13%  gnome-shell      libglib-2.0.so.0.5800.1
   4,13%  gnome-shell      [kernel]
   3,44%  gnome-shell      libgirepository-1.0.so.1.0.0
   3,08%  gnome-shell      libgjs.so.0.0.0
   3,00%  :15965           [unknown]
   2,63%  Xorg             [kernel]
   2,61%  gnome-shell      libgobject-2.0.so.0.5800.1
   2,54%  gnome-shell      libc-2.28.so
   2,10%  gnome-shell      libmutter-cogl-3.so
   1,97%  gnome-shell      libmutter-clutter-3.so
   1,89%  JS Helper        libmozjs-60.so.0.0.0
   1,83%  :15959           [unknown]
   1,67%  gnome-shell      i965_dri.so
   1,57%  Xorg             Xorg
   1,49%  Xorg             i965_dri.so
   0,84%  swapper          [unknown]
   0,82%  InputThread      [kernel]
Failed to open /tmp/perf-9180.map, continuing without symbols
```

Figure 1: The console browser of `perf top --sort=comm,dso` on an Intel Kaby Lake system with Ubuntu 18.10 and Gnome 3.30.1.

Events

The Admin `perf list` command also gives an optional argument on request (refer to Listing 1). This argument may be the type of the event, for example `hw` or `tracepoint`, or a regular expression, as in Listing 2.

Each end of line contains the event type in square brackets. In addition, you normally have to look into the source code to see the exact meaning of the event. You can use the `--events` or `-e` switch to state the events to be considered.

Listing 1: Outputting Hardware Events (excerpt)

```
$ sudo perf list hardware

List of pre-defined events (to be used in -e):

branch-instructions OR branches [Hardware event]
branch-misses [Hardware event]
bus-cycles [Hardware event]
cache-misses [Hardware event]
[...]
```

Listing 2: Events at Block Level (excerpt)

```
$ sudo perf list 'block:*'

List of pre-defined events (to be used in -e):

block:block_bio_backmerge [Tracepoint event]
block:block_bio_bounce [Tracepoint event]
block:block_bio_complete [Tracepoint event]
block:block_bio_frontmerge [Tracepoint event]
block:block_bio_queue [Tracepoint event]
[...]
```

top on Steroids

You can create a performance counter profile in real time with `perf top` (Figure 1). The overview is similar to that of `top`, but you can jump directly to the individual events.

`perf stat` counts the events of the entire system until you abort by pressing Ctrl + C, or the specified command terminates. Listing 3 shows the subcommand in action; it lists two events for the `factor` command.

Data Recorder

The `perf record` command can generate a performance counter profile for a specific command. If you want an overview of the entire system, or if the process to be examined is unknown, simply add the `sleep` command:

```
sudo perf record -ag sleep 5
```

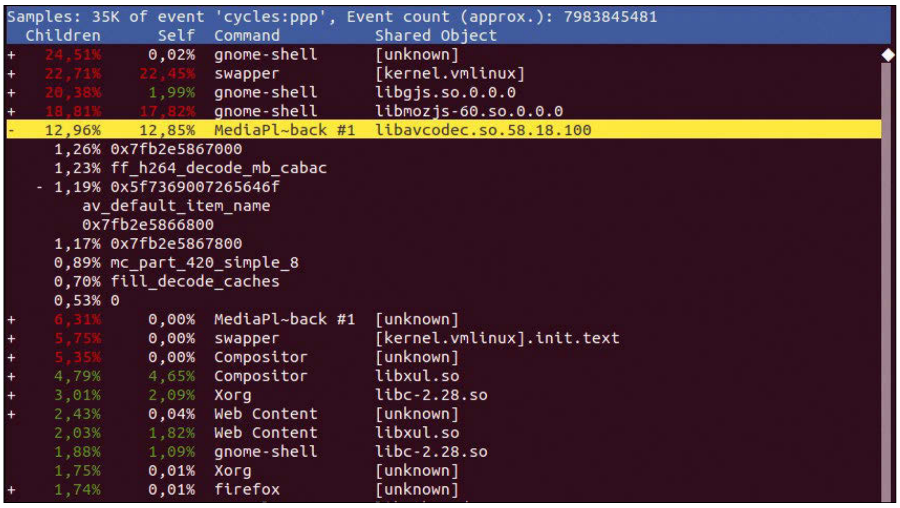
This line creates the default `perf.data` file in the current directory. The `-a` switch ensures that `perf` considers all CPUs, and `-g` generates a call graph. `-i` lets you feed a file to `perf record`.

Listing 3: Output from perf stat

```
$ sudo perf stat -e branches -e branch-misses factor 120808125801214124898080833
120808125801214124898080833: 13 29 911 7589 21089 77471 28369829

Performance counter stats for 'factor 120808125801214124898080833':

191,242 branches
10,332 branch-misses # 5.40% of all branches

0.000649438 seconds time elapsed

0.000699000 seconds user
0.000000000 seconds sys
```

The `perf report` command evaluates the stored data and uses the `perf.data` file. As with `perf top`, the command presents an overview. After pressing Enter, you also see details for the functions.

By the way, the `--sort= comm,dso` switch (see Figure 1) provides a better overview. Figure 2 shows the output for an Intel Kaby Lake system with Gnome 3.30.1 playing a movie in Firefox 63.0.3. The figure shows that the CPU and not the GPU is used for decoding.



```
Samples: 35K of event 'cycles:ppp', Event count (approx.): 7983845481
  Children      Self  Command          Shared Object
+   24,51%     0,02%  gnome-shell      [unknown]
+   22,71%    22,45%  swapper          [kernel.vmlinux]
+   20,38%     1,99%  gnome-shell      libgjs.so.0.0.0
+   18,81%    17,82%  gnome-shell      libmozjs-60.so.0.0.0
-   12,96%    12,85%  MediaPl~back #1  libavcodec.so.58.18.100
     1,26% 0x7fb2e5867000
     1,23% ff_h264_decode_mb_cabac
   - 1,19% 0x5f7369007265646f
        av_default_item_name
        0x7fb2e5866800
     1,17% 0x7fb2e5867800
     0,89% mc_part_420_simple_8
     0,70% fill_decode_caches
     0,53% 0
+    6,31%     0,00%  MediaPl~back #1  [unknown]
+    5,75%     0,00%  swapper          [kernel.vmlinux].init.text
+    5,35%     0,00%  Compositor       [unknown]
+    4,79%     4,65%  Compositor       libxul.so
+    3,01%     2,09%  Xorg             libc-2.28.so
+    2,43%     0,04%  Web Content      [unknown]
     2,03%     1,82%  Web Content      libxul.so
     1,88%     1,09%  gnome-shell      libc-2.28.so
     1,75%     0,01%  Xorg             [unknown]
+    1,74%     0,01%  firefox          [unknown]
```

Figure 2: Perf measures the performance of the default browser with `perf report --sort=comm,dso.`

Event-Driven

If only certain events are of interest, the `-e` switch can help. Listing 4 shows the output on a system with Rsync running.

`--stdio` tells `perf` to display the overview in plain text directly on the console. The recorded event was a request to the kernel block layer. The command line was:

```
sudo perf record -e ↩
block:block_rq_issue -ag
```

Listing 4: Track Events

```
$ sudo perf report --stdio
[...]
# Samples: 1K of event 'block:block_rq_issue'
# Event count (approx.): 1741
#
# Children Self Command Shared Object Symbol
# ........ .... ........... ................ ....................
#
93.74% 93.74% rsync kernel.kallsyms] [k] blk_peek_request
|
|--89.20%-- __lxstat64
| blk_peek_request
|
|--3.45%-- __GI___mkdir
| blk_peek_request
|
|--0.63%-- 0x646c6975622f3930
| __GI___link
| blk_peek_request
|
-0.46%-- 0x6d492f636f642f65
__GI___link
blk_peek_request

89.20% 0.00% rsync libc-2.27.so [.] __lxstat64
|
---__lxstat64
blk_peek_request
[...]
```

On Fire

`perf` can also even create pretty graphics. Thanks to flame graphs, admins and developers can determine the most frequently used software paths. The developer tools integrated into browsers can also generate graphics (Figure 3).

The x-axis shows the load of the individual processes; the y-axis the resolution of each function call. The wider the blocks, the more resources their processes require, which helps you see which areas are problematic are most in need of optimizing.

Admins can also generate flame graphs from `perf` data. You do not have to write the script yourself – look for the script on GitHub [5]:

```
git clone https://github.com/↩
brendangregg/FlameGraph
```

The commands from Listing 5 record the performance data and create a flame graph with `perf script` (Figure 4). The flame graph shows that, although the browser is playing a 720p video, the kernel is pretty much in sleep mode. In the browser, individual sections of the flame graph can be zoomed in and out.

Dynamic Tracing

Using the `perf probe` subcommand and the `--add` switch, admins can define dynamic trace points. Listing 6 shows the procedure

Listing 5: Commands for Creating a Flame Graph

```
sudo perf record -F 99 -a -g -- sleep 10
sudo perf script > out.perf
./stackcollapse-perf.pl out.perf > out.folded
./flamegraph.pl out.folded > out.svg
firefox out.svg
```

BPFtrace

Early in the fall of 2018, Alastair Robertson published the high-level BPFtrace [8] language, which runs in the eBPF VM of the Linux kernel. This allows the kernel to start certain operations, such as filter processes, directly in kernel space, which avoids the need for context switching between kernel and user space and thus puts significantly less strain on the system.

using the Linux `tcp_sendmsg()` function as an example.

Dynamic Duo

The Linux `perf` subsystem and the `perf` tool collection offer many possibilities for analyzing a system. Admins can easily collect and evaluate data with existing tools and scripts to achieve an overview. Correct interpretation of the data requires some knowledge of the `perf` subsystem. Because the Linux kernel delivers `perf` directly, you won't need to worry about installing external modules.

Reading Matter

You will quickly find numerous resources for `perf` on the web. First and foremost certainly are Brendan Gregg's [6] pages; Gregg worked on DTrace at Sun and now uses `perf` intensively at Netflix in his work as a performance engineer. His book *System Performance* [7] is regarded as one of the standard works in this field.

Many admins and developers have discovered that better knowledge of `perf` leads to faster problem analysis. Some development environments also include `perf`, and a few training providers offer `perf` training. It seems clear that performance optimization will continue to play an important role in system administration in the future. BPFtrace (see box "BPFtrace," [8]) might be the next exciting project in the pipeline. ■■■

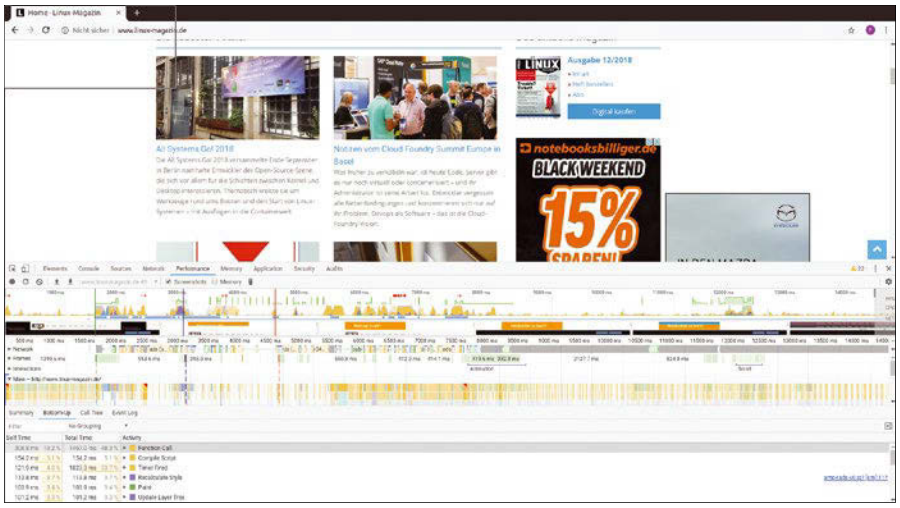


Figure 3: Browser-based developer tools also offer performance analysis features.

Listing 6: tcp_sendmsg as Dynamic Trace Point

```
$ sudo perf probe --addd='tcp_sendmsg'
Added new event:
probe:tcp_sendmsg (on tcp_sendmsg)
[...]
$ sudo perf record -e probe:tcp_sendmsg -aR sleep 5
[ perf record: Woken up 1 times to write data ]
$ sudo perf report --stdio
[...]
```

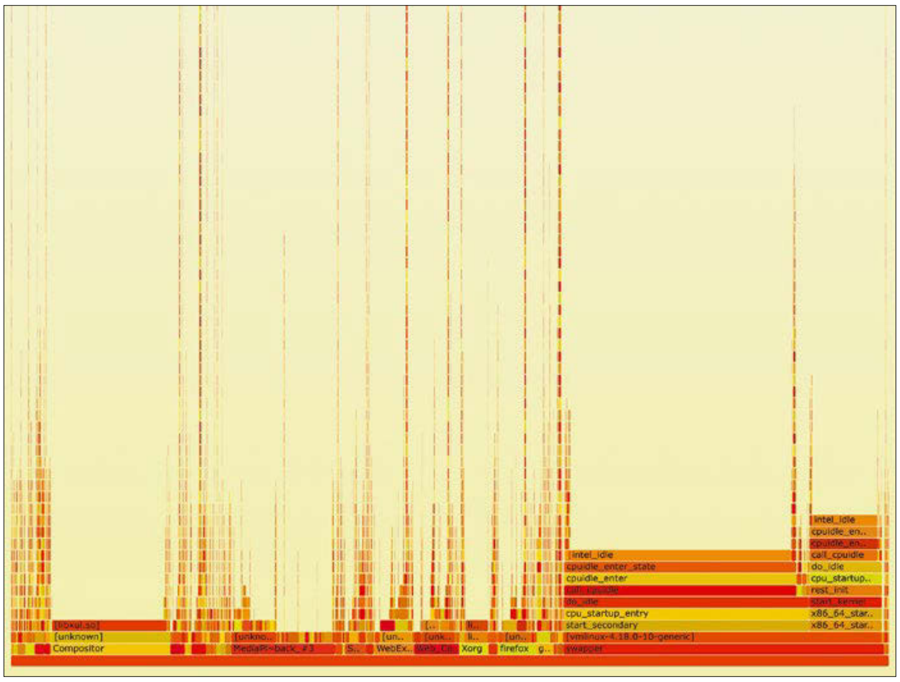


Figure 4: A script-generated flame graph of the `perf` data. If desired, you can zoom in or out at the push of a button.

Info

[1] OProfile: *http://oprofile.sourceforge.net*

[2] SystemTap: *https://sourceware.org/systemtap/*

[3] Official kernel documentation for perf: *https://perf.wiki.kernel.org/index.php/Main_Page*

[4] perf subcommands: *https://perf.wiki.kernel.org/index.php/Tutorial*

[5] Flame graph script: *https://github.com/brendangregg/FlameGraph*

[6] Brendan Gregg on perf: *http://www.brendangregg.com/perf.html*

[7] Gregg, Brendan. *System Performance*. Prentice Hall, 2012: *http://www.brendangregg.com/sysperfbook.html*

[8] BPFtrace: *https://github.com/iovisor/bpftrace/*

Testing of Steam's Wine fork Proton

UNDER STEAM

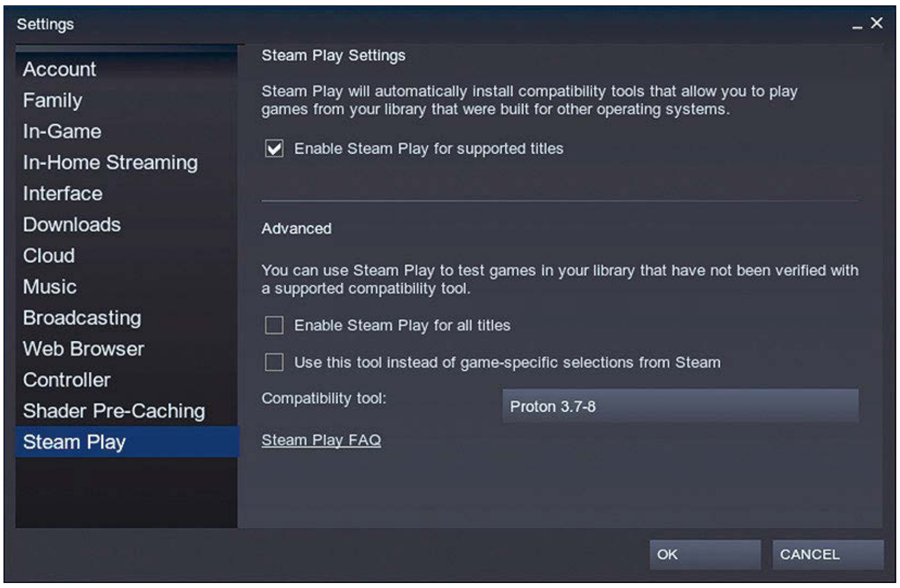
The Proton runtime environment, which is based on Wine, brings a new crop of Steam-powered games to Linux. *By Christoph Langner*

For many years, game developers didn't pay much attention to Linux. Native Linux versions of commercial games were very rare, but with a little luck, you could sometimes get the Windows version to run on your Linux system with the help of the the Wine runtime environment. But slipping a compatibility layer between a Windows game and a Linux OS never was a perfect solution. Wine was only reliable with older games, and getting it working often involved major research and tinkering.

Linux gaming has improved considerably since the early years. In 2013, Valve introduced the Steam client for Linux, which brought native Linux gaming to the most commercially successful gaming platform. However, the economies of the gaming industry ensure that many Windows games will probably never be ported to a native Linux version. Many game developers (and game users) still depend on the Wine environment to run Windows games on Linux.

Steam Play is a service introduced by Steam that lets the user run a purchased game under Windows, Mac OS X, or Linux (if available). More than 3,000 games now run on all common PC operating systems. In order to expand the pool, Steam announced a new version of Steam Play this past summer [1]. For Linux compatibility, Steam Play depends on a fork of the Wine environment called Steam Proton.

As of this writing, only the beta version of Steam Proton, which was presented in August last year, is available to users. The trial version can be activated via *Steam* | *Settings* | *Account*. After restarting, Proton should be listed as a *Compatibility tool* in the settings below *Steam Play* (Figure 1).



Steam Play Settings

Steam Play will automatically install compatibility tools that allow you to play games from your library that were built for other operating systems.

[✓] Enable Steam Play for supported titles

Advanced

You can use Steam Play to test games in your library that have not been verified with a supported compatibility tool.

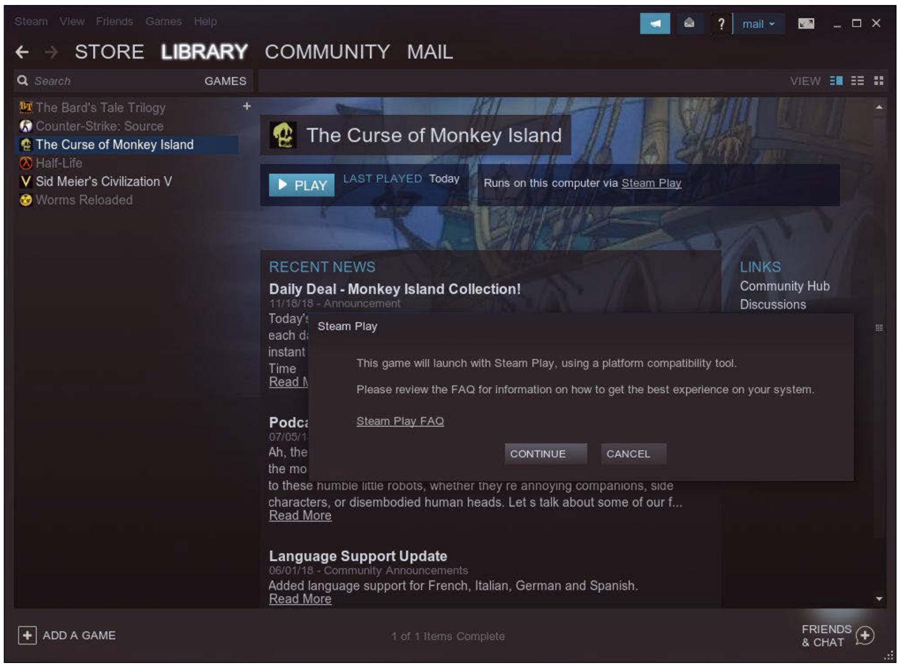
[ ] Enable Steam Play for all titles

[ ] Use this tool instead of game-specific selections from Steam

Compatibility tool: Proton 3.7-8

Steam Play FAQ

Figure 1: For the current version of Steam Play with Proton, you have to activate the beta version of the Steam Client.



Runs on this computer via Steam Play

Daily Deal - Monkey Island Collection!
11/18/18 - Announcement
Today's
each d
instant
Time
Read M

Steam Play

This game will launch with Steam Play, using a platform compatibility tool.

Please review the FAQ for information on how to get the best experience on your system.

Steam Play FAQ

Podca
07/05/1
Ah, the
the mo
to these humble little robots, whether they re annoying companions, side characters, or disembodied human heads. Let s talk about some of our f...
Read More

Language Support Update
06/01/18 - Community Announcements
Added language support for French, Italian, German and Spanish.
Read More

Figure 2: Steam alerts the user that the game will run using "a platform compatibility tool."

Theoretically, you could also build Proton using the source code [2] hosted on GitHub. Steam Play requires a modern distribution with Python 3 and current graphics drivers.

In its announcement, Steam mentions a number of officially compatible titles. These titles include older games, such as the point-and-click adventure game The Curse of Monkey Island, but also (more) recent games with higher demands on graphics performance, including the 2016 reboot of Doom or Tekken 7 from 2017.

Proton Hands-On

The Steam client does not yet display compatible games separately. It is thus a good idea to stick to the list of officially supported games or to look into compatibility with the help of projects like ProtonDB [3]. The ProtonDB database lists over 3,000 compatible games and provides information on compatibility, as well as hardware requirements and user experiences.

The Steam Play beta proved stable in our tests. We were able to download a Windows-only game purchased via the Steam client on Arch Linux and run it directly via the client as if it were native. In the library, the remark *Runs on this computer via Steam Play* indicates that this is not a native Linux game. When the game is launched, a pop-up window also notifies you that a compatibility layer is being used (Figure 2).

The games we tested with Steam Play on our Arch system impressed. The frame rates roughly resembled those when playing the games under Windows 10 on the same computer.

Conclusions

Steam Play arrives along with other advances in the Linux gaming realm, such as the the open source Vulkan API [4], which allows cross-platform hardware access to the GPU. All in all, Linux has never been in better shape when it comes to games.

Valve promises to continually add to the list of games supported by Steam Play, but don't expect *every* game to run on Linux. Games with complex DRM or anti-cheat systems will probably never be fully operational under Linux. ■■■

Info

[1] "A new version of Steam Play": *https://steamcommunity.com/games/221410#announcements/detail/1696055855739350561*

[2] GitHub project for Steam Proton: *https://github.com/ValveSoftware/Proton*

[3] ProtonDB: *https://www.protondb.com*

[4] Vulkan: *https://www.khronos.org/vulkan*

LINUX MAGAZINE
INTERNET OF THINGS
MYCROFT
IS FIREFOX BACK?
MAPPING TOOLS
FREE BOOT!
TERMINAL TUNING
.NET CORE
INNOVATIVE DISTROS
Systemd
PRIVACY
LINUXVOICE

File syncing with unison

The Kitchen Sync

Unison is a handy tool for file syncing, backups, and merging. To get the most out of unison, however, you need to invest time perfecting your preference files to meet your needs. *By Bruce Byfield*

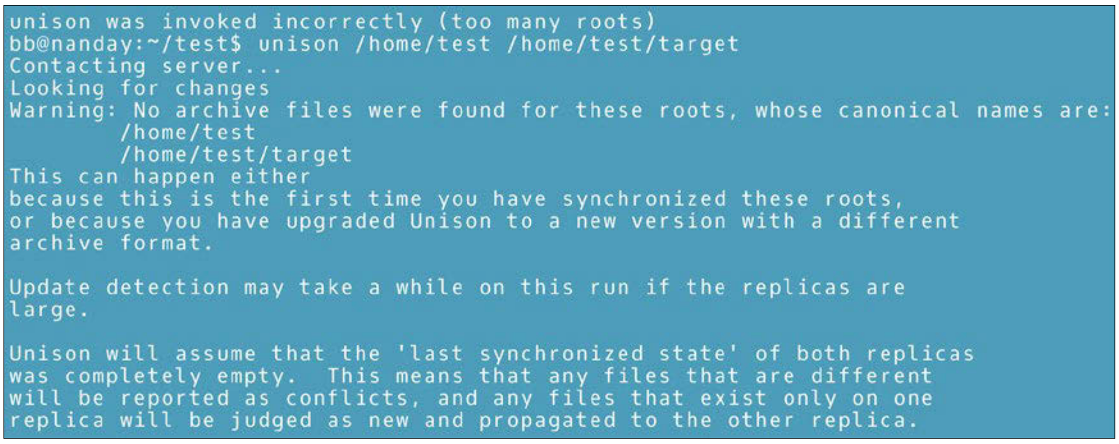
Although `unison` has been around for years, chances are you have never heard of `unison`. For one thing, `rsync` and `ssh` tend to be the default commands for file syncing. For another, `unison`'s documentation is maddeningly incomplete. In the absence of man or info files or usage instructions from the creators, the existing help focuses on building a command to use at the prompt – and, considering that `unison` has 88 options, that is not a very attractive option, even without a command history, especially if you have several sets of files with which you regularly work. So far as I can see, the only distribution that mentions how to simplify the use of `unison` with preference files is Arch Linux, and even it is incomplete – as is the original documentation that Arch references. Yet once preference files are created, the number of on-the-fly options is greatly reduced, and `unison` becomes a handy tool for file syncing, backups, and even merging files.

`unison` is a shell for the `rsync` protocol; it also calls upon external programs like `ssh` and `diff` for its operations. It works with two sets of directories and files, updating one so that the two sets match. Its only drawback is that, if you are syncing in separate machines, each must have the same version of `unison` installed. The basic command structure is:

```
unison PATH1 PATH2
```

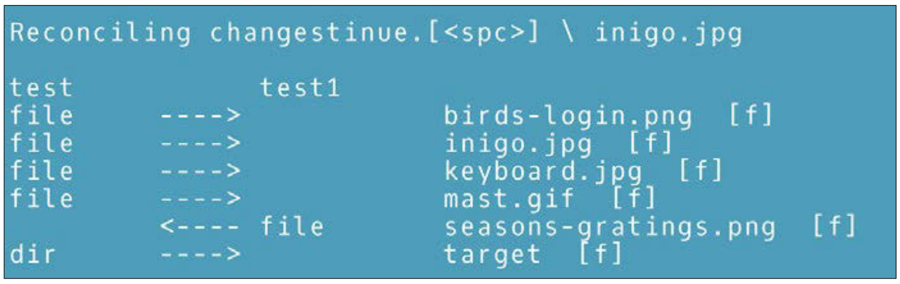
Synchronization is two-way: The most recent version of a file in either directory will overwrite the version in the other. Similarly, any file that exists only in one directory will be copied to the other. The resulting output can have three parts:

1. A standard warning: This warning indicates that this is the first time you have run a synchronization or that archive files do not exist (Figure 1). Scroll past the warning to attempt the synchronization.



```
unison was invoked incorrectly (too many roots)
bb@nanday:~/test$ unison /home/test /home/test/target
Contacting server...
Looking for changes
Warning: No archive files were found for these roots, whose canonical names are:
        /home/test
        /home/test/target
This can happen either
because this is the first time you have synchronized these roots,
or because you have upgraded Unison to a new version with a different
archive format.

Update detection may take a while on this run if the replicas are
large.

Unison will assume that the 'last synchronized state' of both replicas
was completely empty.  This means that any files that are different
will be reported as conflicts, and any files that exist only on one
replica will be judged as new and propagated to the other replica.
```

Figure 1: A warning message indicates that either you are running a particular synchronization for the first time or that archive files do not exist.



```
Reconciling changestinue.[<spc>] \ inigo.jpg

test              test1
file      ---->              birds-login.png  [f]
file      ---->              inigo.jpg  [f]
file      ---->              keyboard.jpg  [f]
file      ---->              mast.gif  [f]
          <----  file        seasons-gratings.png  [f]
dir       ---->              target  [f]
```

Figure 2: The second part of the output is a summary of the changes to be made by the command.

2. A list of proposed changes: Each must be confirmed unless the `-auto` or `-batch` option is used to accept everything automatically. You must enter *y* (yes) to continue (Figure 2).
3. A summary of operations: Arrows to the left show that a file is being transferred from the first root, while an arrow to the right shows a file transfer from the second root. At the end of all operations, a summary displays (Figure 3).

   This basic structure is rapidly complicated by options. All `unison`'s options are prefixed by a single hyphen, with a name that is usually self-explanatory – a useful feature when dealing with so many options. To further simplify matters, options share identical or at least similar names with the fields in a preference file. However, once a preference file is debugged, no root needs to be specified at the prompt. Instead, the structure becomes:

```
unison PREFERENCE-FILE
```

Moreover, the only options you need to add are ones that affect the command's operation, which usually reduces the number of options to add. Often, no options need to be added to the command at all. In addition, preference files can be written to draw on a common file, making the creation of new preference files extremely easy.

Writing Preference Files

Preference files are the sanest way to use `unison`. Preferences are stored in `.unison` in your home directory, with names that end with a `.prf` extension. Typically, you would have a `default.prf` that is used when a preference is not specified with the command, and perhaps other preference files whose names are a description of their contents.

To make a profile, follow these four basic steps using commented lines to help keep track of the sections:

1. Optionally, begin with `label=`, followed by a description of what contents the preference file is designed to sync. This field has no effect on how `unison` interacts with the preference file, being intended only for human convenience. You could add a commented line instead.
2. List the first root directory. For example, for user `bb`, it might be:

```
#Root (source)
root= /home/bb
```

3. List the second root directory to which files will be synchronized. On a separate line, you can also add any `ssh` argument, such as the port to use and the option to run in quiet mode. For example:

```
#Target Directory

root=ssh://nanday.com//home/bb/backup
sshargs=-p4000
sshargs=-q
```

4. List, one per line, the subdirectories and files of the root or source directory to include in the operation. For instance:

```
#Directories and Files to include
path=Projects
path=Fonts
path=Downloads
path=to-do.txt
```

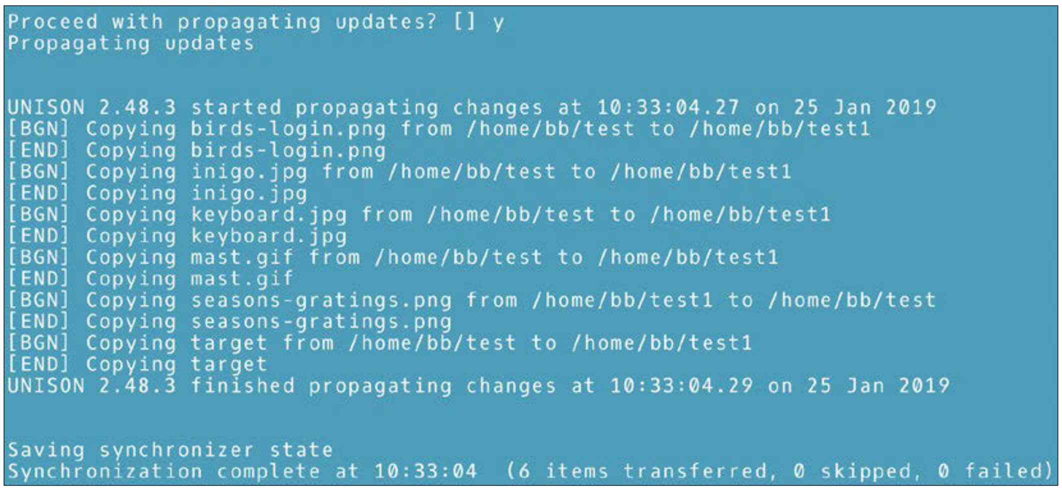
These first three sections (steps 2-4 and optionally step 1), in this order, must be in a preference file. Below them, you can add other sections:

5. List the files to ignore. Regular expressions may be useful in this section:

```
#Files to Ignore
ignore=Virtualbox VMS
ignore=*.tmp
```

6. As files are synced in the second root directory, they will be overwritten. As a precaution, you may want to back up the original files in the second root directory to another location, using these fields. If `backuplocation` is specified as `local`, then backup files are stored in the second root directory; if the specification is `central`, then backup files are stored in another directory that is specified in `backupdir`.

   Backups use the naming convention `.bak.VERSION.FILENAME`. The version



```
Proceed with propagating updates? [] y
Propagating updates


UNISON 2.48.3 started propagating changes at 10:33:04.27 on 25 Jan 2019
[BGN] Copying birds-login.png from /home/bb/test to /home/bb/test1
[END] Copying birds-login.png
[BGN] Copying inigo.jpg from /home/bb/test to /home/bb/test1
[END] Copying inigo.jpg
[BGN] Copying keyboard.jpg from /home/bb/test to /home/bb/test1
[END] Copying keyboard.jpg
[BGN] Copying mast.gif from /home/bb/test to /home/bb/test1
[END] Copying mast.gif
[BGN] Copying seasons-gratings.png from /home/bb/test1 to /home/bb/test
[END] Copying seasons-gratings.png
[BGN] Copying target from /home/bb/test to /home/bb/test1
[END] Copying target
UNISON 2.48.3 finished propagating changes at 10:33:04.29 on 25 Jan 2019


Saving synchronizer state
Synchronization complete at 10:33:04  (6 items transferred, 0 skipped, 0 failed)
```

Figure 3: The third part of the output shows the changes made and their success or failure.

begins with 1 for the most recent and can be limited in number by using the `maxbackups` field. The filename is the name of the original file, including any extension. This convention can be overridden using the `backupprefix` and `backupsuffix` fields, one of which must contain the version. For example:

```
#Backups
backuplocation=central
backupdir=/home/bb/backups
backup=$VERSION.
backupprefix=backup
backupsuffix=
maxbackups=4
```

7. `unison` can call on external commands like `diff` and `diff -3` with the `-diff` option and merge files using `-merge PATTERN`. However, for either option to work, the path to the external command and any options to use with the command must be specified in a preference file. For instance:

```
Diffs and merge
diff=diff -y --suppress-common-lines
merge = ⮐
  diff3 -m CURRENT1 OLD CURRENT2 > NEW
```

8. If you are syncing files on different servers or physical devices, you may want to log each use of `unison`. At the bottom of the preference file, add:

```
#Log
log=true
```

The logfile is located in the `.unison` directory.

Advanced users can save complexity by using preferences another way:

1. In `default.prf`, include only the line:

```
include common
```

2. Create a file called `common` (not `common.prf`). As in the first example, `common` will define the two root directories. It must not contain any `path=` fields but may include `ignore=` and `backup=` fields.
3. Create preference files that contain only `path=` fields for backups, plus the statement `include common`:

```
#Directories to include
path=Projects
path=Fonts
path=Downloads
include common
```

Other than the directories to include, these stripped down preference files will take their characteristics from `common`. Note that if you run `unison` without specifying a preference file when using a `common` file, then all subdirectories will be backed up.

Whether you use basic preference files or set up preference files with a `common` file, the command is simplified to:

```
unison PREFERENCE-FILE
```

However, at least part of the time, you may still want or need to add other options. To give one obvious example, if you are syncing to a flash drive, then chances are you will want to add the `-fat` option because many Linux users format flash drives to use FAT32.

Desktop vs. Command Line

`unison` also supports a GTK+ graphical interface. This interface is often a separate package from `unison`. In Debian, it is called `unison-gtk`.

With `unison-gtk`, you can create basic preference files using the included wizard. However, if you want to list files to ignore, or any other more advanced feature, you will have to open the preference file in a text editor. In particular, you might want to add keyboard shortcuts in the preference file. For example, the lines:

```
#Downloads Backup
path=Downloads
batch=true
key=1
include common
```

This entry will customize `unison` so that, when the GUI is open, pressing the *1* key will signal `unison` to automatically sync any updates found in the `Downloads` directory, using the information found in the `common` file.

However, although `unison-gtk` is convenient for average users, there may still be many times when you want to run `unison` from the command line to take advantage of some of the options. Despite the availability of `unison-gtk`, taking advantage of the prompt still requires a terminal.

Entering Options from the Prompt

Using preferences usually means that no options need to be entered at the prompt. Those that might be occasionally needed are normally general ones about how `unison` operates, rather than about content. If manually entered options are used, they will override those specified in the preference file.

For example, when the output lists changed files, `-sortbysize`, `-sortfirst PATTERN`, or `-sortnewfirst` will change the order of items on the list. You might also choose `-auto` to accept nonconflicting default options without an interactive prompt. Alternatively, you might specify that you are prompted before deleting large numbers of files with `-confirmbigdel` or `-confirmmerge` before merging files. You might also add `-links` to assure the copying of symbolic links.

If one root is on an external drive formatted with FAT32 – a common occurence in Linux – you will need to add the `-fat` option. Similarly, if you are making a backup, you will not want to confirm each operation, so you use the `-auto` option to disable confirmations.

Generally, the more you use `unison` and create preference files to meet your needs, the less likely you are to need to add options manually. If you find yourself adding the same options all the time, check to see if they can be added to the preference file instead. Your use of `unison` will be far easier if you can.

Taking the Time

Like a template in an office suite, `unison`'s preference files can take time to adjust. Ordinarily, you will probably need to use a preference file several times to get it exactly suited to your needs – even without typos.

However, by taking the time to perfect your preference files, you make `unison` far less formidable and considerably more efficient. Once you are set up, `unison` is actually one of the easiest-to-use syncing tools available for Linux. Whether you are making external backups, or syncing files on different machines, a properly configured `unison` is worth consideration as a maintenance tool. ■■■

DrupalCon
SEATTLE 2019
APRIL 8-12, 2019

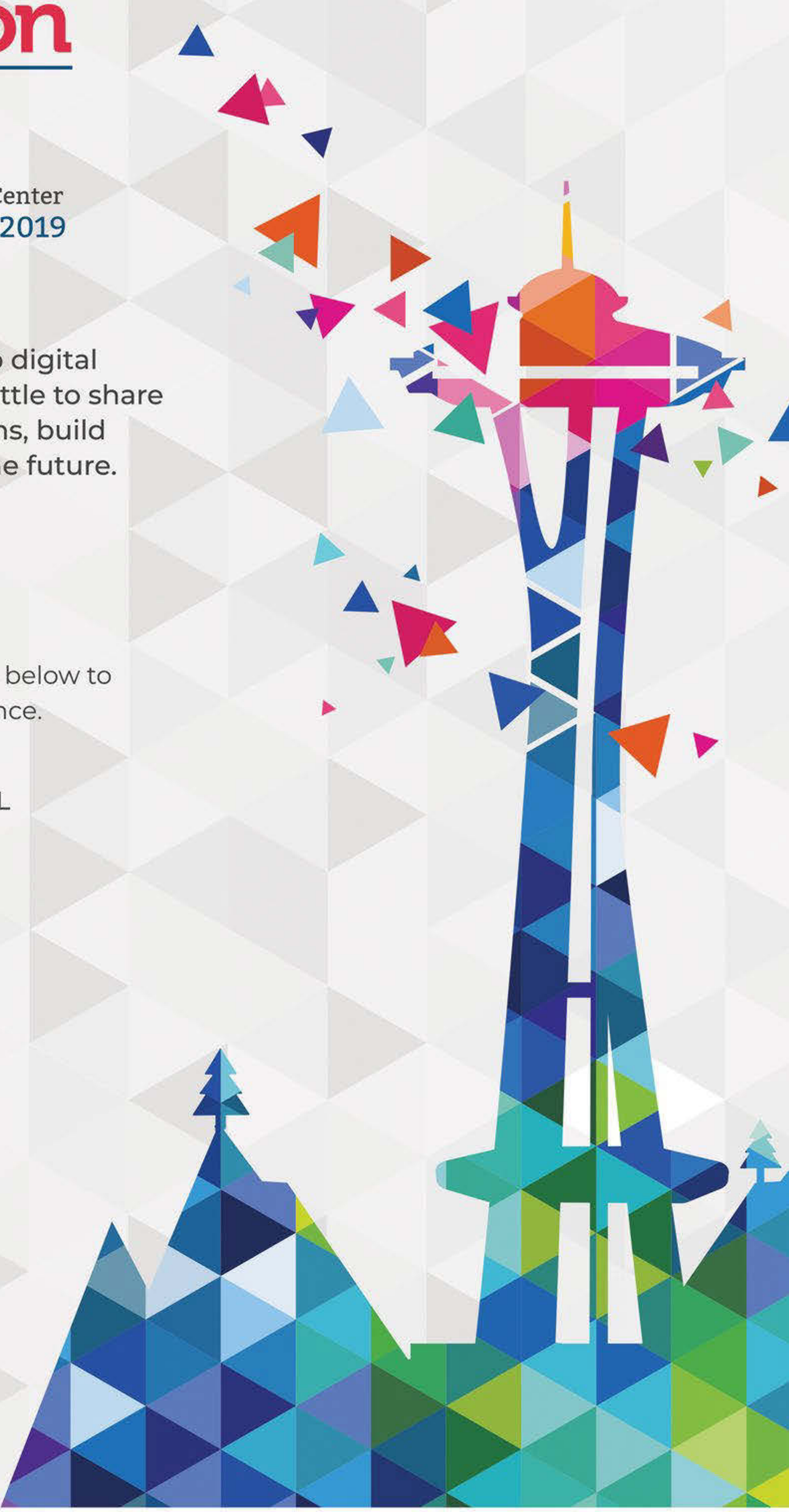
Washington State Convention Center
events.drupal.org/seattle2019

More than 3,000 of the top digital minds will convene in Seattle to share knowledge, create solutions, build relationships and shape the future.

Join us.

Register for one of the tracks below to optimize your onsite experience.

- CONTENT & DIGITAL MARKETING
- BUILDER
- AGENCY LEADERSHIP
- EXECUTIVE SUMMIT

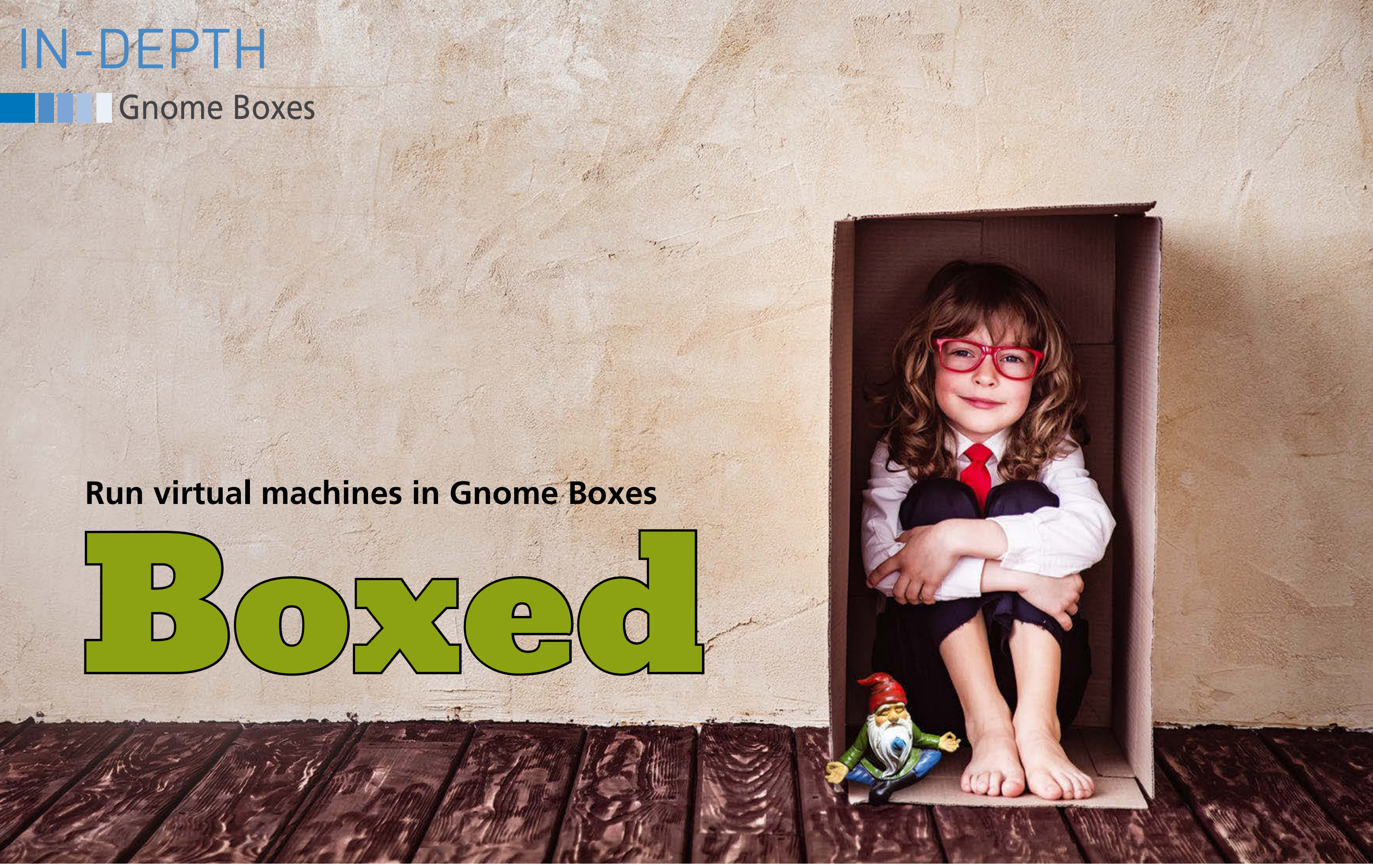


DIAMOND SPONSORS

PANTHEON Website Management Platform

Acquia

platform.sh

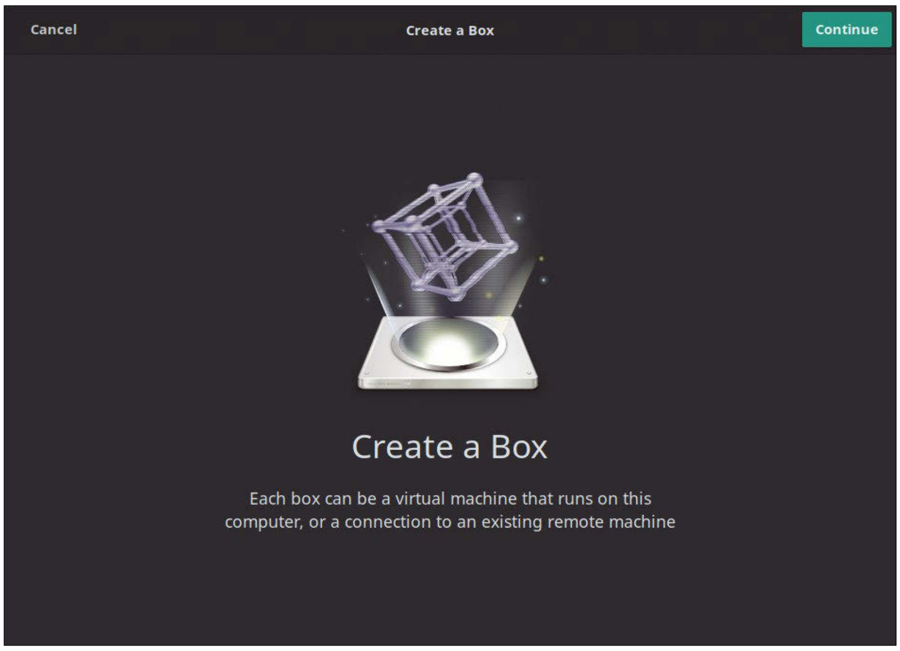


Run virtual machines in Gnome Boxes

Boxed

In the past, using virtual machines required expensive programs such as VMware or open source add-ons such as VirtualBox. Today, thanks to Gnome Boxes, many distributions native support for virtual machines. *By Christoph Langner*

If you want to set up a virtual machine (VM) with a graphical user interface on Linux, you might be inclined to go with VirtualBox or VMware's commercial offerings. These applications, which have been established for years, offer many functions. However, due to their full version's proprietary licenses, they are not found in the package sources of popular Linux distributions.



Create a Box

Each box can be a virtual machine that runs on this computer, or a connection to an existing remote machine

Figure 1: The setup wizard lets you download numerous distributions directly from Gnome Boxes.

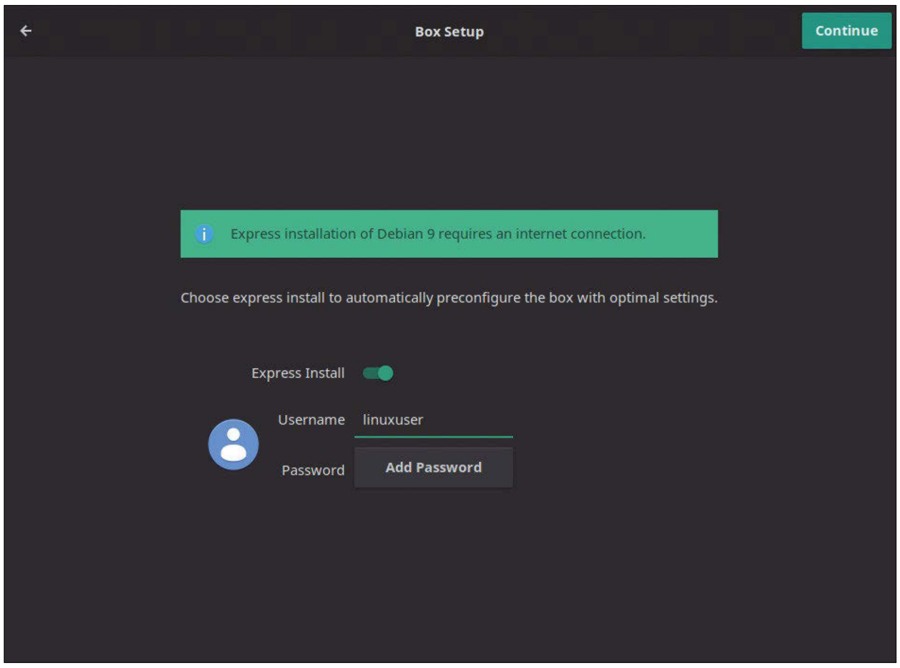
With VirtualBox, you would have to install the Extension Pack alongside the program to use the application's full functionality. In addition, it is important to pay attention to licences. VirtualBox's source code is released under the GPL v2.0, but you can only use the proprietary add-on free of charge [1] for personal use or testing purposes.

Boxes Out of the Box

Because Gnome Boxes, a front-end tool for Kernel-based Virtual Machine (KVM), is created within the framework of the Gnome Desktop Environment [2], it does not require the installation of any additional software (see the "SPICE Mix" box). With KVM directly integrated into

SPICE Mix

Boxes uses the SPICE protocol [4] for communicating between the desktop and the VM. Much like VNC, SPICE transfers the screen content, plus mouse and keyboard input. However, it is not limited to the image; it also transmits audio and couples USB devices with the VM.



Express installation of Debian 9 requires an internet connection.

Choose express install to automatically preconfigure the box with optimal settings.

Express Install

Username linuxuser

Password Add Password

Figure 2: Thanks to *Express installation* for distros like Debian 9, Gnome Boxes sets up the guest operating system without any further questions.

Listing 1: Linking Storage Locations

```
$ mv ~/.local/share/gnome-boxes /<I>Path<I>/<I>Folder<I>/
$ ln -s /<I>Path<I>/<I>Folder<I>/gnome-boxes ~/.local/share/
```

Tip

Gnome Boxes stores the VMs under `~/.local/share/gnome-boxes/`. Boxes does not offer an option to change this path. If you prefer a different location, the directory can still be changed. To do this, stop all VMs, and then move the directory. Then create a symbolic link between the old and new storage locations (Listing 1).

the Linux kernel, Boxes does not have to worry about virtualization. The software simply provides the VM with the environment, using existing libraries and applications such as libvirt and Qemu [3].

Distributions that deliver a complete Gnome desktop include Gnome Boxes. Examples include Arch Linux, which automatically packs the application onto the hard disk along with the *gnome* metapackage. Other distributions, such as Ubuntu, do not include Boxes during the system's initial installation, but do include the application in the software sources (the *gnome-boxes* package).

When first launched, Boxes boots up in a very tidy window. Pressing *New* in the upper left corner launches the setup of an initial VM. A wizard pops up to help you (Figure 1). Thanks to the *Source Selection* option, you don't even have to load an image of the distribution you want to install. The program directly downloads variants of the most popular distributions. If the desired variant is missing, the search function in the window titlebar quickly finds a matching entry.

However, the database is not up to date. As of November 2018, Boxes did not know anything about Ubuntu 18.10, which was released earlier in October. In such a case, you need to manually download the system image as an ISO file off the web and then enter it as a boot image via the *Select a file* option. Following the selection, Boxes then jumps directly to the next step.

You can customize the VM to your needs by setting the RAM size for the VM and the virtual disk's size. A click on *Create* then starts the VM installation; the installation wizard in your chosen distribution will guide you. (See the "Tip" box for information on the directory path.)

The process is faster if you install distributions or operating systems supported by libosinfo [5]. In addition to Debian 9, this also includes Windows 7 or Windows 10, but not Ubuntu 18.04 or the recently released Ubuntu 18.10. In this case, Boxes offers an *Express installation*, which loads the system into the VM in a completely automated process. The only option (at least for Linux guests) is the ability to specify the username and password (Figure 2). Pressing *Next* then continues the installation without any further intervention.

During the install, you will already find the new VM in the application's main screen (Figure 3). When the VM is active, the icon shows a colored live image of the system in the

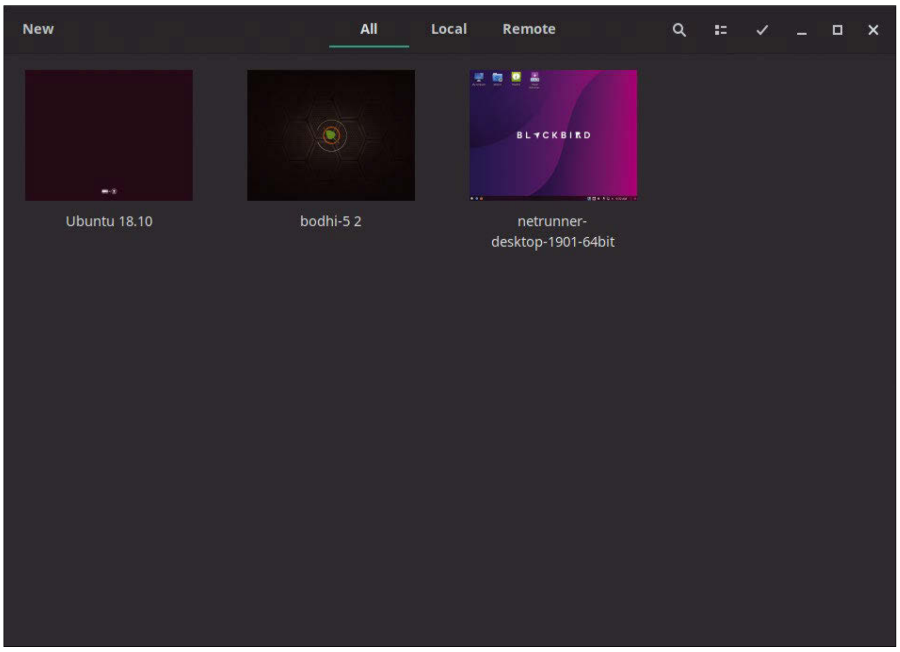
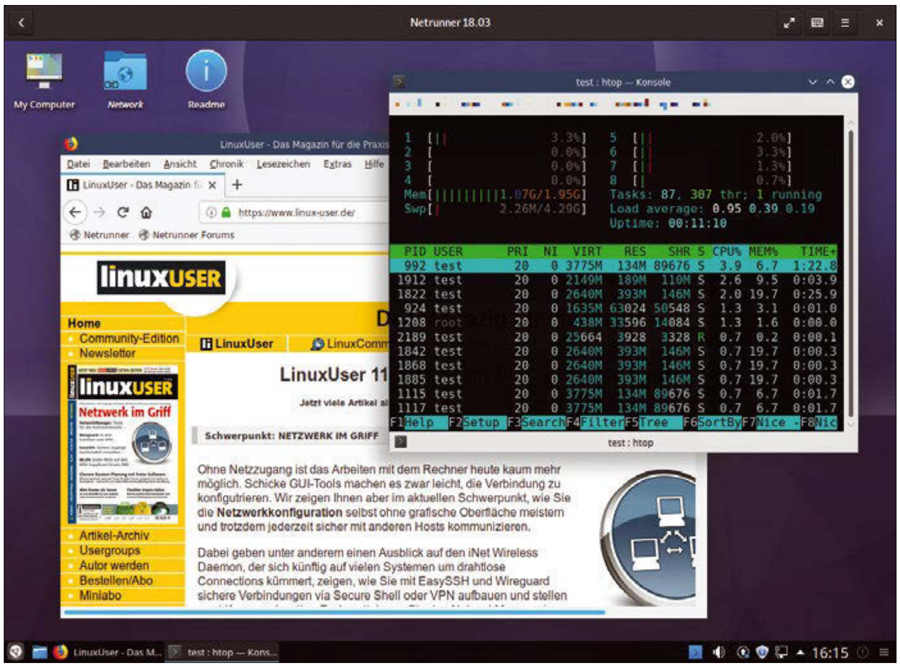


Figure 3: On the home page, Gnome Boxes shows the available VMs. The thumbnail reflects the live state of the systems.



```
1  [||                3.3%]   5  [||               2.0%]
2  [                  0.0%]   6  [||               3.3%]
3  [                  0.0%]   7  [||               1.3%]
4  [                  0.0%]   8  [|                0.7%]
Mem[|||||||||1.07G/1.95G]    Tasks: 87, 307 thr; 1 running
Swp[|         2.26M/4.29G]    Load average: 0.95 0.39 0.19
                              Uptime: 00:11:10

  PID USER      PRI  NI  VIRT   RES   SHR S CPU% MEM%   TIME+
  992 test       20   0 3775M  134M 89676 S  3.9  6.7  1:22.8
 1912 test       20   0 2149M  189M  110M S  2.6  9.5  0:03.9
 1822 test       20   0 2640M  393M  146M S  2.0 19.7  0:25.9
  924 test       20   0 1635M 63024 50548 S  1.3  3.1  0:01.0
 1208 root       20   0  438M 33596 14084 S  1.3  1.6  0:00.0
 2189 test       20   0 25664  3928  3328 R  0.7  0.2  0:00.1
 1842 test       20   0 2640M  393M  146M S  0.7 19.7  0:00.3
 1868 test       20   0 2640M  393M  146M S  0.7 19.7  0:00.3
 1885 test       20   0 2640M  393M  146M S  0.7 19.7  0:00.3
 1115 test       20   0 3775M  134M 89676 S  0.7  6.7  0:01.7
 1117 test       20   0 3775M  134M 89676 S  0.7  6.7  0:01.7
F1Help  F2Setup F3SearchF4FilterF5Tree  F6SortByF7Nice -F8Nic
```

Ohne Netzzugang ist das Arbeiten mit dem Rechner heute kaum mehr möglich. Schicke GUI-Tools machen es zwar leicht, die Verbindung zu konfigutrieren. Wir zeigen Ihnen aber im aktuellen Schwerpunkt, wie Sie die **Netzwerkkonfiguration** selbst ohne grafische Oberfläche meistern und trotzdem jederzeit sicher mit anderen Hosts kommunizieren.

Dabei geben unter anderem einen Ausblick auf den iNet Wireless Daemon, der sich künftig auf vielen Systemen um drahtlose Connections kümmert, zeigen, wie Sie mit EasySSH und Wireguard sichere Verbindungen via Secure Shell oder VPN aufbauen und stellen

Figure 4: If the SPICE Guest Agent is enabled on the virtual system, files can be exchanged between the guest and host. The resolution is adjusted automatically.

File Sharing

With the *Properties | General* option enabled below *Share clipboard*, you can easily exchange data between the guest and host systems. Texts are transferred in both directions using Ctrl+C and Ctrl+V. Simply drag files and folders from the File Manager to the VM application window. You will then be taken to the `Downloads/` folder of the corresponding user. Alternatively, load files into the VM via the hamburger menu and the option *Send file …*. For the opposite direction, that is, from guest to host, this method does not yet work.

To copy data from the VM to the host or to set up a permanent data channel between the VM and the host system, configure a new shared folder below *Properties | General | Devices and shares*. In the file manager, the share will appear as *Spice client folder* under *+ Other locations* (Figure 6).

User Mode Networking

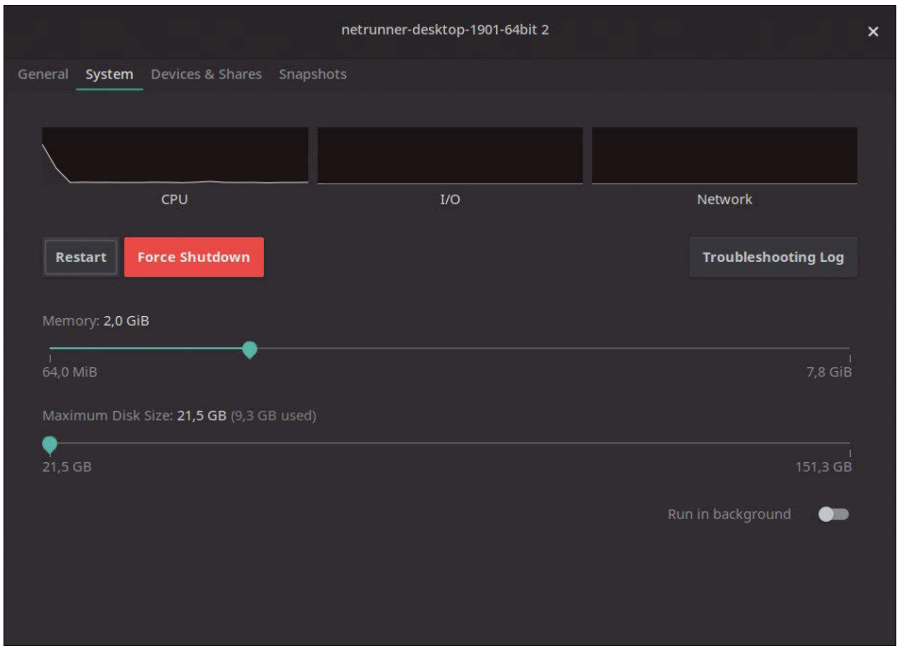
In the basic configuration, Qemu/KVM sets up a separate network between the host system and the guest. From the guest system you can reach the host via IP address *10.0.2.2* or *192.168.122.1* depending on the distribution. However, guests are not allowed to communicate with each other, and the guest cannot be reached from the host. To run a LAN-accessible service, such as a mail or database server, on the guest, you first need to configure a network bridge on the host [8].

thumbnail. An inactive VM appears as a gray thumbnail. Clicking on the icon opens the VM in large format (Figure 4).

To return to the overview, use the left arrow in the top left corner of the application window; the VM continues to run in the background. You need to explicitly shut down the VM before Boxes really switches it off. If the mouse pointer is trapped in a VM, you can free it using the Ctrl + Alt keyboard shortcut.

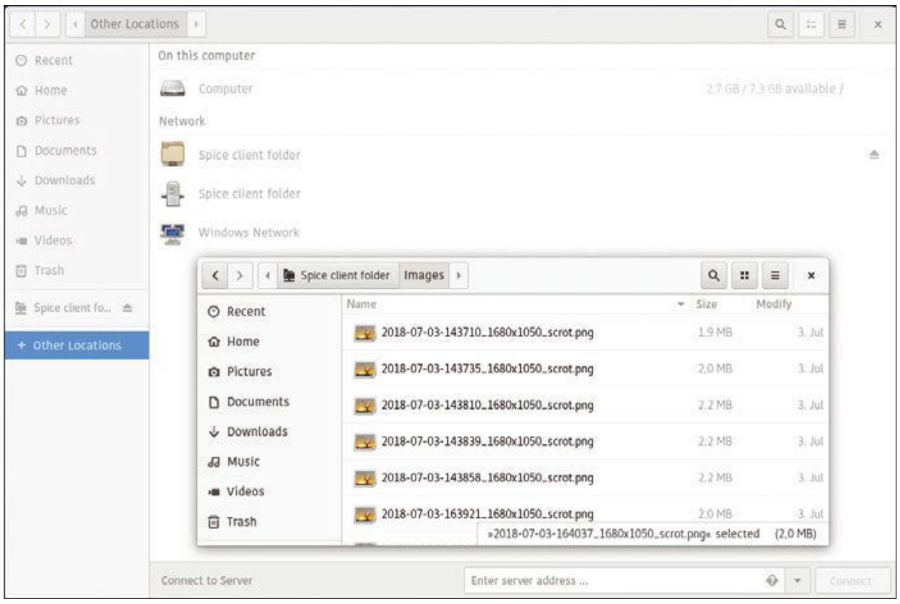
Using the context menu in the overview (or the active VM's hamburger menu), you can access *Properties*. This is where you can share the contents of the clipboard between the VM and the host system (see "File Sharing" box) and forward devices connected to the host to the VM below *Devices and Shares*. In the *System* tab, you will find graphical history displays of CPU load, read/write operations, and network traffic. You can also adjust the size of the working memory and the virtual hard disk (Figure 5).

If a VM no longer reacts, you can also force a restart or shutdown here. If you only need the machine for services in the background (such as a network stack for the web server and database – see the box entitled "User Mode Networking"), the *Run in background* option in this window can also be used to prevent the VM interface from opening. An interface configured in the VM is not affected by this option and still requires resources. It is therefore best to switch off the launch of the graphical environment completely in the corresponding VM or install the system without a GUI from the outset.



Memory: 2,0 GiB
64,0 MiB 7,8 GiB
Maximum Disk Size: 21,5 GB (9,3 GB used)
21,5 GB 151,3 GB
Run in background

Figure 5: The optional settings for the individual VMs are very limited. You can only change the size of the RAM and the virtual hard disk.



Name	Size	Modify
2018-07-03-143710_1680x1050_scrot.png	1.9 MB	3. Jul
2018-07-03-143735_1680x1050_scrot.png	2.0 MB	3. Jul
2018-07-03-143810_1680x1050_scrot.png	2.2 MB	3. Jul
2018-07-03-143839_1680x1050_scrot.png	2.2 MB	3. Jul
2018-07-03-143858_1680x1050_scrot.png	2.2 MB	3. Jul
2018-07-03-163921_1680x1050_scrot.png	2.0 MB	3. Jul

»2018-07-03-164037_1680x1050_scrot.png« selected (2.0 MB)

Figure 6: A shared folder from the host system appears in a Gnome Boxes VM as *Spice client folder* in the network neighborhood.

The last tab, *Snapshots*, offers the possibility to take snapshots of the system loaded in the VM – handy if you want to experiment with the system or gain experience with a new distribution. All you have to do is click on the plus button. Using the gear menu next to the individual snapshots, you can restore an old state at the push of a button, rename the snapshot, or delete it from the hard disk.

Conclusions

In everyday life, Gnome Boxes impresses as an easy-to-use virtual system. In particular, current distributions with Gnome desktops fit seamlessly into the host system, from installation

Guests

For the guest and host to understand each other perfectly, the SPICE Guest Agent (the binary is usually named `spice-vdagent`) needs to be running on the guest – the service can be compared with the VMware tools or VirtualBox's Guest Extensions. For guest systems with Gnome desktops (since version 18.04 this also includes Ubuntu), this is usually the case by default. The resolution of the guest system is automatically adjusted to the window size in the live system; files can be transferred using drag and drop and the guest and host clipboards are synchronized. In addition, the image build-up in the VM with the SPICE Agent enabled is considerably accelerated.

However, on systems with other desktops, such as KDE Plasma or Xfce, you need to install the SPICE Agent retroactively. For Debian based distributions, do this as shown in Listing 2. Depending on the distribution, the virtualized system needs a reboot afterwards. In our lab, KDE Plasma 5.12.2 proved to be fully compatible with Netrunner 18.03. With Xfce 4.12 under Linux Mint Xfce 19, the clipboard worked in our lab, but transferring files failed. Also, the VM's resolution was not adjusted after changing the window size, until we ran the command:

```
xrandr --output Virtual-1 --auto
```

Exotic desktops like Moksha 0.3.0 from Bodhi Linux 5.0.0 didn't really understand what to do with the SPICE Guest Agent. According to the developers responsible for the SPICE agent at Red Hat, the fault lies with the desktops themselves [6].

For Windows guests, you need to load and install the SPICE Guest Tools [7] on the VM (Figure 7). On Windows 7 and Windows 10, the clipboard can be synchronized and files can be transferred using drag and drop, but automatic resolution adjustment does not work. For an optimized display, there is only the possibility to configure the resolution manually in the Windows system's settings or to switch the graphic back end to QXL, which is not possible with Boxes itself.

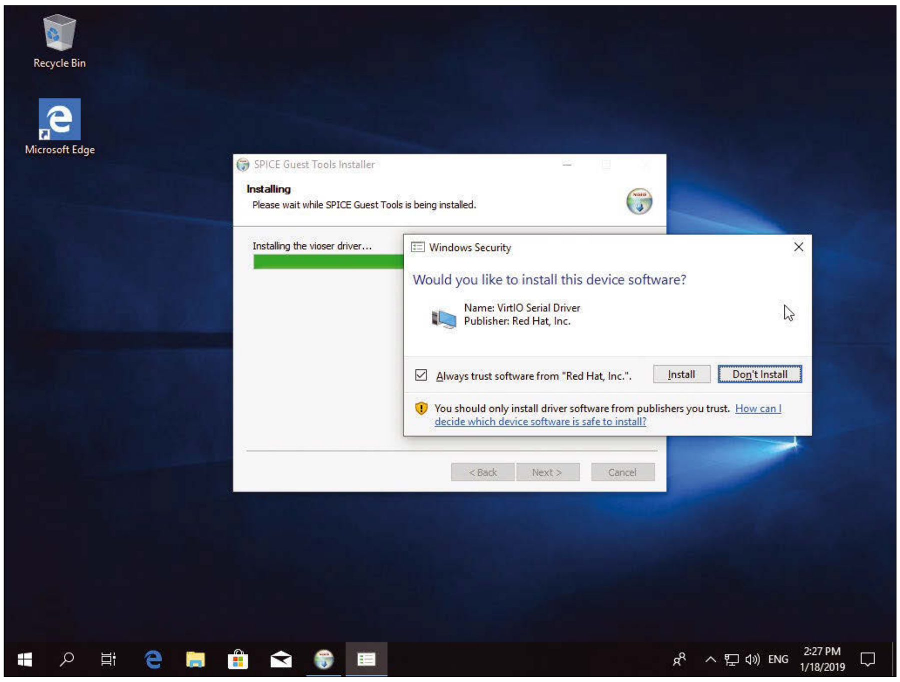
to a running system (see the "Guests" box). However, Boxes does not yet support advanced functions, such as dragging virtualized applications into the host system's window manager.

Listing 2: Installing the SPICE Agent on Debian

```
$ sudo apt install spice-vdagent
$ sudo systemctl enable spice-vdagent
$ sudo systemctl start spice-vdagent
```

The most important functions work without too much overhead. No guest extensions are needed for copying and pasting texts between Linux guests and the host system if you choose the right system. Adjusting the resolution of the virtual system to the size of the application window also works for current distributions without installing additional drivers on the guest system.

All in all, Gnome Boxes is an impressively simple and unpretentious VM solution. A Linux VM can be set up with a few clicks, from downloading the ISO image to setting up the graphical desktop. However, it lacks options for controlling the virtual system. For example, there is no option for managing the number of virtual CPU cores in Boxes. For some, this restriction to the bare essentials may be welcome; for others, Boxes offer enough to justify a switch from VirtualBox. Good thing there's a range of great VM options on Linux. ■■■



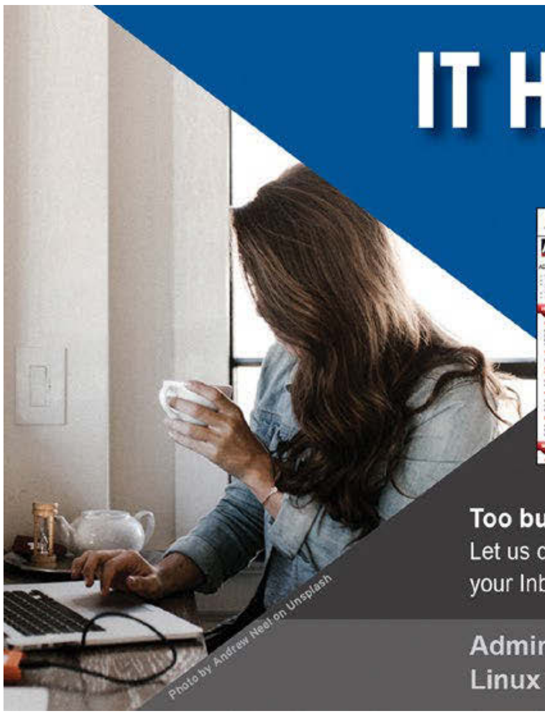
SPICE Guest Tools Installer
Installing
Please wait while SPICE Guest Tools is being installed.
Installing the vioser driver...

Windows Security
Would you like to install this device software?
Name: VirtIO Serial Driver
Publisher: Red Hat, Inc.
☑ Always trust software from "Red Hat, Inc.".
You should only install driver software from publishers you trust. How can I decide which device software is safe to install?

Figure 7: For Windows guests to work with Gnome Boxes, you must install the SPICE Guest tools on the virtual system.

Info

[1] Oracle VM VirtualBox Extension Pack Personal Use and Evaluation License: *https://www.virtualbox.org/wiki/VirtualBox_PUEL*
[2] Gnome Boxes: *https://wiki.gnome.org/Apps/Boxes*
[3] Qemu: *https://www.qemu.org*
[4] SPICE: *https://www.spice-space.org*
[5] libosinfo: *https://libosinfo.org*
[6] "Auto-screen-resolution change of qxl/spice guest not working": *https://bugzilla.redhat.com/show_bug.cgi?id=1290586*
[7] SPICE Guest Tools download: *https://www.spice-space.org/download.html*
[8] Bridged networking in Ubuntu: *https://help.ubuntu.com/community/KVM/Networking#Bridged_Networking*



SUSECon '19

MY KIND OF OPEN

Nashville, TN | April 1-5, 2019

• 150+ sessions
• 20+ expert-led demonstration stations
• 100+ hours of hands-on technology training
• 8+ complimentary certification exams for Linux, OpenStack or Storage

Register today at susecon.com!

SUSE

The sys admin's daily grind: sudoers

Strict Education

"I've seen penguins that can type better than that." If you give sudo the wrong password, you deserve to be shouted at, says sys admin columnist Charly. He is not exempt from the insult and sees it as an opportunity to raise sudoing awareness. By Charly Kühnast

If you work with and on Linux, you are likely to type `sudo` regularly to execute programs with another user's privileges – typically the superuser's. `sudo` will then understandably ask for the appropriate password. In times when my fingers refuse to obey me and I enter the secret password wrong three times in succession, `sudo` rejects my request to escalate. Okay, if you fail to type the password correctly despite having had three attempts, maybe you shouldn't be messing around on the system with root privileges. So far, so good, but free software can surely offer more than that!

sudo visudo

`sudo` looks for its default settings in the `/etc/sudoers` file. You can't just open it with an editor; instead, you need to use the `visudo` command – assuming you're root. (If I'm on the road as a user, I have to type `sudo visudo`, which strangely makes me laugh.) At the beginning of the file are some lines that start with *Defaults*. You need to add two lines here (Figure 1):

```
Defaults insults
Defaults passwd_tries=5
```

The first line causes `sudo` to output a silly saying after every wrong input. The second line increases the number of allowed failed attempts to five. To test this, I deliberately acted stupid again and got the result shown in Figure 2.

Tomorrow's world: I was wondering whether I could perhaps store some insults customized for myself – or better still, for my favorite co-workers? Yes, I can! You need to store a string in `/etc/sudoers`, which is displayed after typing the wrong password:

```
Defaults badpass_message=⮐
  "You shall not pass!"
```

For this to work, you have to remove the `Defaults insults` entry.

"Just What Do You Think You're Doing, Dave?"

That's not bad. A list of different answers, from which `sudo` then randomly selects one, would be even better of course. Unfortunately, this only works by changing the answers in the source code and then recompiling them.

I found some source code files containing useful sayings online [1]: `ins_2001.h`, `ins_classic.h`, `ins_python.h`, `ins_goons.h`, `ins_csops.h`, and `insults.h`.

The file names indicate the movies or TV shows from which the quotes are taken. The `ins_python.h` file has nothing to do with the programming language, of course, but contains legendary sayings from Monty Python, such as, "I fart in your general direction!".

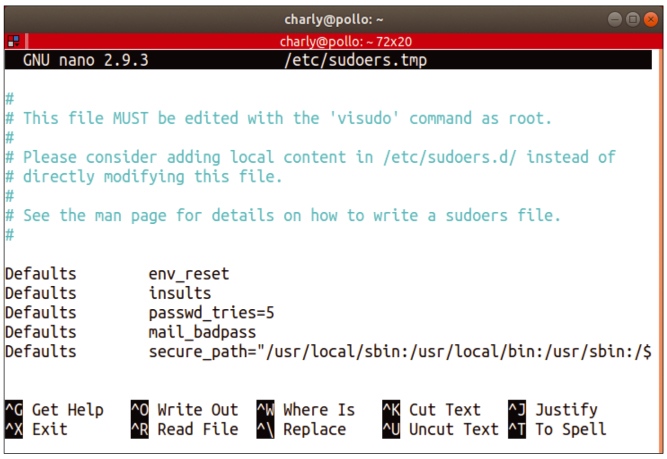
While I am suffering from regular verbal humiliation, the thought occurred to me that I could announce a competition among my co-workers for the best `sudo` insults. See you next time, dummies! ∎

Info

[1] sudoers: *https://github.com/millert/sudo/tree/master/plugins/sudoers*

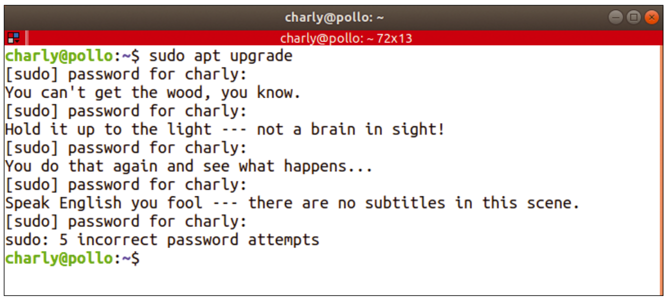
Author

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



```
GNU nano 2.9.3              /etc/sudoers.tmp

#
# This file MUST be edited with the 'visudo' command as root.
#
# Please consider adding local content in /etc/sudoers.d/ instead of
# directly modifying this file.
#
# See the man page for details on how to write a sudoers file.
#

Defaults        env_reset
Defaults        insults
Defaults        passwd_tries=5
Defaults        mail_badpass
Defaults        secure_path="/usr/local/sbin:/usr/local/bin:/usr/sbin:/$

^G Get Help   ^O Write Out  ^W Where Is   ^K Cut Text   ^J Justify
^X Exit       ^R Read File  ^\ Replace    ^U Uncut Text ^T To Spell
```

Figure 1: The file `/etc/sudoers` contains two new *Defaults* entries.



```
charly@pollo:~$ sudo apt upgrade
[sudo] password for charly:
You can't get the wood, you know.
[sudo] password for charly:
Hold it up to the light --- not a brain in sight!
[sudo] password for charly:
You do that again and see what happens...
[sudo] password for charly:
Speak English you fool --- there are no subtitles in this scene.
[sudo] password for charly:
sudo: 5 incorrect password attempts
charly@pollo:~$
```

Figure 2: Users are now hit by four insults before `sudo` black flags them.

Kraft helps small companies keep track of invoices and other docs

Essential Craft

If you're running a small company, and you can't seem to keep up on invoices, orders, and other business documents, time to have a look at Kraft. *By Markus Feilner*

After a few years of silence, 2018 has seen several releases of Kraft [1], an application for the Linux desktop that was designed to help small businesses create quotes, invoices, and other business documents. Kraft is a free, GPL-licensed application that runs on Linux and is built with Qt – thus, it works best on KDE desktop systems. And, as stated on the project's welcome page: You don't need a cloud. Kraft lets you keep control of your data and the data of your customers.

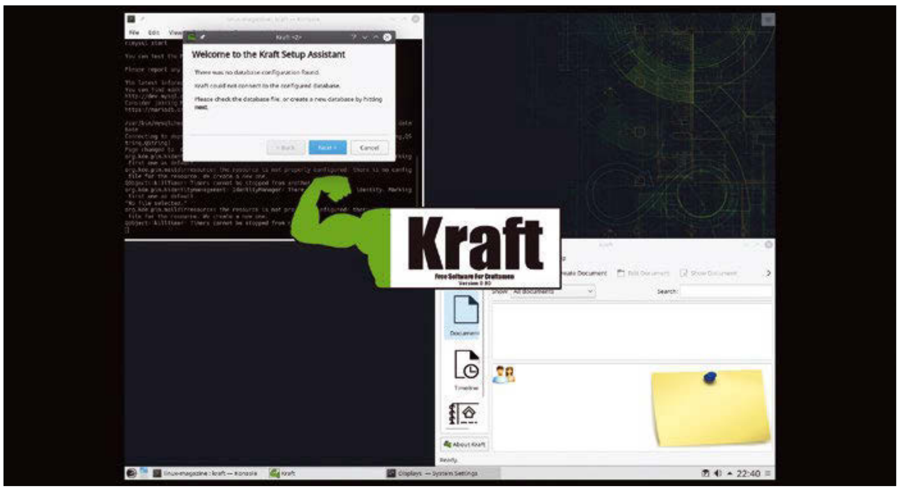
Kraft is highly customizable. It supports two database options, and many of its settings are easily configured through a settings dialog or config file. By predefining hourly rates, units, and wages, and by building a logical structure of the item templates, you can greatly reduce the non-billable hours you spend tending to accounting details. Output appears as a high-quality PDF that you can print on company stationery (paper) or send directly to the customer.

The Kraft project was initiated by the ownCloud developer and former SUSE manager Klaas Freitag in 2007. Since then, Kraft has seen lots of releases and updates. The developers took a long pause starting in 2015, and active work resumed again in 2018. The name *Kraft* is a pun on "craft" and the German word *kraft*, which means "physical strength" – hence the logo (Figure 1).

Freitag is still in charge of large parts of the work, but a growing community is there to help him, and some of the Kraft users have become his customers. See the box entitled "Interview with Kraft Founder Klaas Freitag."

Features

The Kraft project was designed with the assumption that the companies who are using Kraft also work with a tax consultant. Even though Kraft stores all documents, the assumption is that most users will have the docs stored on paper as well. When it comes to document creation or address management, Kraft does not attempt to reinvent the wheel. Wherever possible, Kraft depends on existing solutions, such as KDE's KAddressBook,


Welcome to the Kraft Setup Assistant

There was no database configuration found.

Kraft could not connect to the configured database.

Please check the database file, or create a new database by hitting next.

Figure 1: Kraft shows its powerful logo, along with the wizard and the main window.

Interview with Kraft Founder Klaas Freitag

Linux Magazine: Why and when did you initiate Kraft?

Klaas Freitag: I started Kraft around 2004. My brother, who owns a landscaping company, has used it since the early days, and beside that motivation, I think I started Kraft because this kind of software is lacking in the FOSS portfolio.

Before Kraft, there have been two other versions of comparable software, one on Atari ST and one based on Microsoft Access 2.0. Neither was open source. In 2006, I gave a presentation about Kraft on Akademy in Dublin.

LM: Do you know companies that actually use Kraft?

KF: Yes, I know around 10 companies that use it on a daily basis. It is handcrafters, farmers, and also freelancers in the IT area. A pretty big range, also geographically. There is even somebody in New Zealand working with Kraft.

LM: What are the coolest features?

KF: My favorite features are the templates, the great PDF output, [and] the tagging to support discounted items.

LM: What is missing; where do you need cooperation?

KF: It's a lot of details, small things that kill productivity. And since Kraft is for saving people time, the small details matter. Apart from that, the Kraft project is missing promotion. The target group of users is not too deep into Linux yet, and that has to change, because the Linux world has a lot to offer.

LM: Do you have a business model in mind? Support? Services?

KF: No business motivation from my side. There was once some consulting around Kraft, but I am not sure how active that is at the moment.

LM: How big is the community?

KF: Too small. Developers prefer to work on media players. 🙂

LM: Well, community size and software quality does not seem to be correlated – what were your guiding principles when you started the project? How can you develop a stable, functional open source application without a large community behind it?

KF: I do not know if this can be generalized, but five guiding principles led our development – and they are inspired by the needs of the customers in the small-medium business enterprise:

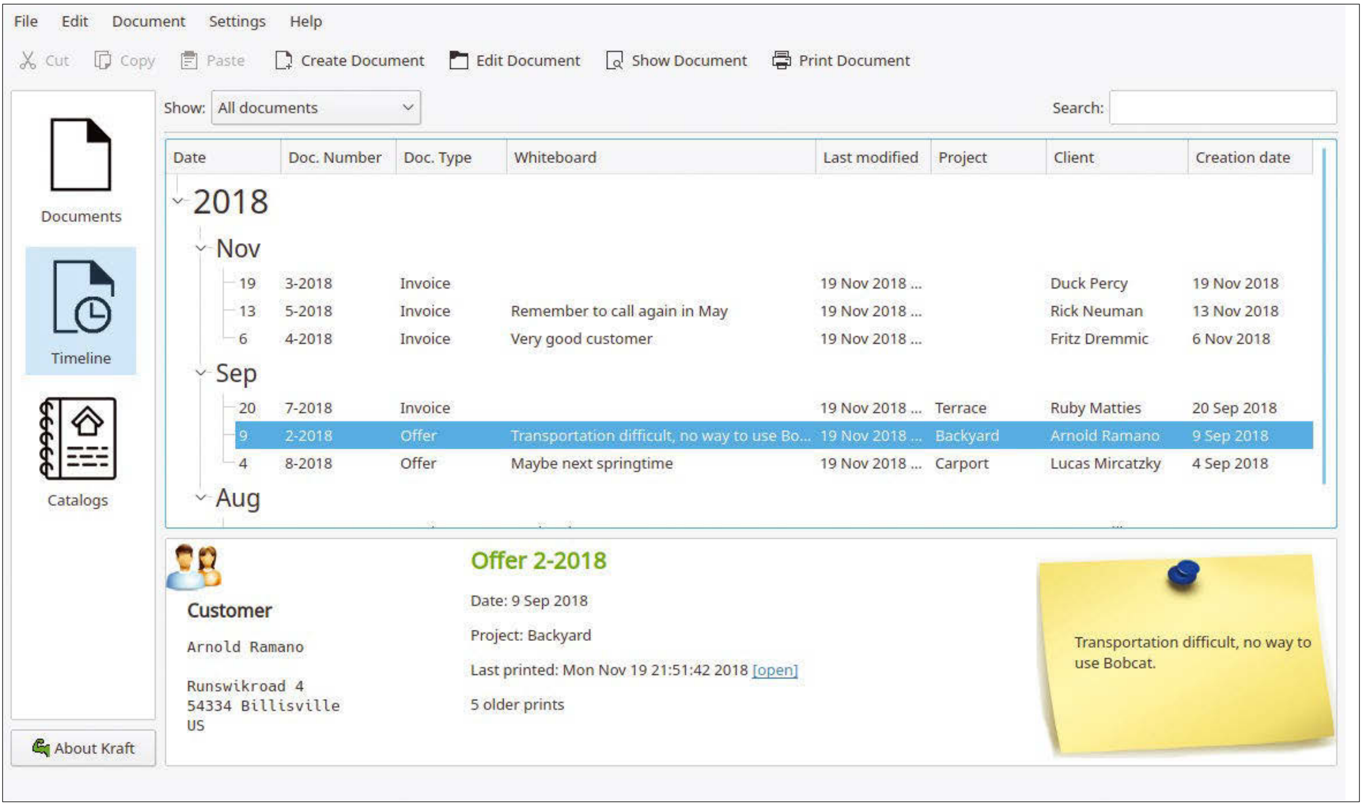
1) Do not try to provide all features one can think of; stay real; KISS.
2) Do not solve problems that are too big (Datev interface, complex XML formats for catalogs, etc.).
3) Do not expect the users to be interested in software in general or even in Linux or open source. Kraft is a tool they can use to give them more time for their profession and company. Nobody should have to deal with office programs.
4) Make the features that exist easy and intuitive to use.
5) Run on the users' computer to protect important data. If it runs in the cloud, it should only be a private cloud.

LM: What are your plans on future developments? If you could dream, what would you like to have?

KF: I would like to see some kind of management for down payments, partial payment, and installments, [and] then integration with Cloud – best in ownCloud. Also on my list are catalog sharing, publishing documents online, much more reporting, and, last but not least, more cross platform stuff, including a Mac OS X version.

LM: What is a question about Kraft?

KF: How can Kraft be better known and grow the community?



Date	Doc. Number	Doc. Type	Whiteboard	Last modified	Project	Client	Creation date
2018							
Nov							
19	3-2018	Invoice		19 Nov 2018 ...		Duck Percy	19 Nov 2018
13	5-2018	Invoice	Remember to call again in May	19 Nov 2018 ...		Rick Neuman	13 Nov 2018
6	4-2018	Invoice	Very good customer	19 Nov 2018 ...		Fritz Dremmic	6 Nov 2018
Sep							
20	7-2018	Invoice		19 Nov 2018 ...	Terrace	Ruby Matties	20 Sep 2018
9	2-2018	Offer	Transportation difficult, no way to use Bo...	19 Nov 2018 ...	Backyard	Arnold Ramano	9 Sep 2018
4	8-2018	Offer	Maybe next springtime	19 Nov 2018 ...	Carport	Lucas Mircatzky	4 Sep 2018
Aug							

Customer

```
Arnold Ramano

Runswikroad 4
54334 Billisville
US
```

Offer 2-2018

Date: 9 Sep 2018

Project: Backyard

Last printed: Mon Nov 19 21:51:42 2018 [open]

5 older prints

Transportation difficult, no way to use Bobcat.

Figure 2: Kraft displays a time line and detailed information about a recent document.

Listing 1: kraftrc

```
[MainWindow]
Height 1024=480
Height 1050=630
State=AAAA/wAAAAD9AAAAAAAAAA5IAAAIbAAAA
        BAAAAAQAAAAIAAAACPwAAAABAAAAAgA
        AAAIAAAAWAGOAYQBpAG4AVABvAG8AbA
        BCAGEAcgEAAAAA/////
wAAAAAAAAAAAAAAHgBkAG8AYwB1AGOAZQBuAHQ
  AVABvAG8AbABCAGEAcgEAAADe/////wAAAAA
  AAAAA
ToolBarsMovable=Disabled
Width 1280=728
Width 1400=914

[WindowSizes]
addressPickerSplitterSize=200,395
addressPickerTreeviewState=AAAA/
[...]
[userdefaults]
doctype=Acceptance of Order
```

which makes it easy to have Google, ownCloud/Nextcloud address books attached without the need for address book adapters. The address management interface has recently been redesigned to make it easier to integrate with other sources, like Mac OS X.

Kraft keeps a plain list of documents in its main view, with the most recent ones on top of the list. If you want to know how much business happened in a certain year or month, just click on the month or year in the tree view (Figure 2). Clicking on a document in the list will reveal the most relevant details about a document.

Installation

If you are running openSUSE, a simple `zypper in kraft` will install Kraft, at least for those who agree to install its dependencies. On an up-to-date Tumbleweed system, the dependencies are:

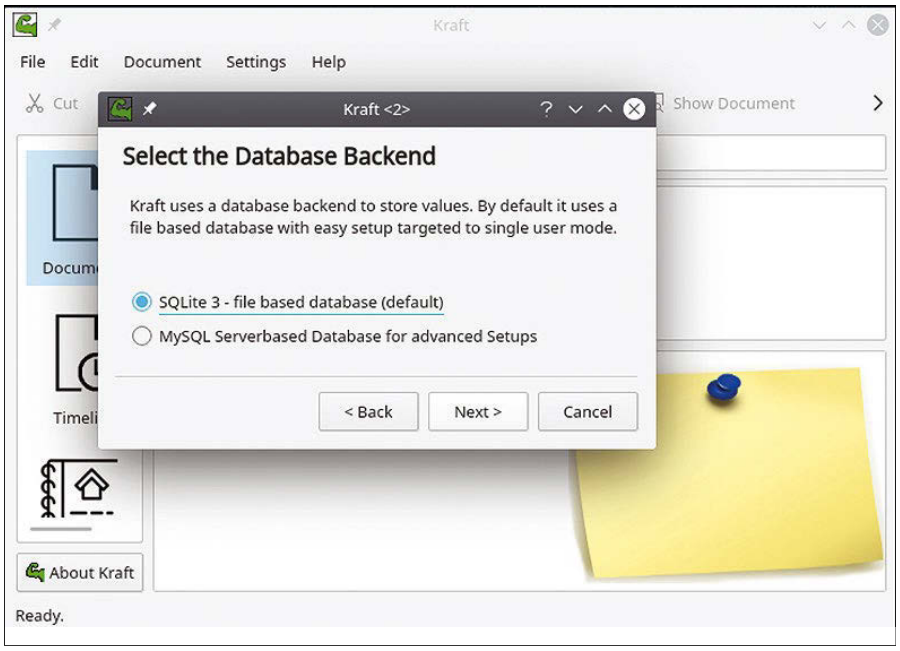
```
kraft
libctemplate3
python2-libxml2-python
python2-olefile
python2-Pillow
python2-PyPDF2
python2-reportlab
```

Kraft is also packaged for Mageia and Ubuntu and available for Arch Linux. Everybody else should be served with the AppImage from the website or by building the sources.

After installation, a simple `kraft` will start the program – and a wizard (Figure 3) will guide you through the setup process. All local data will be stored in `$HOME/.local/share/kraft`, and a config file called `kraftrc` in your local `.config` directory will look like the file in Listing 1. Remember: Kraft is highly integrated in your desktop; thus, if you end up with euros as a currency for your installation, you can wish to change to another currency; you can make the change through your desktop settings – in my case, the control center of my KDE system.

If you want to use a MySQL database as back end for Kraft, all you have to do is log in as root on your `mysqld` (`mysql -u root`) and type `CREATE DATABASE kraft;`. All you need to do now is hand over the database server's address or name plus login credentials to the setup wizard, and there you go.

Some changes and new features may require changes in the `kraftrc` file. For instance, you can configure a different mail client for the "send invoice" function of Kraft. (KDE Kontact/KMail is the default.) If you prefer Thunderbird or another mail client, just add `mailUA=xdg` to the `[system]` section. If the `[system]` section doesn't exist, you will have to create it.



Select the Database Backend

Kraft uses a database backend to store values. By default it uses a file based database with easy setup targeted to single user mode.

[✓ SQLite 3 - file based database (default)]
MySQL Serverbased Database for advanced Setups

< Back | Next > | Cancel

Figure 3: A wizard guides you through your initial setup. Don't worry – you can revert any changes.

Starting Kraft

When you start Kraft for the first time, a wizard guides you through the setup process. The wizard is simple, clear, and focused – and since the GUI also offers a `Reset` function, first time users can play around without risk.

The wizard picks up and sets up the database, tests it, asks for a schema (if needed), and takes on the address book integration.

To create a document in Kraft, the user works in a simple editor that has entries for the document header, the items, and the document footer. Users do not need to care about the layout and can concentrate on the document contents. Kraft does the rest and even enables document workflows from offer to invoice with several steps in between.

The Business Workflow

Many business processes require a predicable sequence of documents. Customers often demand a quote and maybe a second quote after negotiation. If the quote turns into an order, an acceptance of the order should be sent, and finally, after the order is filled, an invoice is written. Kraft leverages information in earlier documents to simplify later documents: Once a quote exists, Kraft will use the information for later documents (such as acceptance of order or invoice). Once all the information is entered, all items in the quote are cop-

Listing 2: Template

```
<pageTemplate id="first">
    <pageGraphics>
        <setFont name="Times-Roman" size="12" />
        <lineMode width="0.5" />
        <lines>2.41cm 2.2cm 19cm 2.2cm</lines>
        <fill color="darkgreen" />
        <drawCentredString x="105mm" y="2.3cm">{{MY_NAME}} {{MY_ORGANISATION}}</drawCentredString>

        <lines>7mm 19cm 12mm 19cm</lines>
        <lines>2mm 14.65cm 12mm 14.65cm</lines>

        <fill color="black" />
        <setFont name="Times-Roman" size="8" />
        <drawString x="25mm" y="24.7cm">{{MY_ORGANISATION}} - {{MY_STREET}} - {{MY_POSTCODE}} {{MY_LOCALITY}}</drawString>
        <lines>25mm 24.6cm 11cm 24.6cm</lines>
        <setFont name="Times-Roman" size="10" />
        <drawCentredString x="105mm" y="1.8cm">{{MY_STREET}} - {{MY_POSTCODE}} {{MY_LOCALITY}}</drawCentredString>
        <drawCentredString x="105mm" y="1.4cm">Telephone {{MY_PHONE}} - Fax {{MY_FAX}}</drawCentredString>
        <drawCentredString x="185mm" y="1.4cm">Page <pageNumber/></drawCentredString>
        <image x="140mm" y="22cm" width="6cm" height="6cm" file="/home/me/kraft/logo.png" />
    </pageGraphics>
    <frame id="address"  x1="2.41cm" y1="20.62cm" width="8.5cm" height="4cm"/>
    <frame id="info"     x1="12cm" y1="20.62cm" width="7.41cm" height="1.5cm"/>
    <frame id="subject"  x1="2.41cm" y1="18.2cm" width="17cm"  height="1.2cm"/>
    <frame id="detail"   x1="2.41cm" y1="3cm"  width="17cm"  height="15cm"/>
</pageTemplate>
```

Listing 3: Signature

```
<condPageBreak height="1cm"/> {{POSTTEXT}} <spacer length="0.3cm" width="1mm"/> <para style="text"> {{GOODBYE}} </para>
 <illustration width="175" height="33"> <image x="15" y="0" width="270" height="30" file="/home/me/kraft/signature.png" /> </
 illustration> <nextFrame/>
```

ied to the subsequent document, so creating an invoice from a quote is as easy as one copy and a quick check.

Kraft includes several useful document templates that are populated automatically from the address book. You are also free to create custom templates.

Kraft's calculation module allows the user to build a structured catalog of templates, including texts, units, and price calculations. The calculation module will consider time calculation based on different hour rates, fixed calculation parts, and also material that is used in calculations.

Tagging and Numbering

Documents like invoices often need a unique number for accounting and tax purposes. Kraft supports so-called number cycles for each document type. Kraft users can use the default incremental format or configure their own numbering format to include extra info, such as static text or the week or month number, in addition to a required unique counter.

Another feature is the ability to tag certain items in the document. For example, you could tag an item in an invoice as *material* or *labor* to allow for different tax or discount rates.

Your Own Logo

Kraft lets you add a logo to every document you send out. Templates [2] are stored in RML, an XML dialect. Listing 2 shows an excerpt from a template file, with the entries `pageTemplate` and `pageGraphics` adapted to show an individual logo. Listing 3 shows how to embed the signature of the company lead.

Conclusion

If you are part of a small business or work as an independent contractor, and you find yourself behind on billing and bidding because the accounting details are *too much trouble*, check out Kraft. ■■■

Info

[1] Kraft: *http://volle-kraft-voraus.de*

[2] List of template variables in Kraft: *http://volle-kraft-voraus.de/Main/TemplateVariables*

Author

Markus Feilner is a seasoned Linux expert from Regensburg, Germany. He has been working with free and open source software since 1994 as a trainer, consultant, author, and journalist. He is currently employed as Team Lead of the SUSE Documentation Team in Nuremberg, Germany.

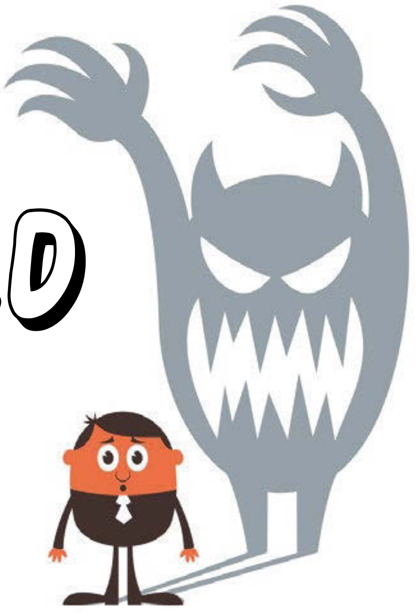


Image processing with Go

SHADOW WORLD

Go comes with an image-processing toolkit right out of the box. In this month's column, Mike Schilli explains how to walk through a photo's pixels to detect the foreground by comparing values against a threshold and shows how to manipulate the original by creating a nice looking silhouette. *By Mike Schilli*

Until recently, my headshot at the end of every "Programming Snapshot" article showed a much younger version of myself, dating back 15 years, so I grudgingly shot a new one the other day. While doing this, the idea occurred to me to try out my new favorite language Go's suitability for image processing. How hard could it be to generate an artful silhouette of the person shown in the photo?

Of course, with some Gimp skills this could be done quite quickly. However, what is far more interesting is the question of how an image processing program walks through the pixels of Figure 1 and finds out which of them actually belong to the person (the foreground) and which to the lighter background. When a foreground pixel is found, the algorithm can then go ahead and set it to black, which in the digital world means it has its red, green, and blue channels set to `0,0,0`. Easy enough, right?

Finding the Foreground

With a light background and a significantly darker object in the foreground, the program in Listing 1 [1] simply finds all pixels whose brightness lies below a previously defined threshold value and blacks them out completely. The `Darken()` function accepts a `draw.Image` type structure from line 9, including the width and height of the image in the `width` and `height` parameters in pixels.

The double `for` loop starting on line 11 runs through the pixels line by line from top to bottom, and the `At()` function called in Line 15 returns the color channels of the current pixel as a value of type `color.Color`, which the `RGBA()` function then converts to a red, green, and blue value and an alpha value (transparency). The latter isn't used by the algorithm; hence, line 14 tells Go to ignore it by providing an underscore in lieu of a variable name to which to assign it.

The upper eight bits of these returned values indicate the color value of the RGB channel from `0` to `255`; a bit-shift operation extracts the relevant bits and compares the outcome with the experimentally determined threshold value of `180`. If one of the channels lies below this value (i.e., the current pixel is darker), line 20 sets the pixel value to

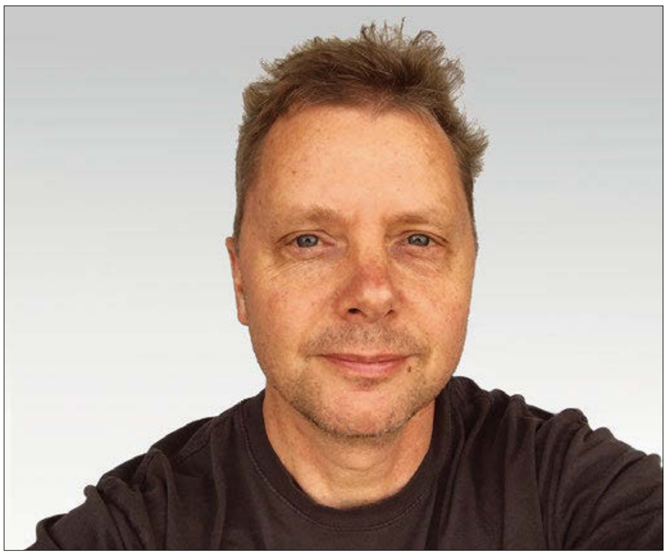


Figure 1: The original version of my headshot fed to the algorithm.

Listing 1: darkenthreshold.go

```
01 package darkenthreshold
02 import (
03   "image/color"
04   "image/draw"
05 )
06
07 var Threshold uint8 = 180
08
09 func Darken(dimg draw.Image,
10              width int, height int) {
11   for x := 0; x < width; x++ {
12     for y := 0; y < height; y++ {
13
14       red, green, blue, _ :=
15         dimg.At(x, y).RGBA()
16
17       if uint8(red >> 8) < Threshold ||
18          uint8(blue >> 8) < Threshold ||
19          uint8(green >> 8) < Threshold {
20            dimg.Set(x, y, color.Black)
21       }
22     }
23   }
24 }
```

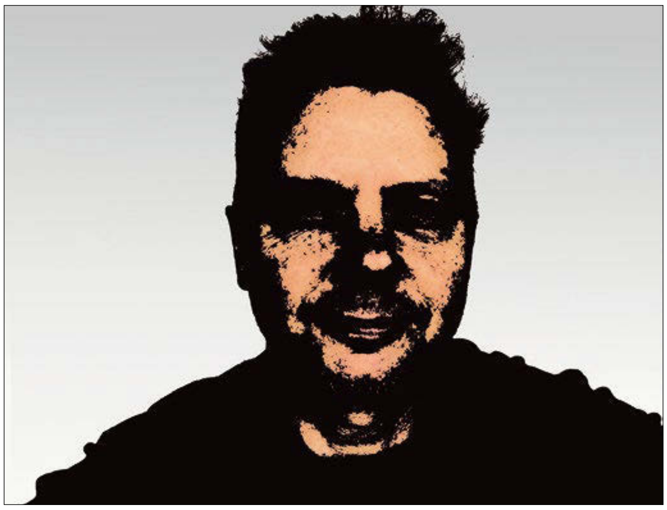


Figure 2: Too low a threshold for blackening gives the image a Warhol-like appeal but does not produce a silhouette.

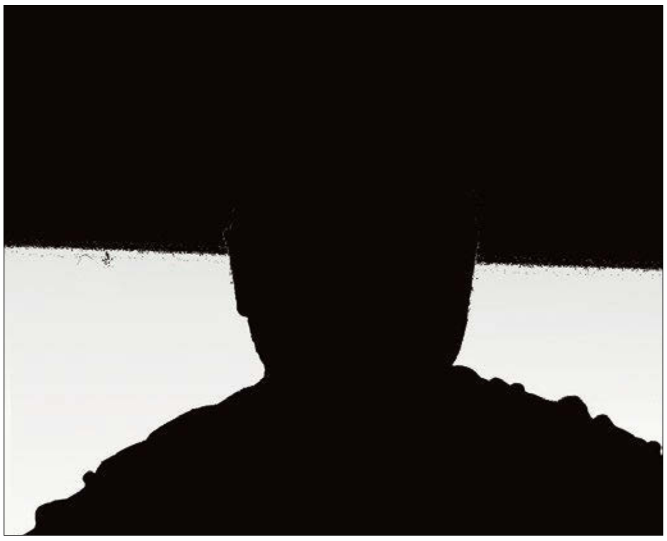


Figure 3: A threshold value that is too high, however, also selects parts of the background.

`color.Black`, that is, `(0, 0, 0)`. The threshold value setting is the crux of the algorithm. If the value is set too low, the procedure does not find all foreground pixels (Figure 2); if it is too high, it blacks the image in places that belong to the background (Figure 3).

Fancy Logging

The main program, which accepts the name of an image file in JPG format, is shown in Listing 2. In order to tell the user how to use the program, the standard `flags` module analyzes the command line, grabs any given flags (e.g., `-v`), and provides them – along with all command-line arguments – in the global `flag` variable after `flag.Parse()` was called in line 24. The image file is then found in `flag.Arg(0)`; if it is missing, line 30 calls the `usage()` function defined in line 16, which displays the correct syntax for the command to the user and aborts the program.

For some user entertainment and general help to figure out what's going on, the main program uses Google's log package, `glog`, which gets imported in line 10. It is pure witchcraft, as it communicates behind the scenes with the `flag` package's command-line parser and also injects a help page explaining the `glog` command-line parameters, which

Listing 2: thresmain.go

```
01 package main
02
03 import (
04   "darkenthreshold"
05   "flag"
06   "fmt"
07   "image"
08   "image/draw"
09   "image/jpeg"
10   "github.com/golang/glog"
11   "os"
12   "path/filepath"
13   "strings"
14 )
15
16 func usage() {
17   fmt.Fprintf(os.Stderr, "usage: " +
18     os.Args[0]+" image.jpg\n")
19   flag.PrintDefaults()
20   os.Exit(2)
21 }
22
23 func main() {
24   flag.Parse()
25   flag.Usage = usage
26
27   defer glog.Flush()
28
29   if len(flag.Args()) != 1 {
30     usage()
31   }
32
33   srcFileName := flag.Arg(0)
34
35   src, err := os.Open(srcFileName)
36   if err != nil {
37     glog.Fatalf("Can't read %s: %s",
38                 srcFileName, err)
39   }
40
41   glog.Infof("Decoding %s\n", srcFileName)
42
43   jimg, err := jpeg.Decode(src)
44   if err != nil {
45     glog.Fatalf("Can't decode %s: %s",
46                 srcFileName, err)
47   }
48
```

Listing 2: thresmain.go (continued)

```
49   bounds := jimg.Bounds()
50   width, height := bounds.Max.X,
51                    bounds.Max.Y
52
53   dimg := image.NewRGBA(bounds)
54   draw.Draw(dimg, dimg.Bounds(), jimg,
55             bounds.Min, draw.Src)
56
57   darkenthreshold.Darken(dimg,
58                          width, height)
59
60   fileSuffix := filepath.Ext(srcFileName)
61   fileBase := strings.TrimSuffix(srcFileName,
62                                  fileSuffix)
63   dstFileName := fmt.Sprintf("%s-s%s",
64                    fileBase, fileSuffix)
65
66   dstFile, err := os.OpenFile(dstFileName,
67                  os.O_RDWR|os.O_CREATE, 0644)
68   if err != nil {
69     glog.Fatalf("Can't open output")
70   }
71
72   jpeg.Encode(dstFile, dimg,
73               &jpeg.Options{Quality: 80})
74   dstFile.Close()
75 }
```

`flag` will display when called incorrectly (Figure 4). Listing 2 also reports the name of the currently decoded JPG file in line 41 for information purposes.

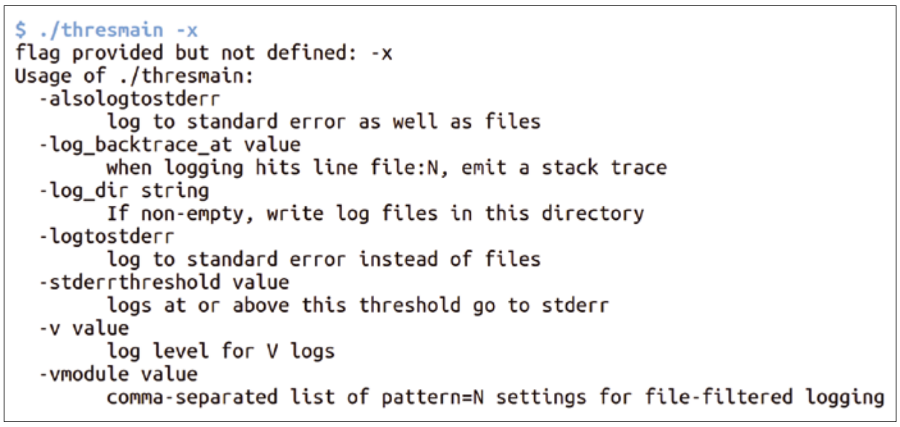
But to where does `glog` actually log? If the command-line parameter `--stderrthresold=INFO`, which redirects all `glog` messages marked as "Info" to `stderr`, is omitted, the log messages can be found in a logfile in your computer's `/tmp` directory. Figure 5 shows its names. The `Flush` method is also important; Listing 2 calls it in line 27 with the `defer` keyword, which means it kicks in at the end of the current block that belongs to the main program. Note that if you forget to call `Flush()`, you will be left wondering why nothing gets logged at all or some entries are missing from the logfile.

To install `glog` in your Go path, from where the main program can then grab it at compile time, just call:

```
go get github.com/golang/glog
```

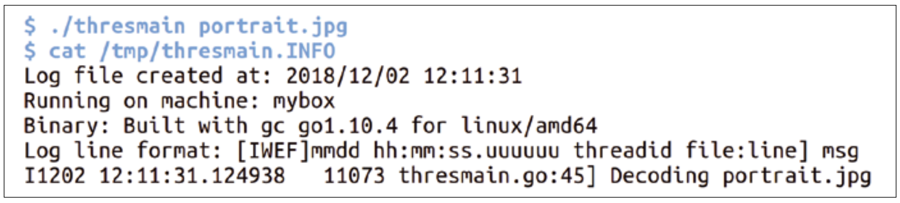
Future programs will have access to it to compile and link against.

Opening a JPG file and extracting its pixel values is not witchcraft thanks to the standard `image/jpeg` package that comes with Go. The `jpeg.Decode()` function grabs a `Reader` interface for the image file, as previously generated by `os.Open()`, and decodes the compressed data. This fails with corrupt files; line 44 thus checks the results and aborts the program with `glog.Fatalf()` from Go's logging module if anything smells fishy.



```
$ ./thresmain -x
flag provided but not defined: -x
Usage of ./thresmain:
  -alsologtostderr
        log to standard error as well as files
  -log_backtrace_at value
        when logging hits line file:N, emit a stack trace
  -log_dir string
        If non-empty, write log files in this directory
  -logtostderr
        log to standard error instead of files
  -stderrthreshold value
        logs at or above this threshold go to stderr
  -v value
        log level for V logs
  -vmodule value
        comma-separated list of pattern=N settings for file-filtered logging
```

Figure 4: Invoked with invalid parameters, the program prints out the permissible parameters of the logging module and aborts.



```
$ ./thresmain portrait.jpg
$ cat /tmp/thresmain.INFO
Log file created at: 2018/12/02 12:11:31
Running on machine: mybox
Binary: Built with gc go1.10.4 for linux/amd64
Log line format: [IWEF]mmdd hh:mm:ss.uuuuuu threadid file:line] msg
I1202 12:11:31.124938   11073 thresmain.go:45] Decoding portrait.jpg
```

Figure 5: `glog` writes messages to a logfile.

Juggling Internal Formats

However, the image data read by `jpeg.Decode()` is not yet available in a format that can be changed dynamically. This is why Listing 2 calls the `draw.Draw()` function from the `image/draw` package to copy the image data in line 54 into a newly created structure of the type `image.RGBA`. Going forward, Listing 1 can read from this data using `At()`and also change pixel values with `Set()`. As described in the man page that comes with the package [2], the interface in `image/draw` aims to warp and transform the processed image in fancy ways, but we're only performing simple pixel `getter` and `setter` functions.

The `draw.Draw()` function gets called on line 54, which in this case only makes the internal format writable and converts the JPG image in `jimg` to the draw image `dimg`. The constant `draw.Src` indicates that the target image (which in this case is empty because it was just created) is overwritten in `dimg`. In contrast, a value of `draw.Over` would have performed a source-over-destination-overlay transformation instead. The given start coordinates in `bounds.Min` define the coordinates of the image origin `(0,0)`, because we're not interested in cropping but instead want to copy the whole image.

All New

To compile the helper library in Listing 1, which is later needed by the main program in Listing 2, move it to the directory src/darkenthreshold under your Go path (usually ~/go) and run go install there.

To compile the main program, call go build thresmain.go, and the resulting binary thresmain will be writing any modified image file (e.g., portrait.jpg) to a new file with a name sporting an -s suffix, as in portrait-s.jpg. To rewrite the filename, Line 60 in Listing 2 extracts the .jpg extension from the name of the old file and then chops it off with TrimSuffix, before the Sprintf function then inserts a -s from the fmt package and puts the suffix back. The target file won't exist yet at this point, so the OpenFile() function in line 66 needs the read/write/create flags (os.O_RDWR and os.O_CREATE) to create a new file.

With jpeg.Encode() and a quality value of 80, Listing 2 encodes the modified data into JPG format and writes the result to the specified file (Figure 6). If you like, you can try other algorithms, such as the Flood Fill [3] method, which colorizes pixels that are similar in value. After all, the raw pixel data is at your fingertips, and there are no limits as to what you can accomplish with them! ■■■

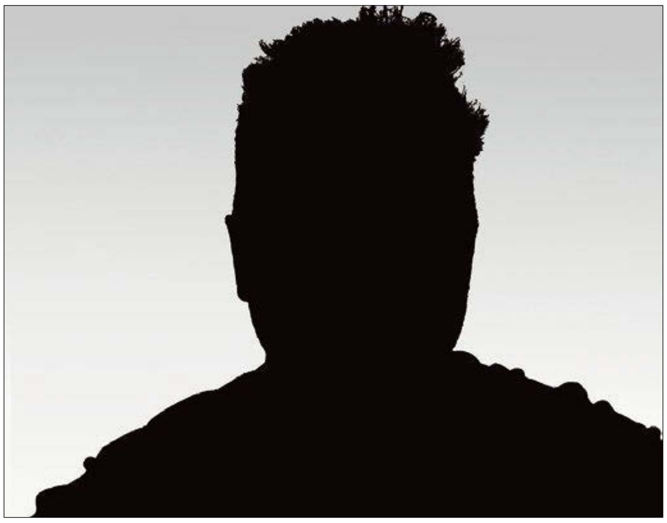


Figure 6: With a suitable threshold value, the algorithm performs the task as expected.

Info

[1] Listings for this article: *ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/221/*

[2] The Go image/draw package: *https://blog.golang.org/go-imagedraw-package*

[3] Flood Fill: *https://en.wikipedia.org/wiki/Flood_fill*

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.

MakerSpace

Open source messaging middleware

IoT with RabbitMQ

Connect multiple protocols and servers together on your IoT projects. *By Pete Metcalfe*

For Internet of Things (IoT) projects, there are a lot of different ways that sensors, devices, and client interfaces can be connected together. For many projects, using a simple Message Queue Telemetry Transport (MQTT) broker is all that you need. However, if you're trying to merge and build IoT projects that use both MQTT and Advanced Message Queue Protocol (AMQP), or you require a REST API, then you should take a look at RabbitMQ.

RabbitMQ [1] is an open source middleware solution that natively uses AMQP communications, but it has a good selection of plugins to support features like MQTT, MQTT WebSockets, HTTP REST API, and server-to-server communications (Figure 1).

In this article, I will set up a RabbitMQ server, and I will look at some of the differences between MQTT and AMQP messaging. Finally, an example of an Arduino MQTT message will be presented as both an MQTT and an AMQP item in a Node-RED dashboard.

Getting Started Locally

RabbitMQ can be installed on Windows, Linux, and macOS systems, and there are also some cloud-based offerings. For small systems, lower-end hardware like a Raspberry Pi can be used. Complete RabbitMQ installation instructions are available online [2]. To install and run RabbitMQ on an Ubuntu system, enter:

```
sudo apt-get update
sudo apt-get install rabbitmq-server
sudo service rabbitmq-server start
```

The next step is to add some plugins. For my project, I loaded the MQTT and web administration plugins:

```
sudo rabbitmq-plugins ⮐
  enable rabbitmq_mqtt
sudo rabbitmq-plugins ⮐
  enable rabbitmq-management
```

The `rabbitmqctl` command-line tool allows you to configure and review the RabbitMQ server. To add a user (*admin1*) with a password (*admin1*) that has config, write, and read rights for management and administrator access, enter:

```
sudo rabbitmqctl add_user admin1 admin1
sudo rabbitmqctl set_permissions ⮐
  -p / admin1 ".*" ".*" ".*"
sudo rabbitmqctl set_user_tags admin1 ⮐
  management administrator
```

After you've defined an administrative user, the RabbitMQ web management plugin can be accessed at *http://ip_address:15672* (Figure 2).

The RabbitMQ web management tool offers an overview of the present system load, connections, exchanges, and queues.

The RabbitMQ web management tool is excellent for small manual changes. However if you are looking at a doing a

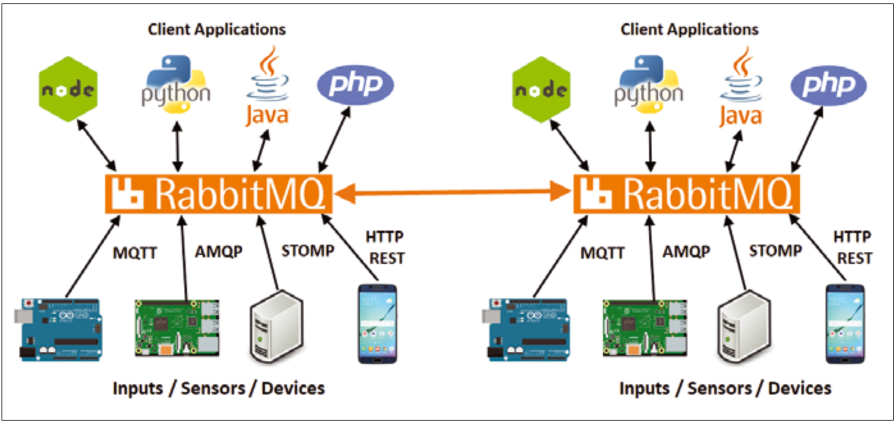


Figure 1: RabbitMQ overview.

Lead Image © Tatiana Tushyna, 123RF.com

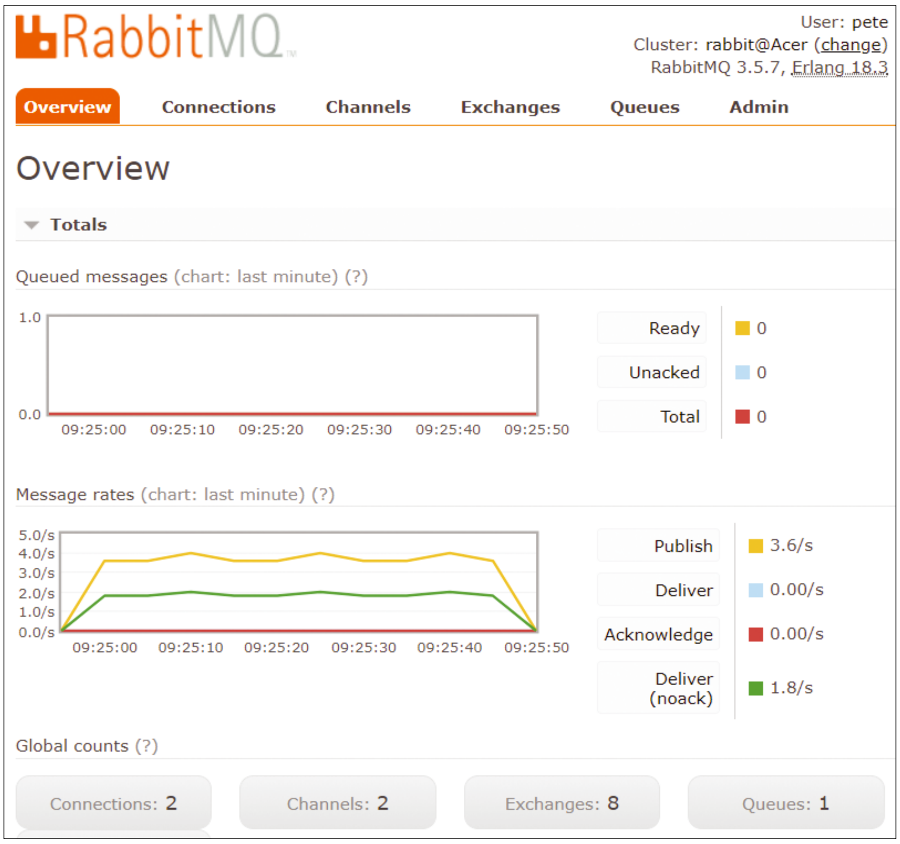
Author

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

large number of additions or changes, then `rabbitmqadmin`, you can use the command-line management tool. Install the tool by entering:

```
# Get the cli and make it available ↩
  to use.
wget http://127.0.0.1:15672/cli/↩
  rabbitmqadmin
```

```
sudo chmod +x rabbitmqadmin
```



Ready 🟨 0
Unacked 🟦 0
Total 🟥 0

Publish 🟨 3.6/s
Deliver 🟦 0.00/s
Acknowledge 🟥 0.00/s
Deliver (noack) 🟩 1.8/s

Connections: 2 Channels: 2 Exchanges: 8 Queues: 1

Figure 2: RabbitMQ web administration.

Listing 1: Python AMQP Publish Example

```python
#!/usr/bin/env python
import pika

node = "192.168.0.121"
user = "pete"
pwd = "pete"

# Connect to a remote AMQP server with a username/password
credentials = pika.PlainCredentials(user, pwd)
connection = pika.BlockingConnection(pika.ConnectionParameters(node,
        5672, '/', credentials))
channel = connection.channel()

# Create a queue if it doesn't already exist
channel.queue_declare(queue='Rasp_1',durable=True)

# Define the properties and publish a message
props = pika.BasicProperties(
    headers= {'status': 'Good Quality',"alarm":"HI"},
    type ="Pi Sensor")
channel.basic_publish(exchange='',
    routing_key='Rasp_1',body='99.5', properties = props)

connection.close()
```

Comparing MQTT and AMQP

It's useful to comment about some of the differences between MQTT and AMQP.

MQTT is a lightweight publish- and subscribe-based messaging protocol that works well with lower-end hardware and limited bandwidth. For Arduino-type applications where you only need to pass some sensor data, MQTT is an excellent fit.

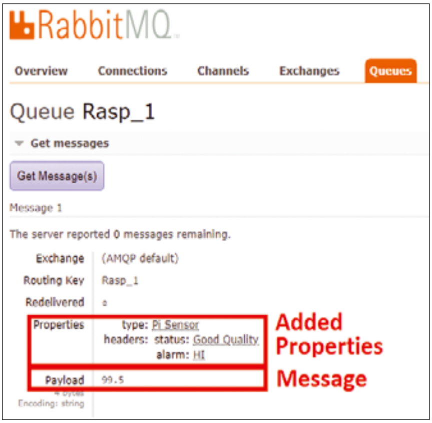
AMQP has more overhead than MQTT, because it is a more advanced protocol that includes message orientation, queuing, routing, reliability, and security. Presently, there are no mainstream AMQP Arduino libraries, but numerous programming options for Raspberry Pi, Linux, Windows, and macOS systems exist. An AMQP IoT example would be to send sensor failures and alarms to dedicated maintenance and alarm queues.

MQTT and AMQP Queues

One of the biggest differences in queues is that MQTT queues are designed to show you the last available message, where as AMQP will store multiple messages in a queue.

A published MQTT message contains a message body, a retain flag, and a quality of service (QoS) value.

An AMQP message can be published with added properties, such as time stamp, type of message, and expiration information. AMQP messages also support the addition of custom header values. Listing 1 is a Python publish example that defines the message type to be `"Pi Sensor"`, and it includes custom headers for status and alarm state.



Queue Rasp_1

Message 1

The server reported 0 messages remaining.

Exchange	(AMQP default)
Routing Key	Rasp_1
Redelivered	○
Properties	type: Pi Sensor headers: status: Good Quality alarm: HI
Payload 4 bytes Encoding: string	99.5

Figure 3: AMQP queue with custom properties.

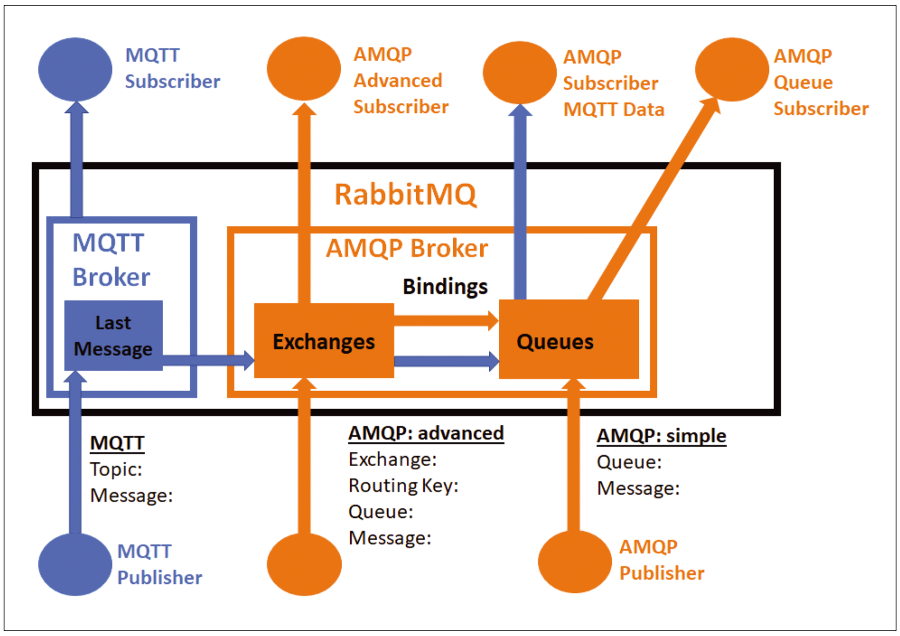


Figure 4: RabbitMQ messaging overview.

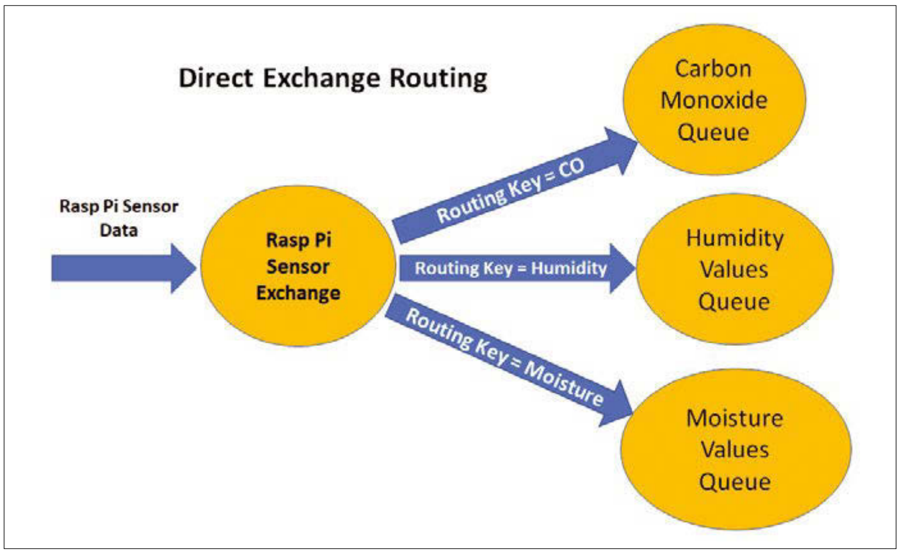


Figure 5: RabbitMQ direct exchange routing.

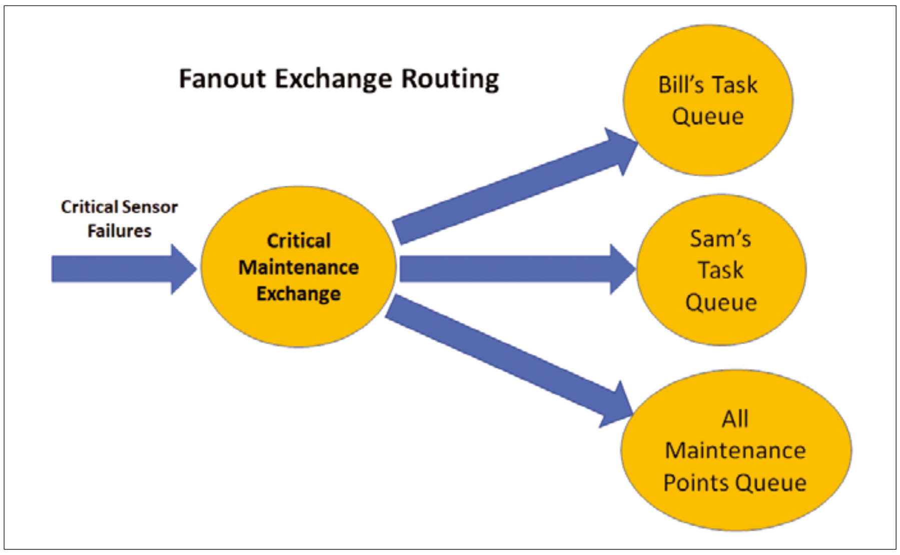


Figure 6: RabbitMQ fanout exchange routing.

The results from the Listing 1 example can be examined using the *Queue* | *Get Message* option in the RabbitMQ web management interface (Figure 3).

RabbitMQ Messaging

There are a variety of ways to which AMQP messages can be published and subscribed to (Figure 4). The simplest way is to create a queue, and then messages can be published and subscribed to from that queue.

To help with the distribution and filtering of messages, AMQP supports a number of different exchange types. Messages in an exchange use bindings based on a routing key to link them to a queue.

The main types of exchanges are direct, fanout, headers, and topic. An IoT example of a direct exchange would be if a group of Raspberry Pi sensor values were going into a Rasp Pi sensor exchange. When the Rasp Pi publishes a sensor result to the exchange, the message also includes a routing key to link the message to the correct queue (Figure 5).

An IoT example of a fanout exchange would be critical sensor failures. A sensor failure message is sent to Bill and Sam's work queues and the All Maintenance Points queue all at the same time (Figure 6).

Connecting MQTT

After the MQTT plugin is installed, RabbitMQ can act as a standalone MQTT broker.

For an MQTT project, any ESP8266-supported Arduino hardware can be used. There are a number of MQTT Arduino libraries that are available. For this project, I used the *PubSubClient* library that can be installed using the Arduino Library Manager.

As a test project, I used a low cost MQ-2 Smoke Gas Sensor ($3) [3] that measures a combination of LPG, alcohol, propane, hydrogen, CO, and even methane. Note to fully use this sensor, some calibration is required. On the MQ-2 sensor, the analog signal is connected to Arduino pin A0 and the `analogRead(thePin)` function is used to read the sensor value.

Listing 2 is an Arduino example that reads the MQ-2 sensor and publishes the result to the RabbitMQ MQTT broker with a topic name of `mq2_mqtt`.

Listing 2: arduino_2_mqtt.ino

```
#include <ESP8266WiFi.h>
#include <PubSubClient.h>

// Update these with values suitable for your network.
const char* ssid = "your_ssid";
const char* password = "your_password";
const char* mqtt_server = "192.168.0.121";
const char* mqtt_user = "admin1";
const char* mqtt_pass= "admin1";

const int mq2pin = A0; //the MQ2 analog input pin

WiFiClient espClient;
PubSubClient client(espClient);

void setup_wifi() {
  // Connecting to a WiFi network
  WiFi.begin(ssid, password);
  while (WiFi.status() != WL_CONNECTED) {
    delay(500);
    Serial.print(".");
  }
  Serial.println("WiFi connected");
  Serial.println("IP address: ");
  Serial.println(WiFi.localIP());
}

void reconnect() {
  // Loop until we're reconnected
  Serial.println("In reconnect...");
  while (!client.connected()) {
    Serial.print("Attempting MQTT connection...");
    // Attempt to connect
    if (client.connect("Arduino_Gas", mqtt_user, mqtt_pass)) {
      Serial.println("connected");
    } else {
      Serial.print("failed, rc=");
      Serial.print(client.state());
      Serial.println(" try again in 5 seconds");
      delay(5000);
    }
  }
}

void setup() {
  Serial.begin(9600);
  setup_wifi();
  client.setServer(mqtt_server, 1883);
}

void loop() {
  char msg[8];
  if (!client.connected()) {
    reconnect();
  }

  sprintf(msg,"%i",analogRead(mq2pin));
  client.publish("mq2_mqtt", msg);
  Serial.print("MQ2 gas value:");
  Serial.println(msg);

  delay(5000);
}@L:
```

Once the MQTT value is published, any MQTT client can subscribe to it. Listing 3 is a Python MQTT subscribe example.

Listing 3: mqtt_sub.py

```
import paho.mqtt.client as mqtt

def on_message(client, userdata, message):
    print ("Message received: "  + message.payload)

client = mqtt.Client()
client.username_pw_set("admin1", password='admin1')
client.connect("192.168.0.121", 1883)

client.on_message = on_message       #attach function to callback

client.subscribe("mq2_mqtt")
client.loop_forever()                 #start the loop
```

Read MQTT Messages Using AMQP

MQTT clients can subscribe to MQTT messages directly, or it's possible to configure RabbitMQ to have the MQTT data accessible using AMQP. The routing of MQTT messages to AMQP queues is done using the *amp.topic* direct exchange (Figure 7).

To configure RabbitMQ to forward MQTT, the following steps are required:

1. Create a new RabbitMQ queue.
2. Create a binding between the MQTT exchange and the queue.

These two steps can be done in a number of ways: manually, in the RabbitMQ config file, using the `rabbitmqadmin` command-line tool, or via a program. To use `rabbitmqadmin` enter:

Listing 4: Checking the Queue for Messages

```
./rabbitmqadmin get queue=mq2_amqp
+-------------+----------+--------------+---------+
| routing_key | exchange | message_count | payload |
+-------------+----------+--------------+---------+
| mq2_mqtt    | amq.topic | 77           | 157     |
+-------------+----------+--------------+---------+
```

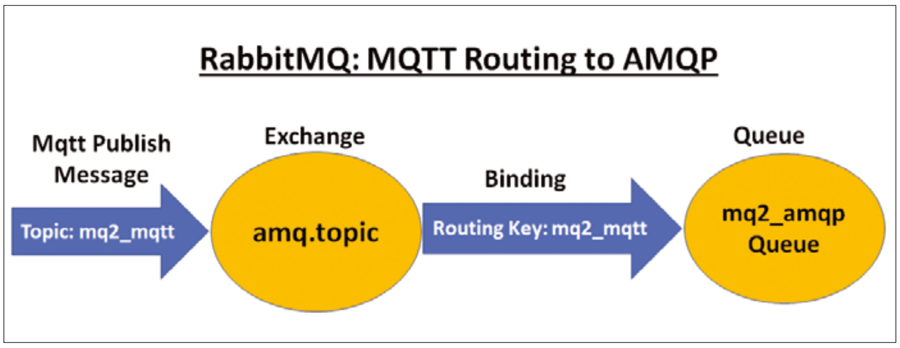


Figure 7: Routing MQTT messages to AMQP queues.

```
./rabbitmqadmin declare ⮐
  queue name=mq2_amqp durable=true
./rabbitmqadmin declare binding ⮐
  source=amq.topic \
  destination_type=queue ⮐
  destination=mq2_amqp ⮐
  routing_key=mq2_mqtt
```

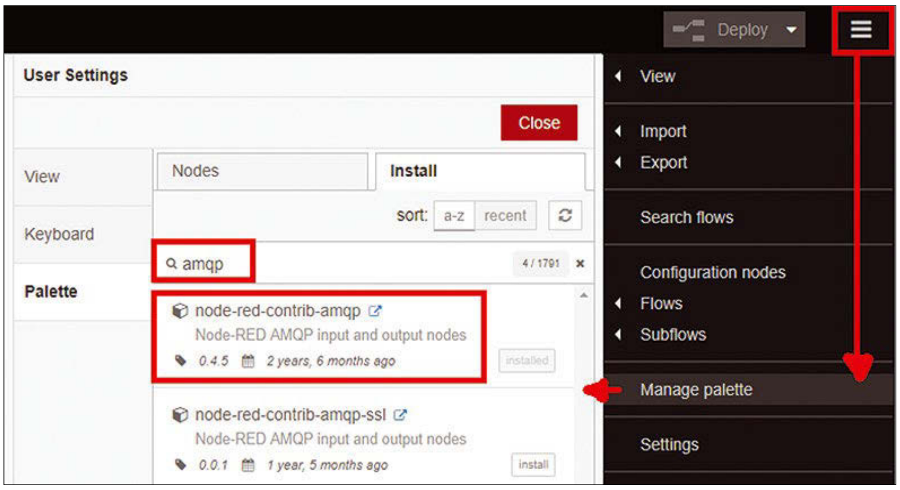
`rabbitmqadmin` can also be used to check the queue for messages (Listing 4).

Node-RED Dashboard

Node-RED [4] is a visual programming environment that allows users to create applications by dragging and dropping nodes on the screen. Logic flows are then created by connecting the different nodes together.

Node-RED has been preinstalled on Raspbian Jesse since November 2015. Node-RED can also be installed on Windows, Linux, and Mac OS X. To install and run Node-RED on your specific system see [5].

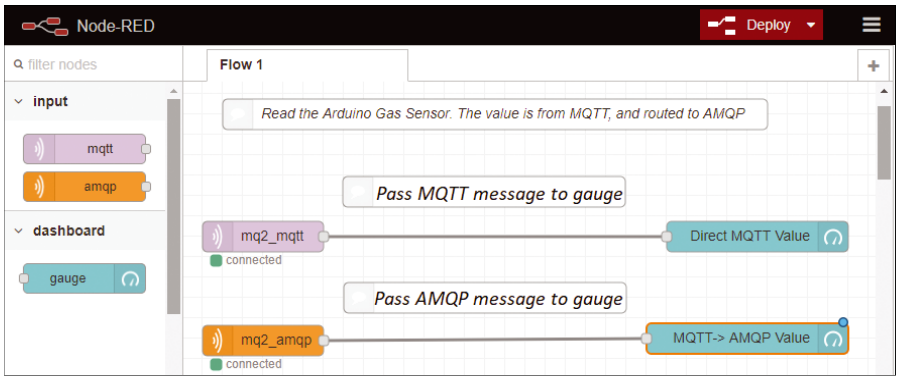
To install the AMQP components, select the *Manage* palette option from the right side of the menubar. Then search for "AMQP" and install *node-red-contrib-amqp* (Figure 8). If your installation of



node-red-contrib-amqp
Node-RED AMQP input and output nodes
0.4.5 2 years, 6 months ago

node-red-contrib-amqp-ssl
Node-RED AMQP input and output nodes
0.0.1 1 year, 5 months ago

Figure 8: Installing AMQP on Node-RED.



Read the Arduino Gas Sensor. The value is from MQTT, and routed to AMQP

Pass MQTT message to gauge

Pass AMQP message to gauge

Figure 9: Node-RED dashboard logic.

Node-RED does not have dashboards installed, search for and install *node-red-dashboard*.

For this Node-RED MQTT and AMQP example, I will use a *mqtt* and a *amqp* node from the input palette group, along with two *gauge* nodes from the dashboard group (Figure 9).

Nodes are added by dragging and dropping them into the center Flow sheet. Logic is created by making connection wires between inputs and outputs of a node. After the logic is laid out, double-click on each of the nodes to configure their specific properties. You will need to specify the MQTT and AMQP definitions of your RabbitMQ IP address, user rights, MQTT topic, and AMQP queue name. You will also need to double-click on the *gauge* nodes to configure the look-and-feel of the web dashboard.

After the logic is complete, hit the *Deploy* button on the right side of the menubar to run the logic. The Node-RED dashboard user interface can be accessed at *http://ipaddress:1880/ui*.

For my project, I used a number of different MQ sensors and inputs. Figure 10 is a picture of the Node-RED web dashboard that I created with the same MQTT value being shown natively and as a AMQP queued value.

Final Comments

I found that RabbitMQ was easy to install and the web administration plug-in, along with `rabbitmqadmin`, made the system very easy to maintain.

If you're just looking to show sensor values, then a basic MQTT broker might be all you need. However, if you're looking at some future applications like alarm, maintenance, or task lists, then AMQP exchanges and queues make RabbitMQ an interesting option. ■■■

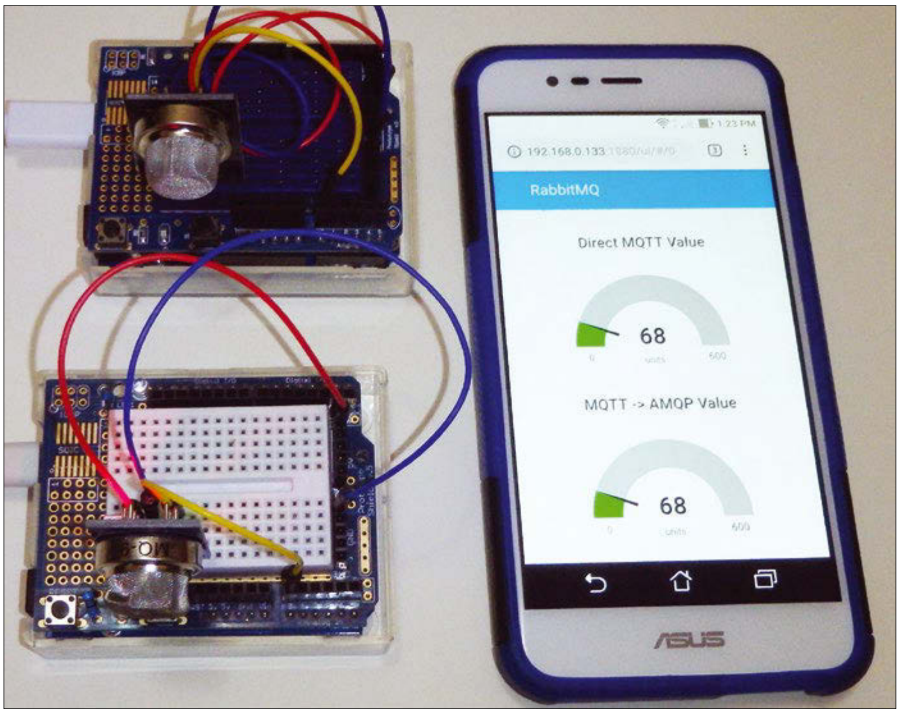


Figure 10: Node-RED dashboard with RabbitMQ data.

Info

[1] RabbitMQ: *https://www.rabbitmq.com/*

[2] RabbitMQ installation instructions: *https://www.rabbitmq.com/download.html*

[3] MQ-2 sensor: *https://www.dx.com/p/mq-2-smoke-gas-sensor-v-1-3module-for-arduino-black-2017356*

[4] Node-RED: *https://nodered.org*

[5] Installing Node-RED: *https://nodered.org/docs/getting-started/installation*

MakerSpace

Solving US electronic voting issues

Open Source Voting

In a quest for better voting machines, open source hardware may hold the answers. *By Bruce Byfield*

Attempts by Russia to interfere with US elections have been headline news in the last year. But the problems with the election process in the United States goes deeper than the public generally realizes and includes obsolete, proprietary systems, a lack of funds for upgrades, and near monopolies on voting machines. As the 2020 US elections near, academics are working to provide solutions to these issues – and open source software and hardware are at the core of these solutions, together with modern interface design. One of the most promising solutions is Prime III, being developed by Juan E. Gilbert (Figure 1), a computer scientist who heads the Human Experience Research Lab at the University of Florida [1].

Figure 1: Juan E. Gilbert has been developing open source voting machines for almost two decades.

The problems being addressed by academics such as Gilbert go back to the 2000 presidential elections. In Florida, poor ballot design combined with difficult-to-use punch card voting machines resulted in an usually high number of voters choosing either too few or too many candidates, with ambiguous results [2]. In the ensuing debate, electronic voting became the leading solution, and in 2002, the Help America Vote Act was enacted to reform the voting process [3]. Unfortunately, the machines used in the 2004 election themselves caused problems; since then, federal funds for updating voting machines have not been available. In practice, American voters often use hardware that is obsolete by three or four generations.

Today, Neal McBurnett notes that the problems continue. In the 2016 elections, recounts in three states were done on the same machines that had produced questionable results, and auditing results were complicated by the fact that the four main vote-handling formats used throughout the United States are proprietary and do not easily communicate with one another [4].

"Everyone recognizes that we have an antiquated system of voting, with machines [that are] out of date," says

Lee C. Bollinger, cochair of the Committee on the Future of Voting at the National Academies of Sciences, Engineering, and Medicine. "It is really imperative that the system be sure enough that it is a system that is credible, and that we can believe in. To that end, you have to do certain things: You have to have an upgradable system; you have to have a paper ballot; you have to [have] audited systems that can reliably tally the vote" [5].

To produce such results, Michael A. McRobbie, the other cochair of the Committee on the Future of Voting recommends a human-readable ballot and auditing of the voting process from start to finish, as well as regular updates of voter databases [6]. With another election less than two years away, the race is on to implement these much needed changes, but time is running short, and upgrades are being done on a county-by-county basis. At best, the next major American election seems likely to be conducted using a variety of technologies, including many of the machines that have created problems since 2004.

As Seth Rosenblatt wrote in 2018, "The industry that provides the hardware and software for the election process has been scarcely studied and often is opaque, even to election administrators, policymakers, and representatives at other governmental and non-governmental organizations that support or directly participate in the election process" [7].

Third Generation Technology

In the current environment, most solutions are coming from academics like Juan Gilbert rather than industry. Gilbert remembers that, shortly after the 2000 election, "my students and I were at [a] conference where we heard a speaker talk about how computers could not be used in voting. They were saying that there were security issues. Essentially, they offered no recommendations on how to fix voting, but explained what couldn't be done. My students and I were very discouraged by this and decided to do something about it. We decided to fix voting in the USA. We went to the lab, brainstormed, and invented the Prime III" – the third generation of voting technology (Figure 2), with the first being paper and pencil, and the second being the current lines of voting machines [8]. Development has continued ever since.

Gilbert and his students identified the major problem was voting machines that stored votes – a tactic that posed security risks and made the results vulnerable to tampering. To overcome this risk, they made the Prime III stateless. That is, it does not store votes. Instead, it produces a paper ballot for each vote cast that is the official ballot of record. Because each ballot is individual, vote tampering is more difficult and likely more obvious, since a stack of forgeries is more noticeable than alterations to a computer file. Just as importantly, to avoid confusion, the paper ballot shows only the office being contested and the voter's choice, not the entire list of candidates. Voters can see that the ballot accurately reflects their choice, and no ambiguities complicate the vote counting.

Another major feature of Prime III is its accessibility. Prime III allows voters to cast ballots by tapping a touchscreen or speaking into a microphone. Those who can't articulate a candidate's name have the option of blowing into the microphone, and headphones and audio instructions are available for those who have trouble reading the screen.

Gilbert adds, "Prime III is open source because we believe it's important that everyone can see the code and determine it's secure. [Moreover, because] Prime III is stateless, it could be run using machines that don't have a hard drive. Prime III could be run from read-only mediums such as DVDs to en-

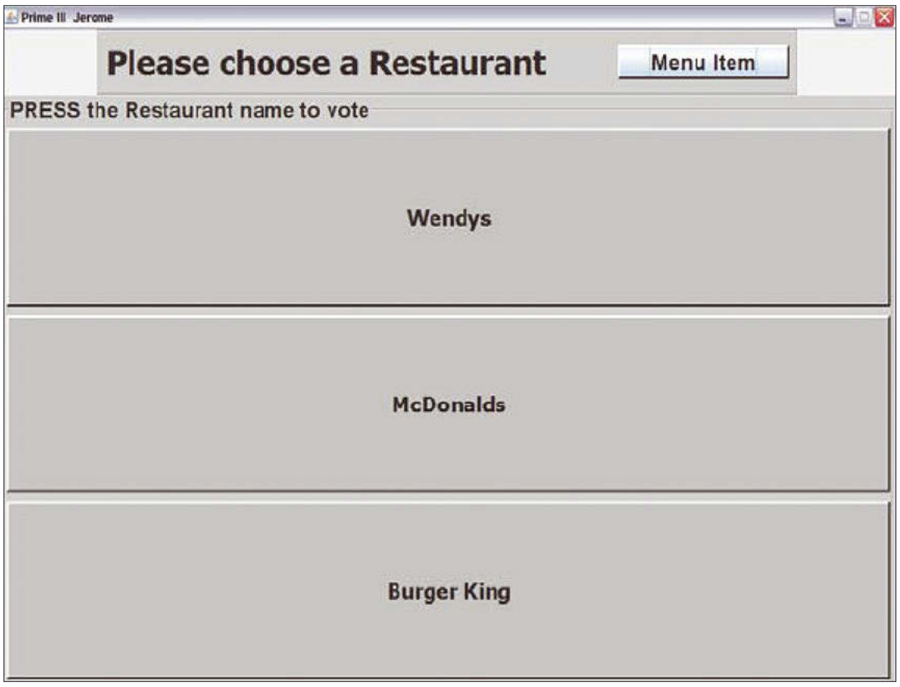


Figure 2: The screen of a test version of Prime III.

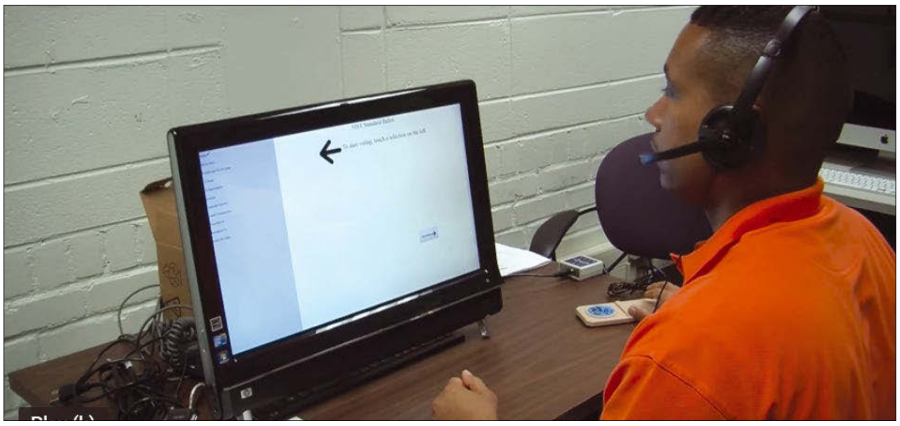


Figure 3: Prime III is assembled from standard computer components.

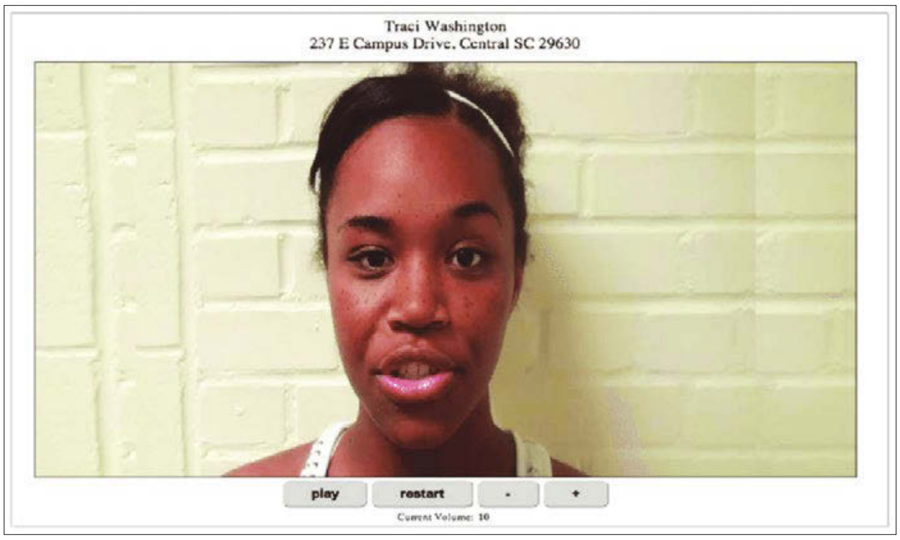


Figure 4: vSquared is a video system for voter verification being developed by Gilbert.

sure the code is what it is. From the hardware perspective, Prime III runs on commercial off-the-shelf components. You can go to Best Buy and order machines and use them in the election. If the hardware is malfunctioning, replace it. The hardware has very little to do with the actual ballot of record." Typically, the hardware consists of a standard computer or tablet, a keyboard, a headset and microphone, and a printer (Figure 3).

In addition to Prime III, Gilbert is also involved in developing systems for other voting issues. For example, VoterPass allows voters to make an appointment to vote, using web browsers, phone apps, or touch buttons on phone. Voters can opt to receive a reminder, and, at the polls, join a dedicated VoterPass voting line [9].

Gilbert is also working on Video Verification (vSquared) for voter ID [10], an issue that has been raised by both conservatives and liberals. "Essentially," Gilbert says, "You are your ID for voting. When you arrive to vote, you say your name and address to the poll worker. They will record a short video clip of you saying your name and address. They will compare you to previous video(s). If someone is going to pose as you, they will have to look like you, sound like you, and know your name and address. This approach is far superior to voter ID, [since] our studies suggest that only TSA agents and probably college bar bouncers are the only people who can accurately detect fake ID[s]. Our studies have shown that poll workers can't do this very well. However, with vSquared, they are much better" (Figure 4).

"We are also working on a new project for absentee voting. We are going to fix the issues of signature verification. Stay tuned," Gilbert says.

Barriers to Use

Prime III has pilot studies dating back to 2004. Two of these studies have been done at the state level in Wisconsin and Florida in 2014 [11]. According to Gilbert, "our pilots have consistently informed us that Prime III works. It works for voters from diverse populations, people with disabilities, and those from different age groups. Whenever we discover an issue or something we can improve, we simply implement it and move on to the next pilot study."

This verification hardly seems surprising. The logic behind efforts like Gilbert's is convincing to anyone with even a superficial understanding of computer issues. Also, although Gilbert does not mention the fact, open source solutions like his are apt to be considerably cheaper than the existing technologies. For example, San Francisco county will pay $21 million in the next three years for a contract with Dominion Voting Systems. By contrast, according to Chris Jerdonek, president of the San Francisco Elections Commission, implementing an open source replacement would cost about $6 million and could be serviced without vendor lock-in. In the ongoing absence of federal assistance, the difference in cost could very well make the difference between correcting existing problems and allowing them to continue.

However, as Gilbert admits, the struggle does not lie in creating technical solutions. Rather, "the struggle has been in getting them into practice." Gilbert identifies a lack of support as a potential problem, saying, "we are not voting machines vendors; therefore, if someone wants to use Prime III, they have to be technically savvy. We help them, but we are not vendors."

To all appearances, the concerns about voting in the United States are legitimate, and practical solutions are ready or in development. The main question now is whether the will exists to implement the solutions. ■■■

Info

[1] Juan E. Gilbert: *https://en.wikipedia.org/wiki/Juan_E._Gilbert*

[2] 2000 election: *https://en.wikipedia.org/wiki/2000_United_States_presidential_election*

[3] Help America Vote Act: *https://en.wikipedia.org/wiki/Help_America_Vote_Act*

[4] Neal McBurnett: *http://sites.nationalacademies.org/cs/groups/pgasite/documents/webpage/pga_184000.pdf*

[5] Lee C. Bollinger on problems: *http://sites.nationalacademies.org/pga/stl/voting/index.htm*

[6] Michael A. McRobbie on solutions: *http://sites.nationalacademies.org/pga/stl/voting/index.htm*

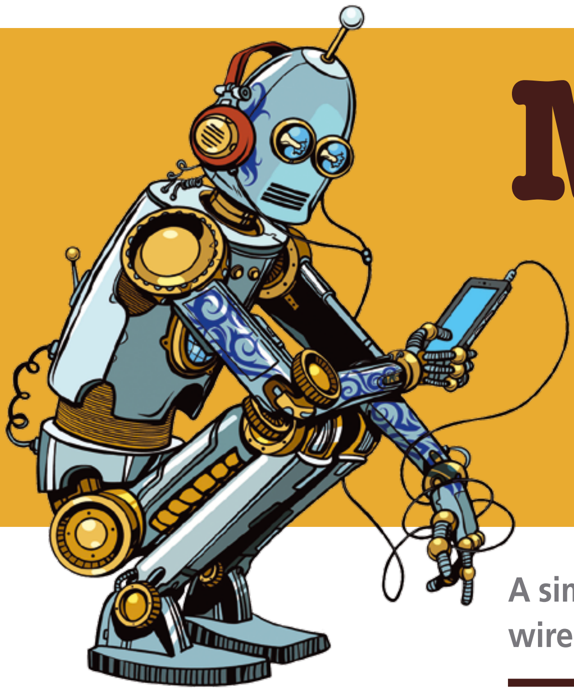
[7] "Open source the secret sauce in secure, affordable voting tech" by Seth Rossenblatt: *https://the-parallax.com/2018/11/06/open-source-secure-votingworks/*

[8] Prime III: *https://www.verifiedvoting.org/one4all/*

[9] VoterPass: *https://www.cise.ufl.edu/~juan/PrimeIII/VoterPass.pdf*

[10] vSquared: *https://www.researchgate.net/publication/283962742_Video_Verification_An_Alternative_form_of_Identify_Verification*

[11] Pilot studies: *http://newsstand.clemson.edu/mediarelations/two-states-to-use-clemson-voting-technology-in-elections/*



MakerSpace

Use an Android smartphone as a Raspberry Pi screen

Recycled

A simple Python program turns a disused smartphone into a wirelessly connected Raspberry Pi display. *By Bernhard Bablok*

New low-end $100 smartphones offer enough performance for everyday life, but the market for used devices is dismal. Instead of letting your smartphones and tablets gather dust, you can put their touchscreens to work and provide a better display than those of the same size marketed for the Raspberry Pi.

One disadvantage is the lack of a connection cable for the output from the Raspberry Pi to the smartphone. Fortunately, you have alternatives, such as a wireless connection, which I will show you how to set up in this article.

Another problem is the display size. Although the 800x480-pixel resolution of a smartphone, for example, is close to that of the official Pi display, the screen area is extremely small; a normal desktop simply cannot be displayed on devices smaller than a 10-inch tablet.

For many projects, you don't need a full desktop, though (e.g., if you only need to output a small amount of data or status information), and you can limit the controls to a couple of buttons. With very little effort, you can write programs with graphical interfaces that fit perfectly on the smartphone screen. In this article is an example program you can use as a model for your own projects.

Alternatives

One approach for displaying a graphical user interface on the smartphone is to split the application and display into two programs. For example, the Rasp Pi could run a program that serves up its output as a website, with the smartphone browser displaying the results.

The disadvantage of this solution is that, whereas a small web server can be integrated into any program – independent of the programming language and without too much overhead, thanks to a wide variety of program libraries – an attractive presentation also requires know-how in web design and JavaScript programming.

A variant of this approach uses an app on the smartphone instead of the web browser. Although this involves significantly more work, it also offers more freedom (e.g., communication over Bluetooth instead of a wireless network).

The smart Blue Dot [1] app, which sets up an Android device as a Bluetooth remote for Raspberry Pi, can be used as a project template, and you can integrate it directly into your own programs thanks to the matching Python library (Figure 1). The well-known Pi Control [2], which is a web-based application that provides status information about your Rasp Pi, is another example and requires you to be familiar with the Android or iOS environment, instead of web design.

If you only want to write a program, use the approach described in the following sections. The program runs on the Rasp Pi and outputs the graphical user interface to another computer on

Author

Bernhard Bablok (*mail@bablokb.de*) is an SAP-HR developer at Allianz Technology SE. When he's not listening to music, riding his bike, or walking, he focuses on Linux, programming, and small computers.

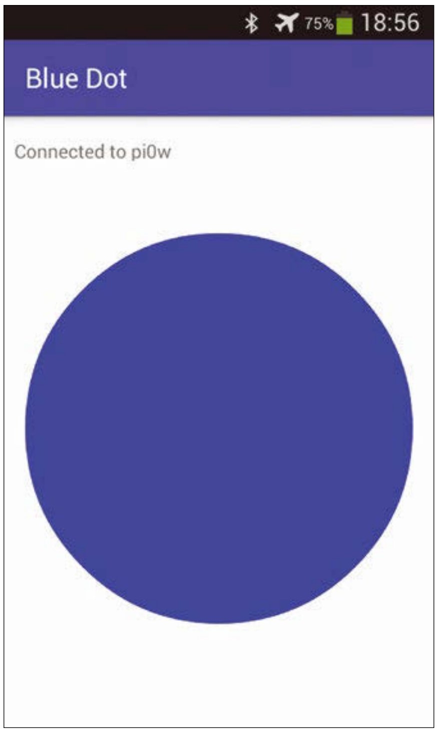


Figure 1: Blue Dot lets you control a Raspberry Pi with a Python application.

the network (i.e., a smartphone or tablet) rather than over its HDMI interface. Although not particularly efficient – both the user data and the user interface need to cross the wire – the programming overhead is manageable.

Old and Good?

Programs running on the Rasp Pi output the graphical user interface via the X server, an independent program that receives the graphics commands and takes care of the rest of the work.

X server is older than Linux itself and dates back to a time when computers were exclusively located in data centers. Special terminals in the office ran only an X server, which displayed programs with graphical interfaces. The server received the graphics commands over the network from the programs in the data center.

The combination of a Rasp Pi and smartphone uses the same approach. The Rasp Pi plays the part of the remote computer and the smartphone the part of the X terminal. Because you only want to see the output of a single program and are not interested in seeing the complete desktop interface, you do not even need to install the Raspbian Pixel desktop.

On the Pi side, you only need the X client. If you have installed Raspbian with a graphical user interface, everything you need is already on board. For the Lite version, you need a separate client, which also handles the installation of the sample program as a desirable side effect.

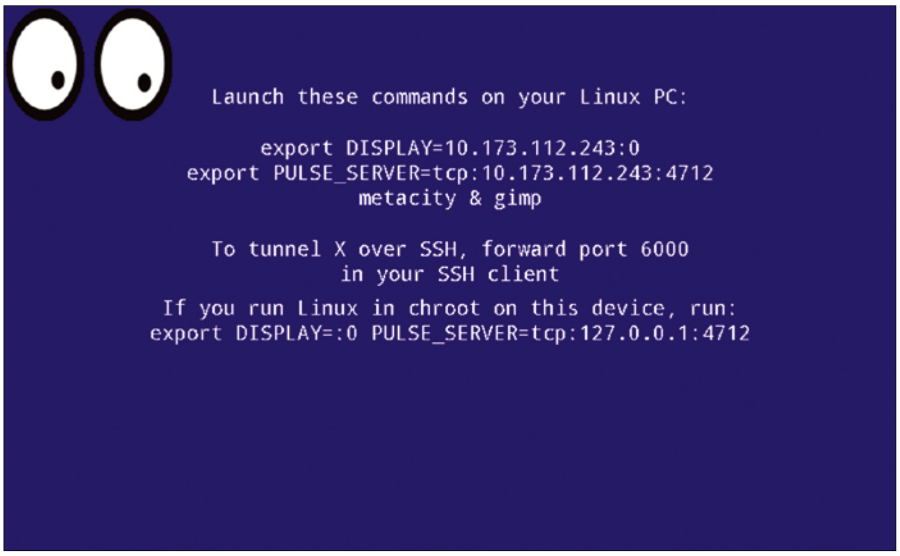
The ancient X architecture has suffered from various issues over the years, especially relating to security and performance. Modern computers (like the Rasp Pi) contain a powerful GPU that does the rendering itself, which does not make sense if you want to transfer the output across the wire and restricts you somewhat in your choice of program libraries: The output from directly rendered programs only displays locally.

X Server

For an Android smartphone, you need to install XSDL [3], a free and open source [4] X server, from the Playstore. In the lab, I even managed to install it on an ancient device with Android 2.3, but the program always crashed when launched. Through lack of experience and no apparent suitable app, I cannot give iOS users any recommendations.

When starting the X server, configure it using the button that occupies the entire top margin. Set the mouse emulation to *Tablet* and the left mouse button to *Touch or hold*. In the *Video* menu, configure the alignment to suit your needs for the project at hand (e.g., landscape, portrait, automatic).

The next configuration step looks a bit arcane: A screen appears that offers



```
Launch these commands on your Linux PC:

export DISPLAY=10.173.112.243:0
export PULSE_SERVER=tcp:10.173.112.243:4712
metacity & gimp

To tunnel X over SSH, forward port 6000
in your SSH client

If you run Linux in chroot on this device, run:
export DISPLAY=:0 PULSE_SERVER=tcp:127.0.0.1:4712
```

Figure 2: The X server after startup. The Xeyes application is running in the upper left corner.

various resolutions. If you do not want to use the native resolution, then tap on the desired alternative. You can also select font scaling afterward. You can change both at each startup; in case of doubt, the server simply starts with the old configuration. Changing the resolution is only possible on displays with a very high resolution anyway.

The X server itself does not provide any interaction; after startup you will only see a blank screen with some instructions. More specifically, the server displays the IP address of the mobile device. Every application you launch then reverts to this background. For example, Figure 2 shows the Xeyes program from the *x11-apps* package that's just for fun: The eyes follow the mouse cursor.

For the Rasp Pi to find the smartphone automatically later on, you need to assign a fixed IP address on the local network to the cellphone. You can either do this via your router by permanently assigning the IP address to the MAC address of the device or in the advanced network settings of the smartphone, which you can normally access with a long press on the network connection.

To avoid managing the lengthy IP address on the Raspberry Pi, you can enter it in the Pi's `/etc/hosts` file and assign a meaningful name to the smartphone.

Call Me

If you use the Raspbian Pixel desktop, an X server is already running and all programs send the graphic output to this local server. However, this can be easily changed for individual programs, because each program checks the `DISPLAY` environment variable. You can change this value by specifying the value at the command line for one program only (line 1) or for all subsequent commands (line 2):

```
$ DISPLAY=192.168.2.34:0 xeyes
$ export DISPLAY=192.168.2.34:0
```

The syntax of the `DISPLAY` variable always follows the pattern <IP address>:<X>.<Y>. On most systems the suffix (i.e., the display/screen number) is `0.0`, or simply `0`. You can also use the hostname of the target device instead of the IP address.

If you always use the same device in combination with the Rasp Pi, you should enter the display variable in the `/etc/profile` file, which sets the variable globally. However, this setting only applies to new sessions, and you do not want to do this for Pixel systems, because the output of the desktop on the locally connected screen requires the default value `DISPLAY=:0.0`.

Superficial

Creating a graphical user interface for a program is one of the most complex tasks in IT. Everyone can and wants to have a say, and everyone has a different idea of what a beautiful interface or ergonomic design is. This topic is non-trivial, because you need to keep the program operable even if it is busy doing other things (e.g., computing).

Fortunately, many libraries exist that make your life simple. The example here uses the Python programming language. Originally, the modern Kivy [5] was intended as a graphics library, but it uses highly efficient OpenGL for direct rendering, which does not work over the network. The project is now based on wxPython, a wrapper for the wxWidgets toolkit implemented in C++.

For your Raspberry Pi, you need to create a program that simulates a multiline liquid crystal display with control buttons (Figure 3). At the push of a button, you can request various items of system information, and the results end up in a text box. The whole thing can be implemented with surprisingly few lines of code.

Before programming, you have to install the required Python library:

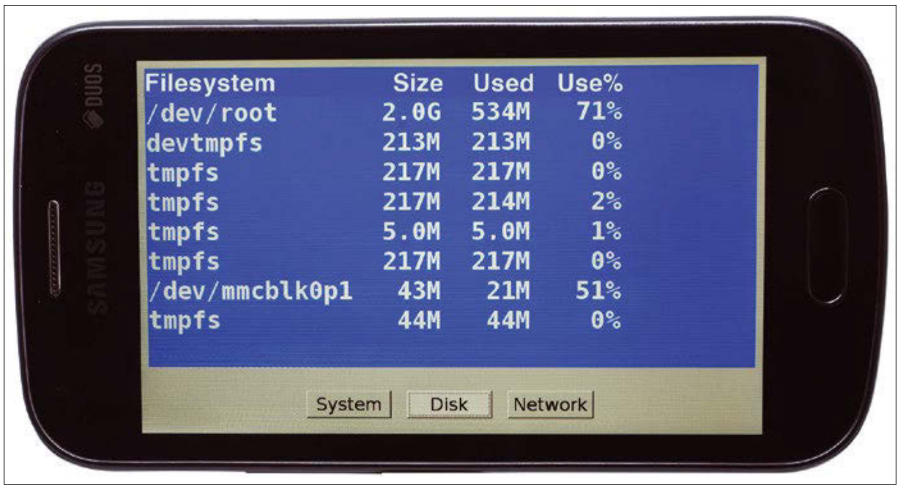

```
Filesystem        Size  Used  Use%
/dev/root         2.0G  534M   71%
devtmpfs          213M  213M    0%
tmpfs             217M  217M    0%
tmpfs             217M  214M    2%
tmpfs             5.0M  5.0M    1%
tmpfs             217M  217M    0%
/dev/mmcblk0p1     43M   21M   51%
tmpfs              44M   44M    0%
```

Figure 3: The sample program on a smartphone with a screen resolution of 800x480 pixels.

```
$ sudo apt-get update
$ sudo apt-get -y install python-wxgtk3.0
```

On Raspbian Lite, the command also installs all the necessary auxiliary libraries for running graphical applications (i.e., the X client), so don't worry if the package list looks very long.

If required, also set up the *wx3.0-examples* package with examples in C++. Because method calls in C++ and Python are almost identical, the examples can be transferred easily to the Python world.

wxPython

The basic architecture of a wxPython program is no different from any other program with a graphical user interface. The main part consists of an infinite loop, known as the event loop.

Every interaction with the program, such as pressing a button, entering text, or moving the window, triggers an event. The event loop processes the interaction by passing it to the appropriate Python methods. The sample program has three buttons and, thus, three methods for processing.

Listing 1: WxVlcdApp.py (Excerpt)

```
#!/usr/bin/python
import os, threading, time, subprocess, re, socket, time
import wx

[...]

--- Auxiliary class for asynchronous update of the textbox
SYSCMD_EVENT_TYPE = wx.NewEventType()
EVT_SYSCMD = wx.PyEventBinder(SYSCMD_EVENT_TYPE, 1)

class SysCmdFinishEvent(wx.PyCommandEvent):
  """ wrap result of a system command into an event """

  def __init__(self, ev_id, value=None):
    wx.PyCommandEvent.__init__(self, SYSCMD_EVENT_TYPE, ev_id)
    self._value = value

  def get_output(self):
    return self._value

--- Main application frame
class AppFrame(wx.Frame):
  """ Application toplevel frame """"

  # --- Constructor
  def __init__(self, parent, title):
    wx.Frame.__init__(self, parent, -1, title, size=(800, 480))
    #self.SetMenuBar(self.create_menu())
    #self.CreateStatusBar()
    self.create_controls()
    self.Bind(EVT_SYSCMD, self.on_syscmd_result)
    self.on_uptime(None)

  # --- Create widgets
  def create_controls(self):
    """ create panel with all controls """
    vbox = wx.BoxSizer(wx.VERTICAL)
    vbox.Add(self.create_outputarea(self), 1, wx.ALL|wx.EXPAND, 10)
    vbox.Add(self.create_buttons(self), 0, wx.ALL|wx.ALIGN_CENTER_
      HORIZONTAL, 10)
```

Listing 1: WxVlcdApp.py (Excerpt) (continued)

```
    self.SetSizer(vbox)

[...]

  # --- Create button bar
  def create_buttons(self, parent):
    button_panel = wx.panel(parent)
    sys_btn = wx.button(button_panel, -1, "System")
    disk_btn = wx.Button(button_panel, -1, "Disk")
    net_btn = wx.Button(button_panel, -1, "Network")
    self.Bind(wx.EVT_BUTTON, self.on_uptime, sys_btn)
    self.Bind(wx.EVT_BUTTON, self.on_df, disk_btn)
    self.Bind(wx.EVT_BUTTON, self.on_ip, net_btn)
    hbox = wx.BoxSizer(wx.HORIZONTAL)
    hbox.Add(sys_btn, 0, wx.ALL, 10)
    hbox.Add(disk_btn, 0, wx.ALL, 10)
    hbox.Add(net_btn, 0, wx.ALL, 10)
    button_panel.SetSizer(hbox)
    return button_panel

  # --- Create text box for output
  def create_outputarea(self, parent):
    text_panel = wx.panel(parent)
    text_panel.SetBackgroundColour(wx.BLUE)
    self._output_text = wx.StaticText(text_panel, -1, "")
    self._output_text.SetForegroundColour(wx.WHITE)
    self._output_text.SetFont(wx.Font(14, wx.MODERN, wx.NORMAL, wx.BOLD))
    self._output_text.SetSize(self._output_text.GetBestSize())
    hbox = wx.BoxSizer(wx.HORIZONTAL)
    hbox.Add(self._output_text, 1, wx.EXPAND, 0)
    text_panel.SetSizer(hbox)
    return text_panel

  --- handler for the uptime button
  def on_uptime(self, evt):
    """ start asynchronous command """
    threading.thread(target=self.get_uptime).start()

[...]

  --- handler for asynchronous update of the textbox
  def on_syscmd_result(self, evt):
    self._output_text.SetLabelText(evt.get_output())

  # --- Exit handler
  def on_exit(self, evt):
    """ exit-handler """
    self.Close()

  # --- Query uptime (executed asynchronously)
  def get_uptime(self):
    hostname = self.read_hostname()
    tm = time.strftime("%H:%M:%S")
    (uptime, idle) = self.read_uptime()
```

In Listing 1, you can see the most important sections of the `WxVlcdApp.py` program, which you can find in my GitHub repository [6]. The event loop described can be found in the last line, which is where the program calls the `MainLoop()` method of the `WxVlcdApp` application class defined earlier.

The class extends the `wx.App` class and consists here of only a single method, `OnInit()`, which generates the application's user interface. To do this, a separate class, `AppFrame`, is defined (line 22) that constitutes the main part of the program.

The `AppFrame` constructor first creates the graphical elements in the `create_controls()` method (line 35), for which you use a `BoxSizer()` – known from other toolkits as `BoxLayout()`, or something similar.

This layout element aligns its children vertically; in the example these are the output text boxes and the buttonbar. The parameters of the `Add()` method define which child is allowed to expand and how it aligns itself. Further details can be found in the documentation for wxPython. Similarly, the `create_buttons()` method (starting in line 45) creates the toolbar. Here, three buttons are arranged in a horizontal `BoxSizer()`.

Additionally, lines 50 through 52 determine which methods the program executes when the individual buttons are pressed. To add more buttons, simply copy and paste the lines in question.

Event Processing

The button methods retrieve system information and write the results to the

Listing 1: WxVlcdApp.py (Excerpt) (continued)

```
094     load = self.read_loadavg()
095     output = """Hostname: %s
096 Time: %s
097 Uptime: %s
098 Idle: %s
099 Load:     %s""" % (hostname, tm, uptime, idle, load)
100     evt = SysCmdFinishEvent(-1, output)
101     wx.PostEvent(self, evt)
102
103 [...]
104
105   # --- Read hostname
106   def read_hostname(self):
107     """ read hostname from /proc/sys/kernel/hostname """
108     with open("/proc/sys/kernel/hostname") as f:
109       hostname = f.read().split()
110       f.close()
111     return hostname[0]
112
113 [...]
114
115 # --- Main application class
116 class WxVlcdApp(wx.App):
117   def OnInit(self):
118     frame = AppFrame(None, "Virtual LCD")
119     self.SetTopWindow(frame)
120     frame.show(True)
121     return True
122
123 # --- Main program
124 if __name__ == '__main__':
125   wait_for_X11()
126   app = WxVlcdApp(redirect=False)
127   app.MainLoop()
```

text field. However, this cannot happen directly because the application "freezes" while the method is running. `MainLoop()` cannot respond to interface events (move window, press button, etc.) during this time.

The button method `on_uptime()` starting in line 74 thus only starts a thread, and the actual work is then handled asynchronously in `get_uptime()` (starting in line 90).

As usual with all graphical libraries, only the main thread (`MainLoop()`) can update the interface. Therefore, the thread creates and sends a self-defined event at the end (line 100). This private event, which I dub `SysCmdFinishEvent()`, contains only the text to be displayed; the definition is at the beginning of the listing.

The program also binds a method to this event (line 31) in the constructor of `AppFrame`. A text field is updated here (starting in line 81), and thus the circle closes.

If you want to fill a field automatically with status information, you can use the same approach. A thread running for a long time simply sends an update event with the new field content at fixed intervals.

Integration

The program still has two problems: If it starts when the X server is not (yet) running, it crashes, and it stops as soon as the X server app stops running or is out of range.

You can solve the first problem by testing for the X server at program launch and then only continuing the program if there is a response (the code is missing from Listing 1 because of a lack of space). As soon as the app starts, the interface pops up. This only works if you don't use multiple Raspberry Pis on one smartphone.

The system daemon solves the second problem for you. You can start the program as a service with the `wxvlcd.service` file (Listing 2), which is also where you set the `DISPLAY` variable. The only special thing about the unit definition is line 10 with the `Restart=always` statement. It ensures that systemd automatically restarts the program after it has ended. Details and alternatives to `always` (e.g., `on-failure`) can be found with the `man systemd.service` command.

Conclusions

Thanks to the X server and wxPython, you can implement simple control and info displays for your Rasp Pi projects with minimal effort – and all communication is wireless.

If you do not have an Android device or would rather use one of the small 3.2-inch displays with the Raspberry Pi, you can still use wxPython. The program evaluates the `DISPLAY` variable, so the program can be output locally or over the network, depending on the configuration.

The mature development stage of wxPython offers another benefit. The toolkit is extremely stable, and you will find a sample solution for almost every need online. Even if you have never programmed a graphical user interface before, you will quickly turn out an attractive GUI just by following the example in this article. ■■■

Listing 2: wxvlcd.service

```
01 [Unit]
02 Description=Virtual LCD service
03 After=multi-user.target
04
05 [Service]
06 Type=simple
07 User=pi
08 ENVIRONMENT=DISPLAY=ip-x11-server:0
09 ExecStart=/usr/local/bin/WxVlcdApp.py
10 Restart=always
11
12 [Install]
13 WantedBy=multi-user.target
```

Info

[1] Project Blue Dot: *https://www.stuffaboutcode.com/2017/04/bluedot-bluetooth-remote-for-raspberry.html*

[2] Pi Control: *https://pi-control.de*

[3] Android X server (app): *https://play.google.com/store/apps/details?id=x.org.server&hl=en*

[4] Android X server (project): *https://github.com/pelya/xserver-xsdl*

[5] Kivy: *https://www.kivy.org*

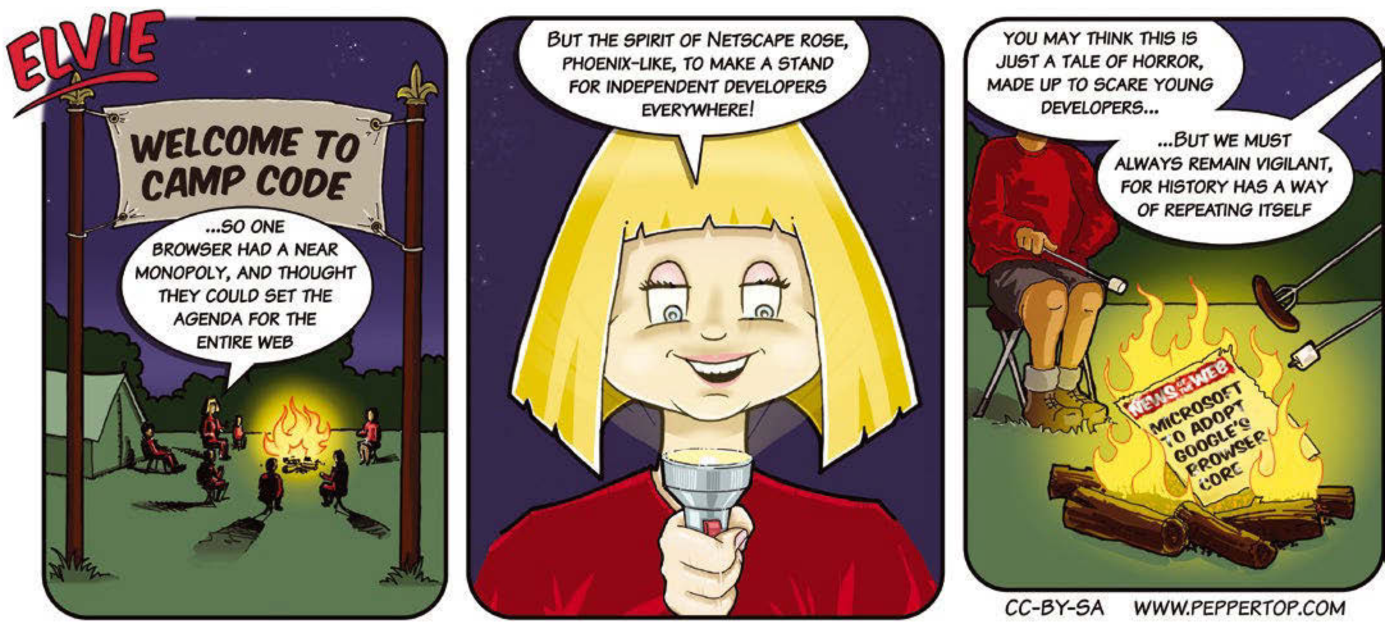
[6] Project page for the article: *https://github.com/bablokb/pi-vlcd*

Linux comes with lots of tools, but the community gravitates to a much smaller constellation of preferred applications. For word processing, most users turn to LibreOffice, although several other word processing tools populate the repositories of the top Linux distros. For a web browser, most users turn to Firefox or Chrome. Gimp is the king of raster (bitmap) graphics tools for the Linux space, with Krita as a leading alternative, but is there more to the story? And shouldn't we, reporters and chroniclers of the Linux space, reach wider across the landscape to bring you the alternatives? This month, we take a look at Pixelitor, a free graphics editor that might not be as multi-featured as Gimp, but that's good news for users who have a more minimalist bent.

And speaking of alternatives, the ip networking utility has been around for years, and it really ought to be the go-to tool right now for network configuration and troubleshooting, but many users still consider it a second-tier alternative to classic utilities like ifconfig, route, and arp. In this month's issue of LinuxVoice, we explore the powers of ip.

Image © Olexandr Moroz, 123RF.com

LINUXVOICE


ELVIE
WELCOME TO CAMP CODE
...SO ONE BROWSER HAD A NEAR MONOPOLY, AND THOUGHT THEY COULD SET THE AGENDA FOR THE ENTIRE WEB
BUT THE SPIRIT OF NETSCAPE ROSE, PHOENIX-LIKE, TO MAKE A STAND FOR INDEPENDENT DEVELOPERS EVERYWHERE!
YOU MAY THINK THIS IS JUST A TALE OF HORROR, MADE UP TO SCARE YOUNG DEVELOPERS...
...BUT WE MUST ALWAYS REMAIN VIGILANT, FOR HISTORY HAS A WAY OF REPEATING ITSELF
NEWS OF THE WEB
MICROSOFT TO ADOPT GOOGLE'S BROWSER CORE
CC-BY-SA WWW.PEPPERTOP.COM

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

MADDOG'S DOGHOUSE

Fifty years ago, limitations in computing had more to do with cost than know-how; maddog takes us back to 1969. BY JON "MADDOG" HALL

The limits of affordability

When you have been in the computer industry for 50 years, you tend to have gray in your hair (assuming you still have some hair left). You also have a series of friends who talk about "way back when."

At the same time, you notice a group of "younger people" who have questions about why things are the way they are and sometimes have some slightly strange ideas about how things came about.

So, old timers, pardon me while I walk through time and loosely paint a path of where we are and how we got there.

Imagine the year 1969 (not really so long ago) when even mainframe computers ran only one task at a time. A machine that could fill a room and used water cooling and enough electricity to make a significant dent in your electric bill only had one-quarter to one-half megabyte of core memory.

That machine might also have one or two disk drives that cost $32,000, apiece and each would hold 180MB of disk storage. The processors were able to do between 35,000 to a bit over 16 million instructions per second, depending on the model.

The address space of a task at that time was about 16 million bytes for this mainframe computer, with a 24-bit address space. All of the program had to fit into memory at one time, unless you used a trick called "overlays," which caused pain and suffering to the programmer.

Only large companies and governments could afford anything more, and often not even them. Since most computer companies only built machines that could be sold, improvements in size and capacity moved slowly.

It was not that we did not know *how* to solve big problems; we could not *afford* to solve big problems.

"Operators" mounted magnetic tapes, fed punched cards, and tore off printer listings to be distributed to the employees who needed them.

Programs read directly from card readers and wrote directly to printers. This limited the speed of the program to what card readers could read in and what printers could print out. Good programs started reading the next card while they were processing the existing card and scheduled the next line to be printed when the printer sent the interrupt saying that the last line was finished printing. The operators would not launch the next job until the last one had finished printing.

Then "spooling" was invented … a trade off of disk space and memory for faster access to data and printing. Jobs could now be "queued" to run whenever resources in the machine allowed, and more memory was added to allow multiple jobs to run at one time.

Security was still locking the door at night (no terminals attached to this machine), networking was carrying the boxes of cards and tapes around, and graphics was printing ASCII art on line printers.

Graphics, audio, and video that we take for granted today were unknown to the general public, and most people never saw a real computer or touched a computer keyboard until they went to university or started working for a company. There were no computers in the high school or at home.

A single transistor in those days might cost $1.50, and integrated circuits were just beginning to replace discrete transistors in the late 1960s, so having any larger memories or faster CPUs was simply not affordable.

I remember buying 128,000 bytes of semiconductor memory in 1978 for about $128,000, about twice what you might pay for a decent house at the time. A few years later (1981), I paid $10,000 for a megabyte of RAM, and a couple of years after that I paid $2,000 for the same megabyte of RAM.

You needed to balance the speed of the CPU with the amount of RAM you had, the amount of disk storage, and a series of other elements. What would be the sense (in most cases) of having a CPU that could do billions of instructions a second accessing memory that was only kilobytes in size with a disk measured in megabytes? Or having 10GB of RAM with a 5MB storage device?

We had supercomputers for a long time, and we knew how to solve big problems, but the average company could not afford a monolithic supercomputer, so a lot of the "big problems" went unsolved.

The concept of the Beowulf supercomputer allowed people to solve the same problems for approximately 1/40th the price, or to solve problems that are 40 times bigger for the same amount of money.

The graphics, networking, and processing capabilities we have today (including mobile phones) grew out of a combination of need and technology that grew over time and continued to grow, but these capabilities are still limited by our ability to afford them. ■■■

New Pixelitor release with interesting features

Lightweight

Pixelitor offers the basic functions of a full-fledged image editing program, along with some useful filters and a few pitfalls.

BY KARSTEN GÜNTHER

The best thing about Linux is the great variety of available free tools – whatever it is you need to do, the open source community has at least one – and probably several – tools for the task. Raster graphics is no exception. The famous Gimp (Gnu Image Processor) is the highest profile graphics tool in the Linux space. Gimp ships with many popular distros, and the ones that don't include Gimp by default make it available through package repositories. Many users adore Gimp, but it isn't for everybody. Some users think Gimp is too big and confusing, with too many different features. Others prefer a tool that is targeted to a more narrow set of tasks.

Another free tool in the raster graphics space is Pixelitor, an open source graphics editor based on Java. Pixelitor [1] is truly a cross-platform application that caters to users on Mac OS and Windows as well as Linux. Pixelitor supports "layers, layer masks, text layers, filters, and multiple undo." The latest version ships with more than 110 filters and color adjustments (such as the Photo Collage filter shown in Figure 1). But beginners beware: the project developers make it clear that Pixelitor is "… intended to be powerful rather than simple. You should have some experience with image editors in order to use it, because there isn't much documentation available yet, [although] many concepts (layers, blending modes, cropping, Gaussian blurring, unsharp masking, histograms, etc.) are the same" as with other similar tools. Pixelitor doesn't provide a lot of hand-holding for beginners, but many converts believe it offers a more compact and streamlined interface for users who know what they are doing.

We decided it was time to take a closer look at Pixelitor and the capabilities it brings to the Linux space. But I'll start with a note on formats. Pixelitor uses the PXC file format as its native format, which should not be confused with the Picture Exchange (PCX) format [2] used with Gimp and other tools. Pixelitor can export files into the usual export formats, such as JPEG, PNG, BMP, and GIF (without transparency), but the Pixelitor developers recommend using PXC to save all internal information, such as layers, selections, and paths, and other elements.

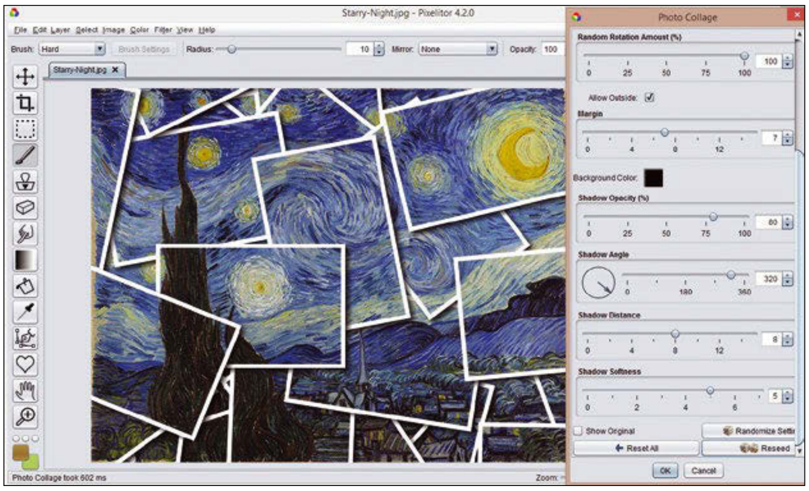
If you wish to save layers and open the file somewhere else, like in Gimp or Krita, OpenRaster (`.ora` extension) is the better choice as an export format. OpenRaster format will preserve the layers, although it will not retain layer masks and other enhanced features.

Installation

Pixelitor requires Java version 8 or higher to run. To start it, enter:

```
java -jar Pixelitor.jar
```

Usually programs that you do not install from the repositories of the distribution are located in `/opt/`, but Pixelitor will typically reside in `/opt/pixelitor/4.2/`. The current program file can be found online [3].



Brush: Hard
Radius: 10
Mirror: None
Opacity: 100
Random Rotation Amount (%): 100
Allow Outside: ✓
Margin: 7
Background Color: ■
Shadow Opacity (%): 80
Shadow Angle: 320
Shadow Distance: 8
Shadow Softness: 5
Photo Collage took 602 ms

Figure 1: Pixelitor's Photo Collage filter creates a collage from a single image.

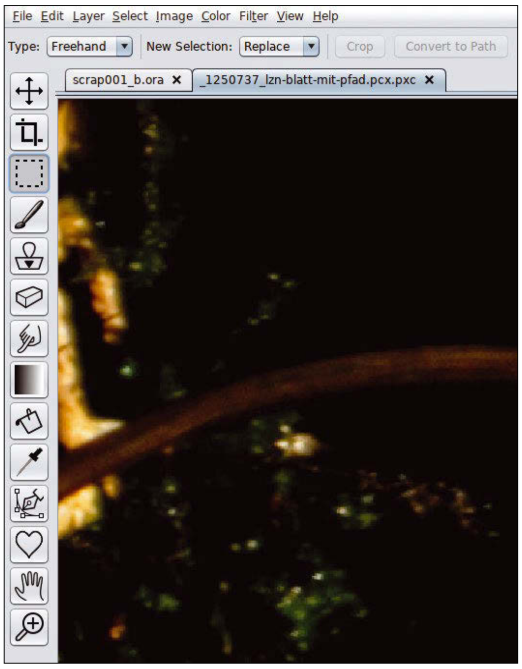

File Edit Layer Select Image Color Filter View Help
Type: Freehand
New Selection: Replace
Crop
Convert to Path
scrap001_b.ora
_1250737_lzn-blatt-mit-pfad.pcx.pxc

Figure 2: The tools in Pixelitor reside in a toolbar on the left side of the window.

The other important note is that Pixelitor currently only supports a color depth of 8 bits, which was considered sufficient for PC-based applications in years past but is definitely not up to the quality of today's high-quality professional tools.

Getting Started

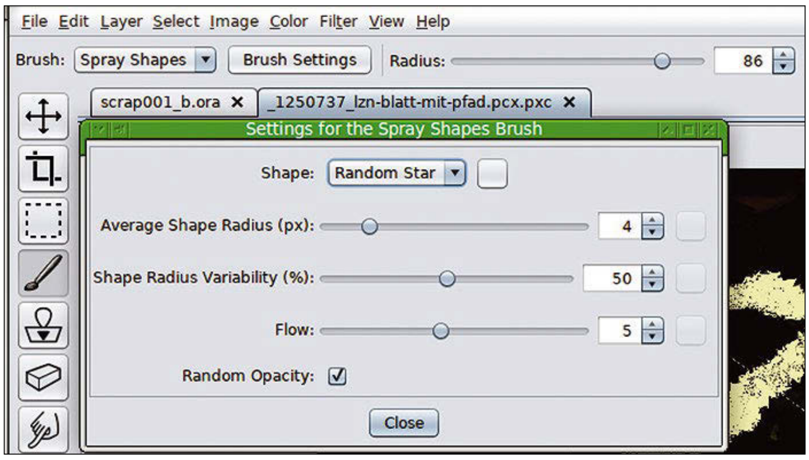
See the box entitled "Installation" for first steps to install PixelitorAfter launching, the program shows a tip to help you use the software. You can load an image from the File menu or by dragging it onto the window with the mouse. Additionally, the program supports pasting from the clipboard (Ctrl+V).

Tools work in Pixelitor as in Gimp: Exactly one tool is active at any given time. Filters and other functions can be used in parallel, but this does not affect the tool. The software is limited to a set of essential tools, which it offers in the toolbar on the left (Figure 2). Compared to Gimp, the tools are more generic and have significantly fewer settings.

The application has a *Move Tool* at the top that only worked on layers in our lab, so I cannot say whether and how moving selections or paths works. During cropping, you can find guidelines under the *Guides* drop-down, much as Gimp offers for various aspect ratios.

A universal *Brush Tool* is available for drawing and painting. Different brush tips simulate various tools, such as an airbrush and a calligraphy pen. Only a few of the tips support settings (Figure 3), which means tools that otherwise work well are difficult to adjust or adapt. The brush can be adjusted down to the *One Pixel* level.

Figure 3: Some tools allow you to adjust parameters more precisely in a Settings dialog.


File Edit Layer Select Image Color Filter View Help
Brush: Spray Shapes
Brush Settings
Radius: 86
scrap001_b.ora
_1250737_lzn-blatt-mit-pfad.pcx.pxc
Settings for the Spray Shapes Brush
Shape: Random Star
Average Shape Radius (px): 4
Shape Radius Variability (%): 50
Flow: 5
Random Opacity: ✓
Close

Cloning (indicated by a stamp icon for the *Clone Stamp Tool*) is essentially a healing tool that takes the brightness values in the area into account. However, Pixelitor does not display the area from which the pixels originate nor the size of the brush tip during cloning; thus, you need to rely on your intuition to understand the scope of your actions.

The tools for erasing and smearing are similar to those for painting. The same brush tips are available for the *Eraser Tool* as for painting, whereas the *Smudge Tool* has only a hard and a soft brush for smearing. Because alpha channels are automatically added to the layers, the eraser always creates a transparency. The *Smudge Tool* allows you to create a virtual background color with the *Finger Painting* option at the top of the window. Without this option, smearing works like the basic settings for the tool of the same name in Gimp.

As expected, Pixelitor only supports one color gradient: an even gradient from foreground to background color. As with Gimp, the *Gradient Tool* fills the entire image if no limiting selection exists. The same applies to the *Paint Bucket Tool* (fill tool), which supports colors, but not patterns. The *Color Picker Tool* pipette works as expected: If you enable it, each mouse click changes the color for the foreground.

If required, you can also create paths as Bézier curves, which you can adapt and later convert into a selection, if required. The *Pen Tool* has two modes: *Build* and *Edit*. However, you can only remove nodes you set previously in *Edit* mode, and you need the mouse to toggle between modes in a drop-down at the top of the window, which is annoying in daily work. In the context menu (Figure 4) of the *Pen Tool*, you will find the *Delete Point* function.

The heart icon is the *Shapes Tool* for creating predefined shapes. The *Effects* button lets you further modify these figures; however, such gim-

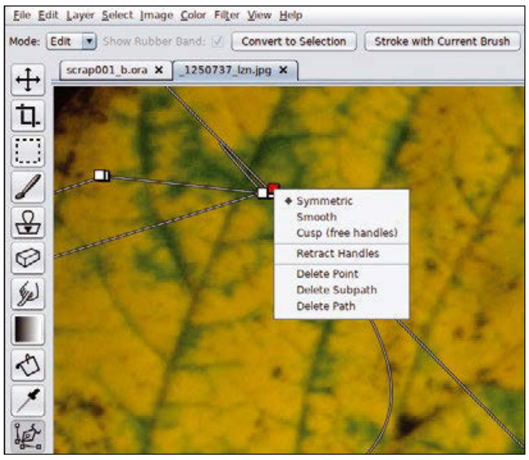
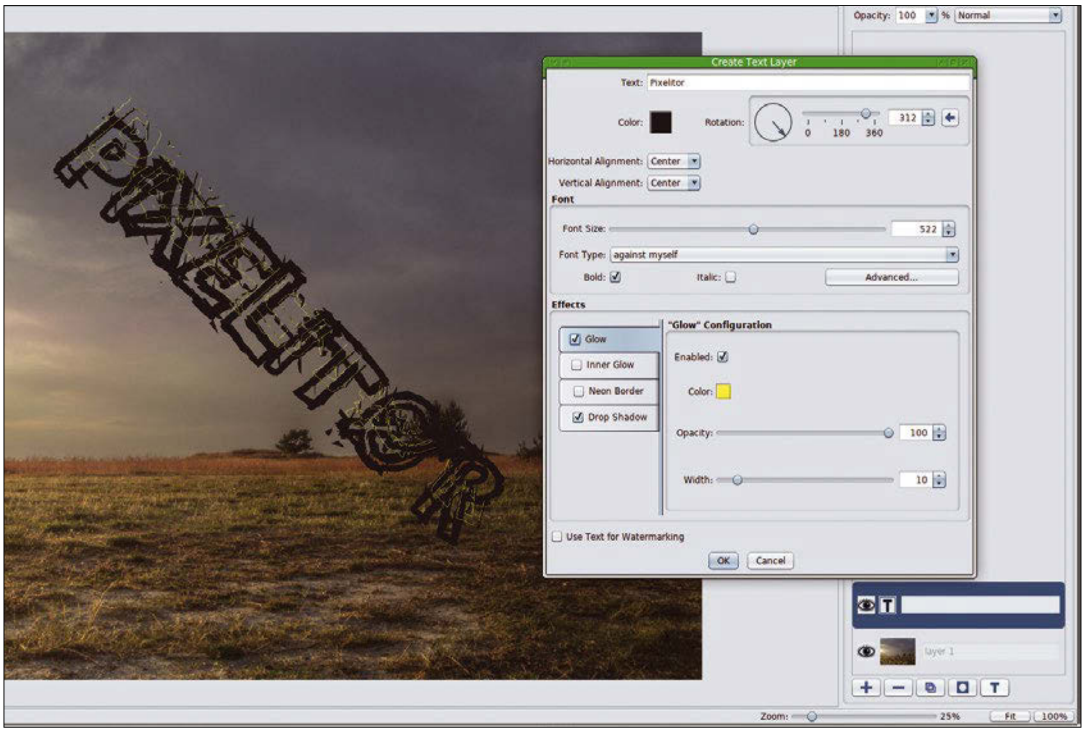


Figure 4: You can create or edit paths with the ***Pen Tool***. Additional functions are available in the context menu.

micks are of limited value in everyday work.

The *Hand Tool* moves the view, much like the Navigator in Gimp. A real navigator is also available: You will find it under *View | Show Navigator*. However, it will not dock. As a rule, you only need these aids if you cannot intuitively move the displayed section with the mouse. The magnifying glass is the *Zoom Tool* and works as you would expect.

At the bottom of the left sidebar is a color selector for the foreground and background color, with three buttons for special color settings (*Reset Default Colors*, *Swap Colors*, and *Randomize Colors*). As with the filters, the randomize button lets you select colors or other parameters randomly.


Create Text Layer

Text: Pixelitor

Color: ■ Rotation: 312

Horizontal Alignment: Center

Vertical Alignment: Center

Font

Font Size: 522

Font Type: against myself

Bold: ☑ Italic: ☐ Advanced...

Effects

☑ Glow ☐ Inner Glow ☐ Neon Border ☑ Drop Shadow

"Glow" Configuration

Enabled: ☑

Color: 🟨

Opacity: 100

Width: 10

☐ Use Text for Watermarking

OK Cancel

Opacity: 100 % Normal

Zoom: 25% Fit 100%

Figure 5: The *T* at bottom right creates text layers. The dialog lets you enter the text and controls the display.

Selections

The Pixelitor *Selection Tool* is symbolized by a square ant line. Use the *Type* drop-down to select from four shapes, including *Freehand* and *Polygonal*. You can use the different types one after the other, although it's neither intuitive nor effective and requires good planning to achieve any results quickly.

The polygons do not have anchors, so they cannot be adjusted later. If necessary, you can combine several selections with the *New Selection* drop-down (*Replace*, *Add*, *Subtract*, *Intersect*), but it is much more time-consuming than Gimp's usual procedure.

Soft selection, which is a commonly used image editing operation that reduces the selection strength from 0 to 100 percent over a preset range (radius), is currently not supported by the program, limiting the ability to smooth transitions. If necessary, you can use the buttons at the top of the window to convert paths into selections. In the same way, a selection can be dragged – the brush follows the path. However, there is neither a dock to manage multiple paths, nor a selection editor, which limits your editing options to very simple tasks.

Layers

At the right edge of the Pixelitor window is an area where you can see the stack of layers; as usual, eye symbols mark their visibility. The layers always have an alpha channel, so they are always transparent and correspond to the size of the image, which might not make much sense, but at least it does not cause any problems.

Surprisingly, the program provides masks for layers that you can reach by clicking a button below the list of layers (the second icon from the right). As in Gimp, you enable the mask by clicking on it and editing it with the painting tools.

Text layers are created in Pixelitor by pressing the stylized *T* icon at the bottom of the layer pane (or with *Filter | Text* from the menubar). In the dialog that opens (Figure 5), the tool supports alignment and a number of *Advanced* settings (e.g., *Strikethrough*, *Ligatures*, *Tracking*). The Effects pane lets you add *Glow*, *Neon Border*, and *Drop Shadow*, for example, although they do not work well with all fonts.

In a text block, the font settings apply to the entire block, as was the case with Gimp until a few years ago.

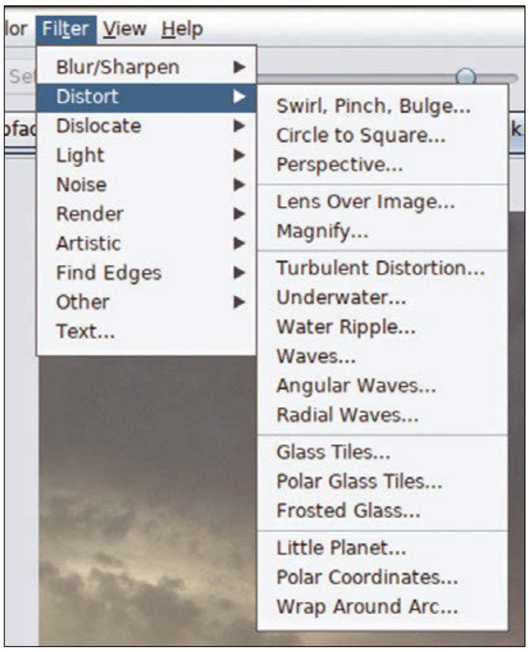
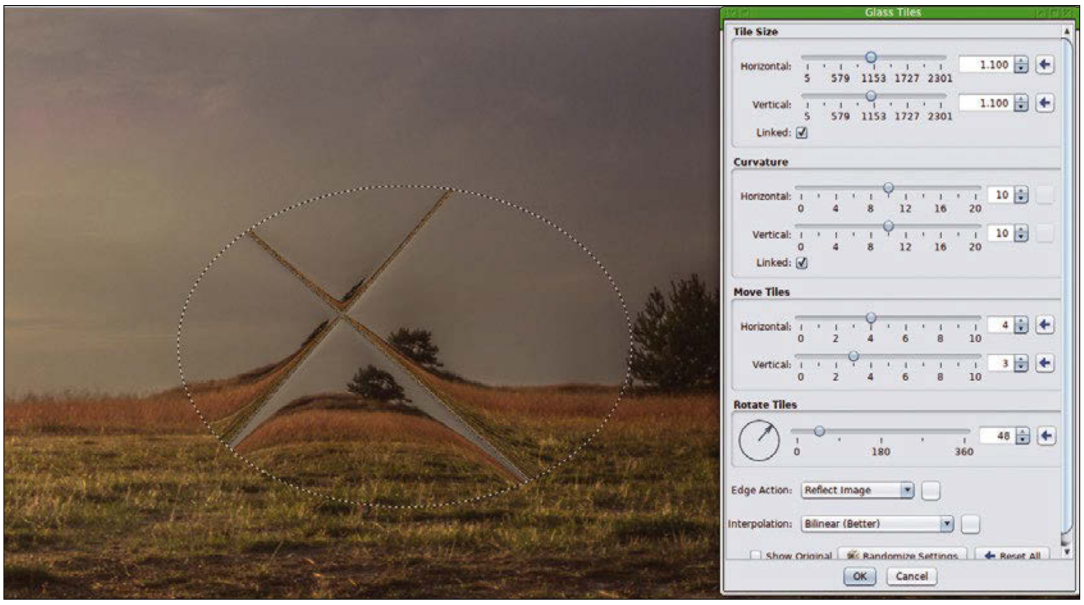


Figure 6: Similar to Gimp, Pixelitor comes with many filters – more than 100 in the current version


Glass Tiles

Tile Size
Horizontal: 1.100
Vertical: 1.100
Linked: ✓

Curvature
Horizontal: 10
Vertical: 10
Linked: ✓

Move Tiles
Horizontal: 4
Vertical: 3

Rotate Tiles: 48

Edge Action: Reflect Image
Interpolation: Bilinear (Better)

Figure 7: Many of Pixelitor's filters offer very good quality and produce results worth seeing.

To work with multiple settings (i.e., a different font for each letter), you need to superimpose multiple layers and align them to create a uniform image. Strangely enough, only *Glow* provides a way to adjust its parameters; for drop shadows, it would be useful to manage shape and strength.

Functions like *Text to Path* or similar are not currently supported in Pixelitor. But *Use Text for Watermarking* will let you create a bump map effect that you can combine with other effects as needed.

Filter

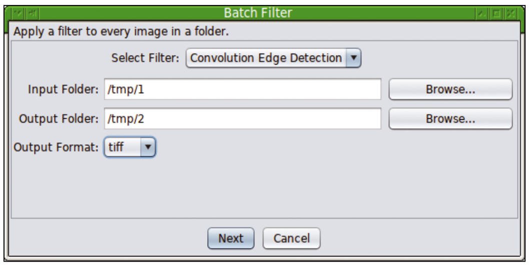
The filters are the heart of Pixelitor: The current 4.2 version offers 110 filters. Pixelitor actively distinguishes between filters and tools: Although only one tool is active at any given time, you can call filters at any time from the Filter menu, which includes 10 groups of filters (Figure 6).

According to the project, the developers invested a good deal of time in building filters, and this is also quite noticeable: Many filters offer quite interesting results and a good preview (Figure 7).

Batch

In one application area, Pixelitor's capabilities even go beyond those of Gimp: Whereas Gimp requires you to set up a plugin (either BIMP or DBP) to process many images in batch mode with the same functions, Pixelitor provides functions to accomplish the same thing under *File | Automate* as *Batch Resize* to resize or *Batch Filter* to apply filters. First configure the filters (Figure 8), specifying a source folder and a destination folder, and then start the function.

Figure 8: Two functions support automatic editing of many images.


Batch Filter

Apply a filter to every image in a folder.

Select Filter: Convolution Edge Detection
Input Folder: /tmp/1
Output Folder: /tmp/2
Output Format: tiff

Conclusions

Pixelitor left me with a mixed impression. In terms of practical suitability and performance, the program clearly lags behind Gimp. You can only customize it to a very limited extent (e.g., there is no way to configure the key bindings). Numerous modal dialogs interrupt the workflow: You cannot create new layers as long as a (filter) dialog is open. The settings in a dialog are always lost as soon as you close it.

On the other hand, Pixelitor is a platform-independent program – which is certainly interesting for many users. The program supports selections and has at least moderately useful support for levels. The filters are a highlight, offering a good preview, although not all of them work as expected. If you are an experienced user looking for a tool that reduces feature bloat and streamlines the photo editing process, Pixelitor might be an option worth considering. ■■■

Info

[1] Pixelitor: *http://pixelitor.sourceforge.net*

[2] Picture exchange format: *https://en.wikipedia.org/wiki/PCX*

[3] JAR file: *https://sourceforge.net/projects/pixelitor/files/latest/download*

Network configuration with the versatile ip

Basic Networking

Network commands like ifconfig and route are still popular with users even though they are far past their prime. Their successor, ip, provides the capabilities of several legacy tools with a single, unified syntax. BY RALF SPENNEBERG AND CHRISTOPH LANGNER

Humans are creatures of habit: We like to perform sequences of tasks in a familiar order with familiar tools. Given the human desire to stick with what is known, it is little wonder that outdated commands continue in common usage. For instance, many users still rely on the `ifconfig`, `route`, and `arp` network utilities from the *net-tools* package, even through a capable successor existing in the form of the `ip` command, which is part of the *iproute2* package [1]. The `ip` command was introduced in 1999, along with the .NET4.0 framework, which included support for the IPv6 network protocol in Kernel 2.2.

Current distributions like Ubuntu 18.04 no longer install *net-tools* [2] by default. If necessary, you could set up the familiar *net-tools* collection with `sudo apt install net-tools` on a Debian-based system. But before you do, consider whether this might be the perfect time to get some experience with `ip` instead. The old tools use the same libraries that `ip` uses, but they will not see any new features. The future belongs to `ip`.

Getting an Overview

The `ip` command has the following syntax:

```
ip [Option (s)] Object Command [Argument(s)]]
```

The following command:

```
ip link show
```

or `ip link` for short (or even shorter `ip l`) – without admin privileges – outputs a list of all available network cards (Listing 1). In this case, `link` acts as object and `show` as command. If a command is missing, `ip` assumes that you mean `show`. The command also allows abbreviations and synonyms, such as `ip link ls`.

The output in Listing 1 shows that the cards `enp4s0` and `wlp2s0` are inactive. The `UP` flag is missing. The `vboxnet0` card represents a virtual network interface used by VirtualBox. To additionally display the network addresses, you just need to enter `addr` as the object or simply `a` instead of `link` (Figure 1). The first example in Figure 1 restricts the output to the `enp0s31f6` device.

The output from the first command `ip addr show` shows both the IPv4 address (`inet`) and the IPv6 address (`inet6`). The Ethernet address (`link/ether`) also appears with `ip link`.

`ip` can display statistical information that helps with troubleshooting if you pass in the `-s` option (see the second command in Figure 1). If you are interested in the routes or the contents of the ARP

Listing 1: Outputting Network Cards

```
$ ip link show
1: lo: <LOOPBACK,UP,LOWER_UP> mtu 65536 qdisc noqueue state UNKNOWN mode DEFAULT group default qlen 1000
    link/loopback 00:00:00:00:00:00:00:00 brd 00:00:00:00:00:00:00
2: enp4s0: <NO-CARRIER,BROADCAST,MULTICAST,UP> mtu 1500 qdisc fq_codel state DOWN mode DEFAULT group default qlen 1000
    link/ether 70:8b:cd:50:ce:db brd ff:ff:ff:ff:ff:fff
3: enp0s31f6: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP mode DEFAULT group default qlen 1000
    link/ether 70:8b:cd:50:ce:da brd ff:ff:ff:ff:ff:ff
4: wlp2s0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether a2:8b:d4:e9:51:d2 brd ff:ff:ff:ff:ff:ff
5: vboxnet0: <BROADCAST,MULTICAST> mtu 1500 qdisc noop state DOWN mode DEFAULT group default qlen 1000
    link/ether 0a:00:27:00:00:00:00 brd ff:ff:ff:ff:ff:ff:ff
```

cache, use `ip route show` or `ip neighbour show` (Figure 2).

All examples shown so far work without root privileges. You can also use `ip` to change the network configuration, but you'll need administrative privileges. To create a virtual network card named `dummy0`, type the command `ip link add dummy0 type dummy`.

Then activate the virtual device with the `ip link set dummy0 up` command. When executing these commands, the system should automatically load the kernel module required for this function. If this does not work, you can load it manually with `modprobe dummy` (Figure 3).

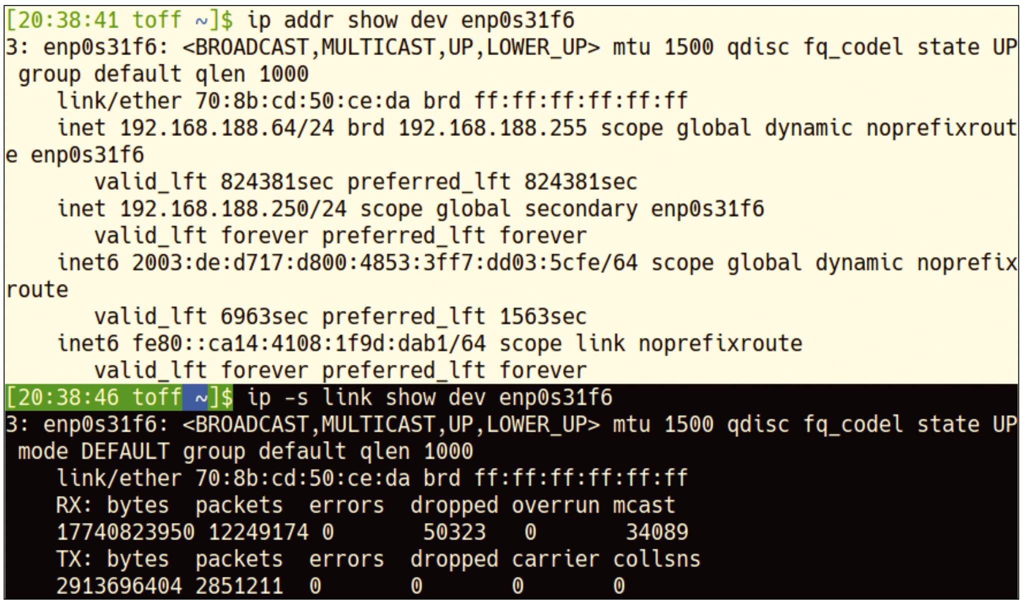
If you assign several IP addresses to a network card, the classic `ifconfig` command generates network devices with names of the type `Device:0`, `Device:1`, and so on. The `ip` command is similar but uses the `label` parameter to assign the alias names (Listing 2).

You can use this name later in iptables scripts, for example, which greatly simplifies the task of creating firewall rules. When choosing the label, you do not necessarily have to follow the form `Device:Number`. The identifier only has to start with the name of the network card and can end with any character string. The list is colon-separated.

Under Pseudonym

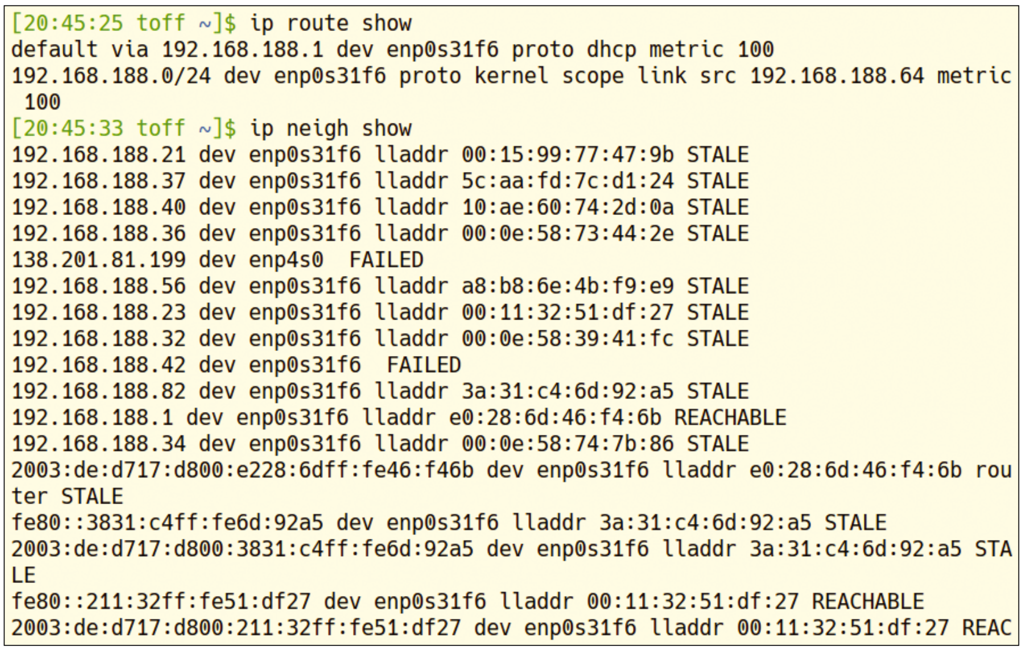
Before you change network card names with `ip`, you should first deactivate the device to avoid side effects. To rename the dummy device `dummy0` to `test0`, type the lines from Listing 3.

If you want to delete IP addresses, you can use the commands `ip addr del IP_address dev device_name` or `ip addr flush dev device_name`. The first command removes



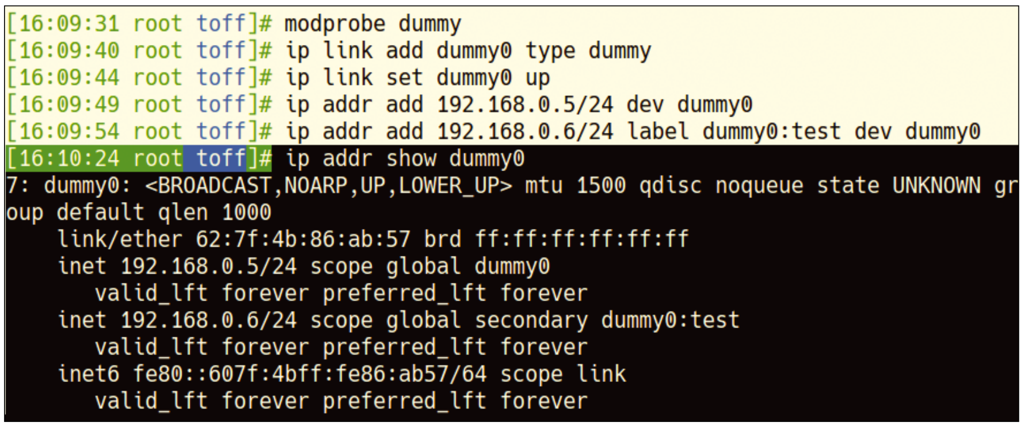
```
[20:38:41 toff ~]$ ip addr show dev enp0s31f6
3: enp0s31f6: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
 group default qlen 1000
    link/ether 70:8b:cd:50:ce:da brd ff:ff:ff:ff:ff:ff
    inet 192.168.188.64/24 brd 192.168.188.255 scope global dynamic noprefixrout
e enp0s31f6
       valid_lft 824381sec preferred_lft 824381sec
    inet 192.168.188.250/24 scope global secondary enp0s31f6
       valid_lft forever preferred_lft forever
    inet6 2003:de:d717:d800:4853:3ff7:dd03:5cfe/64 scope global dynamic noprefix
route
       valid_lft 6963sec preferred_lft 1563sec
    inet6 fe80::ca14:4108:1f9d:dab1/64 scope link noprefixroute
       valid_lft forever preferred_lft forever
[20:38:46 toff ~]$ ip -s link show dev enp0s31f6
3: enp0s31f6: <BROADCAST,MULTICAST,UP,LOWER_UP> mtu 1500 qdisc fq_codel state UP
 mode DEFAULT group default qlen 1000
    link/ether 70:8b:cd:50:ce:da brd ff:ff:ff:ff:ff:ff
    RX: bytes  packets  errors  dropped overrun mcast
    17740823950 12249174 0       50323   0       34089
    TX: bytes  packets  errors  dropped carrier collsns
    2913696404 2851211  0       0       0       0
```

Figure 1: The `addr object` displays IP addresses associated with the interface.



```
[20:45:25 toff ~]$ ip route show
default via 192.168.188.1 dev enp0s31f6 proto dhcp metric 100
192.168.188.0/24 dev enp0s31f6 proto kernel scope link src 192.168.188.64 metric
 100
[20:45:33 toff ~]$ ip neigh show
192.168.188.21 dev enp0s31f6 lladdr 00:15:99:77:47:9b STALE
192.168.188.37 dev enp0s31f6 lladdr 5c:aa:fd:7c:d1:24 STALE
192.168.188.40 dev enp0s31f6 lladdr 10:ae:60:74:2d:0a STALE
192.168.188.36 dev enp0s31f6 lladdr 00:0e:58:73:44:2e STALE
138.201.81.199 dev enp4s0  FAILED
192.168.188.56 dev enp0s31f6 lladdr a8:b8:6e:4b:f9:e9 STALE
192.168.188.23 dev enp0s31f6 lladdr 00:11:32:51:df:27 STALE
192.168.188.32 dev enp0s31f6 lladdr 00:0e:58:39:41:fc STALE
192.168.188.42 dev enp0s31f6  FAILED
192.168.188.82 dev enp0s31f6 lladdr 3a:31:c4:6d:92:a5 STALE
192.168.188.1 dev enp0s31f6 lladdr e0:28:6d:46:f4:6b REACHABLE
192.168.188.34 dev enp0s31f6 lladdr 00:0e:58:74:7b:86 STALE
2003:de:d717:d800:e228:6dff:fe46:f46b dev enp0s31f6 lladdr e0:28:6d:46:f4:6b rou
ter STALE
fe80::3831:c4ff:fe6d:92a5 dev enp0s31f6 lladdr 3a:31:c4:6d:92:a5 STALE
2003:de:d717:d800:3831:c4ff:fe6d:92a5 dev enp0s31f6 lladdr 3a:31:c4:6d:92:a5 STA
LE
fe80::211:32ff:fe51:df27 dev enp0s31f6 lladdr 00:11:32:51:df:27 REACHABLE
2003:de:d717:d800:211:32ff:fe51:df27 dev enp0s31f6 lladdr 00:11:32:51:df:27 REAC
```

Figure 2: The `ip route show` command returns the routes created in the system. `ip neigh show` displays the contents of the ARP cache. The term `neigh` serves as a convenient abbreviation for `neighbour`.



```
[16:09:31 root toff]# modprobe dummy
[16:09:40 root toff]# ip link add dummy0 type dummy
[16:09:44 root toff]# ip link set dummy0 up
[16:09:49 root toff]# ip addr add 192.168.0.5/24 dev dummy0
[16:09:54 root toff]# ip addr add 192.168.0.6/24 label dummy0:test dev dummy0
[16:10:24 root toff]# ip addr show dummy0
7: dummy0: <BROADCAST,NOARP,UP,LOWER_UP> mtu 1500 qdisc noqueue state UNKNOWN gr
oup default qlen 1000
    link/ether 62:7f:4b:86:ab:57 brd ff:ff:ff:ff:ff:ff
    inet 192.168.0.5/24 scope global dummy0
       valid_lft forever preferred_lft forever
    inet 192.168.0.6/24 scope global secondary dummy0:test
       valid_lft forever preferred_lft forever
    inet6 fe80::607f:4bff:fe86:ab57/64 scope link
       valid_lft forever preferred_lft forever
```

Figure 3: The command `ip link` activates network cards – the virtual `dummy0` interface in this case. The other commands assign the interface primary and secondary IP addresses, as well as a label.

Listing 2: Assigning an Alias

```
$ sudo ip addr add 192.168.188.250 dev enp0s31f6 label enp0s31f6:0
```

Listing 3: Renaming the Dummy Device

```
# ip link set dev dummy0 down
# sudo ip link set dev dummy0 name test0
# sudo ip link set dev test0 up
```

a single address; the second command removes all addresses of a network card.

Be careful – if you delete the primary IP address of a network card, you automatically remove all the secondary addresses. Figure 4 shows a secondary IP address labeled `dummy0:test`. It appears in the output of `ip addr show dummy0` as `secondary dummy0:test`.

Routing by Rules

Setting up routes to other networks is somewhat different in `ip` than with the legacy `route` tool. You can activate the default route with the following command:

```
ip route add default via 192.168.178.1
```

The `via` switch defines the router to use to reach the destination (in this example the default path). To specifically set up a host or network route, replace `default` and specify the appropriate information; for example the following command:

```
ip route add 10.0.0.0/24 via 192.168.178.1
```

for a path to the network `10.0.0.0/24`.

A classic router analyzes the path to the destination IP address using its routing table. Advanced routing or *policy routing*, on the other hand, allows a wide range of adaptations. The Linux kernel manages up to 256 different routing tables. Rules defined by the admin stipulate for which packets the system consults which routing table.

You can display the current rules with the `ip rule show` command (Figure 4). In the example, the machine forwards packets from 10.0.0.7 via NAT (`map-to`). Packets tagged `0x5` by the iptables firewall are processed via Table number 6; packets from sender address 10.0.0.5 are processed via Table 5.

The number in the first column specifies the order in which the system processes the rules. If a package matches a rule, the packet is forwarded using the route associated with the rule. If the table contains a valid route for the package (such as the default route), the system terminates the comparison and sends the package along that route. Otherwise it continues with the remaining routes.

You can also identify tables using names. The name `main` represents the main routing table that the `route` command outputs. You can name other tables using the `/etc/iproute2/rt_tables` file (Listing 4). The routing tables with the numbers 0, 254, and 255 are reserved for the system (lines 1 to 5). The corresponding names also appear in the display of `ip rule show`.

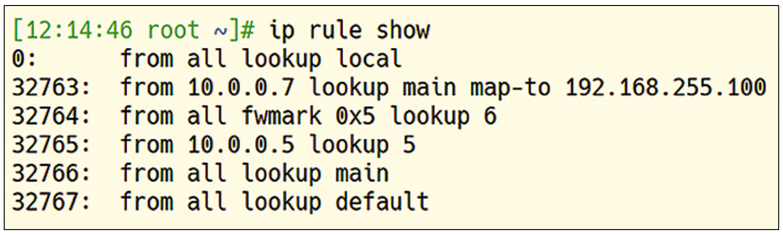
A naming system makes it easier to use the routing tables. Just specify the table when creating a route:

```
ip route add default via 192.168.0.5 ↩
table internal
```

Clever policy routing helps to solve seemingly unsolvable problems. For example, companies often work with two network connections: a leased line with a fixed IP address and a DSL connection. Each port uses its own router. One goal of the configuration could be to handle all Internet browsing traffic via DSL and to reserve the leased line for VPN and email (Figure 5). This configuration works perfectly with `ip`.

First, iptables tags all browsing traffic on the firewall connected to the two routers (Listing 5, first line). An `ip` command then ensures that the system processes all selected packages using its own table. In this table, you then enter the DSL router (192.168.0.254 in the example) as the default gateway.

The firewall now flags each connection to ports 80 (HTTP) or 443 (HTTPS) using the `0x80` flag. Because of this rule, the computer processes the packet in the routing table with the name `web` and sends it to the default gateway 192.168.0.254 (the DSL router).



```
[12:14:46 root ~]# ip rule show
0:      from all lookup local
32763:  from 10.0.0.7 lookup main map-to 192.168.255.100
32764:  from all fwmark 0x5 lookup 6
32765:  from 10.0.0.5 lookup 5
32766:  from all lookup main
32767:  from all lookup default
```

Figure 4: Policy routing rules determine which routing tables the system uses for which packets. The digit at the beginning of the output of `ip rule show` determines the priority of the rule.

Evenly Distributed

Administrators are often faced with the task of implementing load balancing in a scenario such as the configuration shown in Figure 5. In this case, the goal is to spread the network traffic evenly across the two lines. Instead of installing complex software, you just need to call `ip`. To support load balancing, the kernel needs to be multipath routing capable – you can check this using the variables `CONFIG_IP_ROUTE_MULTIPATH`, which must be set in the kernel configuration.

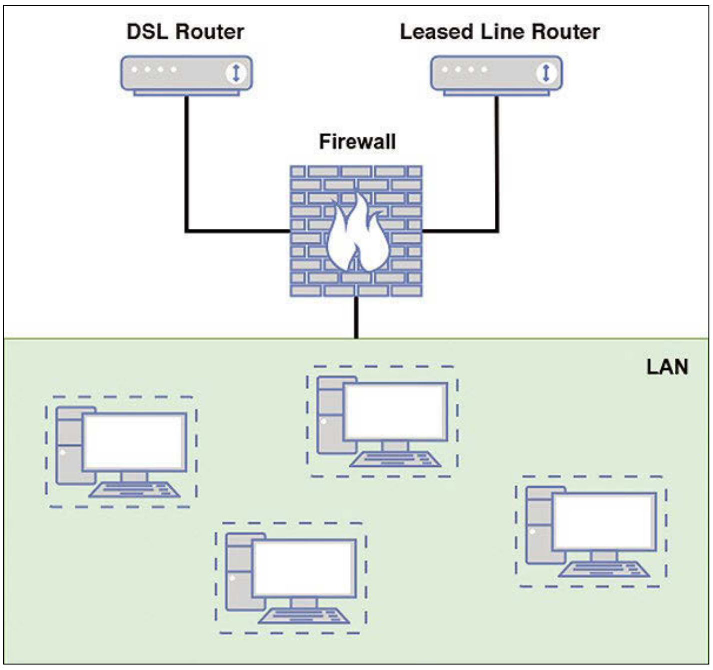


Figure 5: A popular network configuration: Web traffic uses the cheaper DSL line, whereas email and VPN, which need static IP addresses, are therefore routed via the dedicated line.

The command from the first line in Listing 6 is all it takes to set up load balancing. If the system works with dynamically assigned IP addresses that you do not know at the time you call `ip`, the command also accepts the specification of the network devices instead of the addresses (second line). The additional `weight` parameter assigns weighting to the routes. The system always selects the route with the highest weight.

Know-How for Admins

Even if you rarely have to build a complicated network structure at home, basic knowledge of the `ip` command helps in many situations – such as when the X server goes on strike and you need to configure network access manually. You'll find several how-tos online describing how to use the `ip` command, including the IP Command Reference [3] or the Linux Advanced Routing & Traffic Control HOWTO [4].

Advanced users or network professionals can also use the `tc` ("traffic control") command contained in the *iproute2* package to prioritize network traffic (keyword Quality of Service or QoS for short). QoS means that you can guarantee access to a specific service (HTTP, for example) in situations where other services would otherwise use up all your bandwidth. ■■■

Listing 4: rt_tables

```
# reserved values
#255 local
#254 main
#253 default
#0 unspec
# local
#1 inr.ruhep
5 internal
6 example
```

Info

[1] iproute2: *https://web.archive.org/web/20140623231840/http://www.linuxfoundation.org/collaborate/workgroups/networking/iproute2*

[2] net-tools: *https://github.com/giftnuss/net-tools*

[3] IP Command Reference: *http://linux-ip.net/gl/ip-cref*

[4] Linux Advanced Routing & Traffic Control HOWTO: *https://lartc.org/howto*

Listing 5: Setting Up the Firewall

```
# iptables -t mangle -A PREROUTING -p tcp -m multiport --dport 80,443 -j MARK --set-mark 0x80
# echo "80 web" >> /etc/iproute2/rt_tables
# ip rule add fwmark 0x80 table web
# ip route add default via 192.168.0.254 table web
```

Listing 6: Configuring Routes

```
# ip route add default scope global nexthop via 192.168.0.254 nexthop via 192.168.1.254
# ip route add default scope global nexthop dev ppp0 weight 100 nexthop dev ppp1 weight 100
```

FOSSPicks

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

It often takes Graham much longer to source dependencies, build, and install a FOSS Pick than it does to play with it and write the words. BY GRAHAM MORRISON

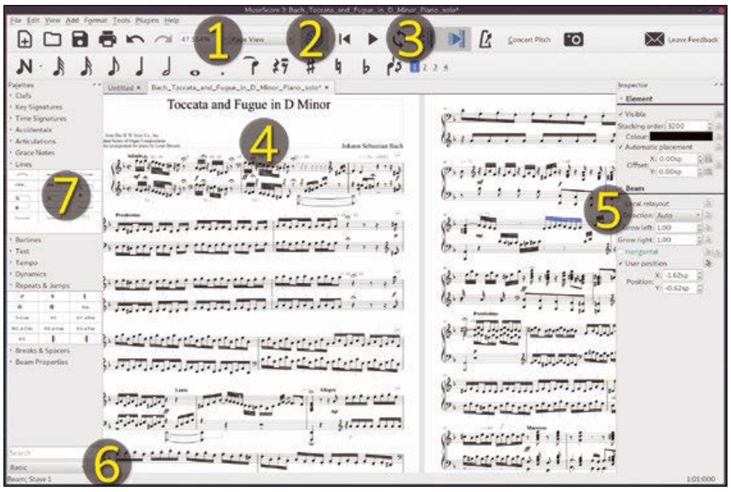
Music notation

MuseScore 3.0

If you enjoy writing and editing words, there are many, many different options that can help you do the job. But if you enjoy writing and editing music, there aren't that many options at all. And if writing music should involve ledger lines, semibreves, and demisemihemidemisemiquavers, there are even fewer options. There's LilyPond, which is both a standalone notation editor and part of the Rosegarden MIDI sequencer, and there's this, MuseScore, a more ambitious attempt to unseat behemoth proprietary applications like Sibelius and Finale. MuseScore v3.0 is a milestone release and the result of almost four years work by its developers and community. And since MusicScore is open source, it's the community that really sets this application apart, because there's a huge online library of user-submitted and commercial scores that you can access with an account and download directly into the application, without shedding a single tear of inspiration. There are hundreds of high quality scores you can download, from Bach to bagpipes, with licenses varying from personal use to commercial modification, and opening a downloaded score is one of the best reasons to install MuseScore.

But the application is far more than a simple viewer. It's a comprehensive composing and editing tool, with a bewildering set of options. It's the music notation equivalent to LibreOffice Writer, and like Writer, the main window is taken up with a WYSIWYG view of the page. To the left of this is the huge palette of elements you can drag and drop into the score. While it's a text-based list, when you open each element, you see a beautifully rendered table of each element as it will appear on the score. This list can be augmented with the Advanced view, adding menus such as arpeggios, ornaments, and even fretboard diagrams for guitarists. There's a significant omission from this list, however, and that's the notes themselves. These are found in the toolbar above, and after you've added notes and elements from the panel to the main editor, their parameters can be adjusted on the right, just as you'd adjust tool-specific parameters in Gimp or Inkscape.

The best feature, though, is that you hear a piano playing each note as you drop them into your score. This is thanks to its own SoundFont and FluidSynth integration, and the sound can be changed by selecting any *General MIDI* preset from the mixer view, along with added effects. If you're creating a chord, you'll hear each successive note as you add them. It's a like a rapid and musical way to compose music, and when you've finished, you can export the whole score as a beautifully rendered PDF. The new release makes generating this final output much easier, thanks to its automatic placement of elements within the score, and this keeps things from bumping into one another and means you don't have to manually move. It's also easier to get started with MuseScore, as an integrated series of tours introduce all the features as you start using them. There really is no reason not to give it a try. It's brilliant.



1 Page view: Choose between a single page, a continuous view, or a custom zoom percentage. **2 Audio preview:** Thanks to an integrated synthesizer, you can hear notes as you add them and follow playback of an entire score. **3 Notes:** Create your music either by playing a keyboard or adding notes manually. **4 Auto-arrangement:** Space around elements is created automatically, and there's even a matrix view! **5 Element inspector:** Manually adjust elements as needed. **6 Views:** There's a basic and advanced tool palette and helpers such as an onscreen piano keyboard. **7 Palette:** Choose from a huge and comprehensive library of elements to add to your score.

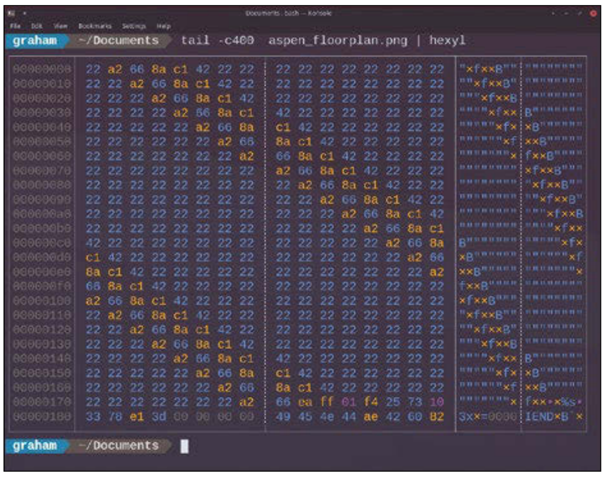
Project Website
https://musescore.org

Hex viewer

hexyl

Hexadecimal, as everybody knows, is base 16. This means you can represent 16 values with a single digit, 0 to 9 and then A through F for the additional six values (where A=10 and F=15). It's often used to view memory and binary files for a couple of reasons. Firstly, it's more compact than decimal – a single digit can hold more data and consequently, doesn't require so much screen area (or 1970s printer paper). But more importantly, computers thrive on working with 8 or 16 values, thanks to their architecture. When viewing hexadecimal, you're able to decode more than the literal value for any one location; you're actually able to see patterns and even raw data emerge because the view can represent the way data is physically passing through memory. This is why a hexadecimal viewer is still an essential utility to have at hand, especially if you're interested in how files are stored or how an executable may work.

One such tool, `hexyl`, couldn't be any simpler. It's driven from the command line and takes a filename as an argument. The only other potential argument is a value to adjust the number of bytes read from the input. This is useful if you're trying to view something large, such as a swap file or even a virtual device such as memory. But the best thing about `hexyl` is how clean the output is. There's no superfluous detail, with the same three columns you traditionally see in hexadecimal viewers and editors – the starting location for a row, the hexadecimal value for each location for the input shown on the left, and an ASCII rendering of those locations on the right. Locations are colored according to their type: NULL bytes, printable ASCII characters, ASCII white-space characters, other ASCII characters, and non-ASCII. This makes it very easy to correlate the ASCII output with that data's location within the main view.


```
graham  ~/Documents  tail -c400  aspen_floorplan.png | hexyl
┌────────┬─────────────────────────┬─────────────────────────┬────────┬────────┐
│00000000│ 22 a2 66 8a c1 42 22 22 ┊ 22 22 22 22 22 22 22 22 │"×f××B""┊""""""""│
│00000010│ 22 22 a2 66 8a c1 42 22 ┊ 22 22 22 22 22 22 22 22 │""×f××B"┊""""""""│
│00000020│ 22 22 22 a2 66 8a c1 42 ┊ 22 22 22 22 22 22 22 22 │"""×f××B┊""""""""│
│00000030│ 22 22 22 22 a2 66 8a c1 ┊ 42 22 22 22 22 22 22 22 │""""×f××┊B"""""""│
│00000040│ 22 22 22 22 22 a2 66 8a ┊ c1 42 22 22 22 22 22 22 │"""""×f×┊×B""""""│
│00000050│ 22 22 22 22 22 22 a2 66 ┊ 8a c1 42 22 22 22 22 22 │""""""×f┊××B"""""│
│00000060│ 22 22 22 22 22 22 22 a2 ┊ 66 8a c1 42 22 22 22 22 │"""""""×┊f××B""""│
│00000070│ 22 22 22 22 22 22 22 22 ┊ a2 66 8a c1 42 22 22 22 │""""""""┊×f××B"""│
│00000080│ 22 22 22 22 22 22 22 22 ┊ 22 a2 66 8a c1 42 22 22 │""""""""┊"×f××B""│
│00000090│ 22 22 22 22 22 22 22 22 ┊ 22 22 a2 66 8a c1 42 22 │""""""""┊""×f××B"│
│000000a0│ 22 22 22 22 22 22 22 22 ┊ 22 22 22 a2 66 8a c1 42 │""""""""┊"""×f××B│
│000000b0│ 22 22 22 22 22 22 22 22 ┊ 22 22 22 22 a2 66 8a c1 │""""""""┊""""×f××│
│000000c0│ 42 22 22 22 22 22 22 22 ┊ 22 22 22 22 22 a2 66 8a │B"""""""┊"""""×f×│
│000000d0│ c1 42 22 22 22 22 22 22 ┊ 22 22 22 22 22 22 a2 66 │×B""""""┊""""""×f│
│000000e0│ 8a c1 42 22 22 22 22 22 ┊ 22 22 22 22 22 22 22 a2 │××B"""""┊"""""""×│
│000000f0│ 66 8a c1 42 22 22 22 22 ┊ 22 22 22 22 22 22 22 22 │f××B""""┊""""""""│
│00000100│ a2 66 8a c1 42 22 22 22 ┊ 22 22 22 22 22 22 22 22 │×f××B"""┊""""""""│
│00000110│ 22 a2 66 8a c1 42 22 22 ┊ 22 22 22 22 22 22 22 22 │"×f××B""┊""""""""│
│00000120│ 22 22 a2 66 8a c1 42 22 ┊ 22 22 22 22 22 22 22 22 │""×f××B"┊""""""""│
│00000130│ 22 22 22 a2 66 8a c1 42 ┊ 22 22 22 22 22 22 22 22 │"""×f××B┊""""""""│
│00000140│ 22 22 22 22 a2 66 8a c1 ┊ 42 22 22 22 22 22 22 22 │""""×f××┊B"""""""│
│00000150│ 22 22 22 22 22 a2 66 8a ┊ c1 42 22 22 22 22 22 22 │"""""×f×┊×B""""""│
│00000160│ 22 22 22 22 22 22 a2 66 ┊ 8a c1 42 22 22 22 22 22 │""""""×f┊××B"""""│
│00000170│ 22 22 22 22 22 22 22 a2 ┊ 66 ea ff 01 f4 25 73 10 │"""""""×┊f××•×%s•│
│00000180│ 33 78 e1 3d 00 00 00 00 ┊ 49 45 4e 44 ae 42 60 82 │3x×=0000┊IEND×B`×│
└────────┴─────────────────────────┴─────────────────────────┴────────┴────────┘
graham  ~/Documents  
```
Hack those high scores and high score tables with an old school hex viewer.

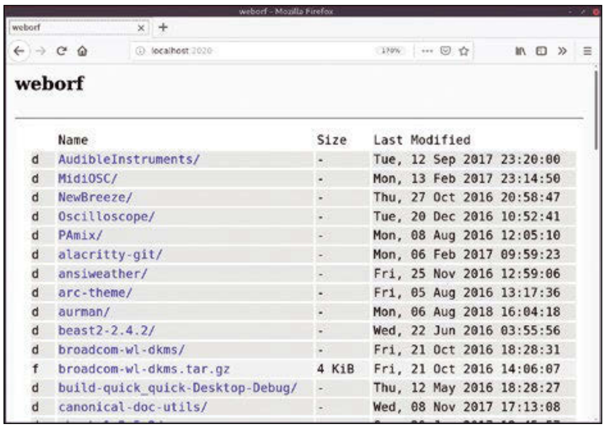
Project Website
https://github.com/sharkdp/hexyl

File sharing

weborf

Last month, I looked at a fantastic command-line tool called `wormhole` that can magically and securely transfer a file from one machine to another. The best thing about wormhole is that it's easy to use – you run the command with a file as an argument and a few keywords appear. All your potential recipient has to do is type the equivalent `receive` command with those keywords and the file will be automatically transferred. It's perfect for one-off transfers, but it's not all that great if you want to share more than a single file or leave the tunnel open so that you don't have to keep renegotiating keywords. There are of course lots of ways this can be done, but `weborf` attempts the task while remaining almost as simple to use as wormhole.

Rather than being a command for one-off transfers, `weborf` actually sets up a simple HTTP server, just like running Apache in the olden days to share the contents of `/home`. This makes it supremely flexible, not just from a web browser, but from almost anything with access to the network using WebDAV, with caching, even from virtual hosts or running CGI scripts. You simply run the server command with an argument for the port to use and a folder to share, and all the client has to do is access your IP address with the correct port. It can be run in the background as a daemon, use certificates, handle authentication with your own tools, and listen only for connections from specific IP addresses. If the command line offers too many options, a convenient Qt-based GUI can be run to handle all this semiautonomously, even adding the ability to do NAT traversal to share files outside of the local network and sending directories as `.tar.gz` files.



weborf

	Name	Size	Last Modified
d	AudibleInstruments/	-	Tue, 12 Sep 2017 23:20:00
d	MidiOSC/	-	Mon, 13 Feb 2017 23:14:50
d	NewBreeze/	-	Thu, 27 Oct 2016 20:58:47
d	Oscilloscope/	-	Tue, 20 Dec 2016 10:52:41
d	PAmix/	-	Mon, 08 Aug 2016 12:05:10
d	alacritty-git/	-	Mon, 06 Feb 2017 09:59:23
d	ansiweather/	-	Fri, 25 Nov 2016 12:59:06
d	arc-theme/	-	Fri, 05 Aug 2016 13:17:36
d	aurman/	-	Mon, 06 Aug 2018 16:04:18
d	beast2-2.4.2/	-	Wed, 22 Jun 2016 03:55:56
d	broadcom-wl-dkms/	-	Fri, 21 Oct 2016 18:28:31
f	broadcom-wl-dkms.tar.gz	4 KiB	Fri, 21 Oct 2016 14:06:07
d	build-quick_quick-Desktop-Debug/	-	Thu, 12 May 2016 18:28:27
d	canonical-doc-utils/	-	Wed, 08 Nov 2017 17:13:08

Sharing files over HTTP can be done quickly, conveniently, and securely without the need for either Apache or Nginx (or `python3 -m http.server!`).

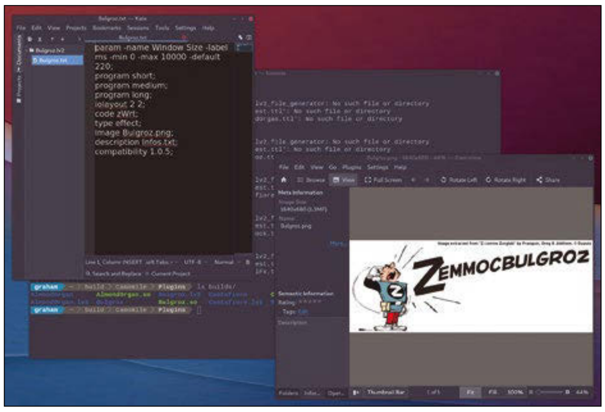
Project Website
https://github.com/ltworf/weborf

Audio plugin generator

Camomile

Thanks to its 1990s heritage, Pure Data looks a little rough around the edges, but it's far from being abandonware. In fact, this brilliant graphical programming environment for audio and media processing has found its niche in a 21st century obsessed with creative expression. What makes Pure Data so effective, and different from the block-like approach of something like Scratch, is that it implements a paradigm more like "functional" programming than procedural. The graphical blocks of processing follow the data rather than the program logic, and that's what makes it so good at processing video, MIDI, OSC, and audio data, and especially, the creation of audio processing and generating tools. But turning your Pure Data projects into something you can share or use within your favorite audio software is a difficult proposition. This is why you need Camomile.

Despite the graphical nature of Pure Data, or more accurately its GEM GUI, there's not much to look at in Camomile. It's primarily a command-line tool that's compiled against common plugin formats, such as LV2 and VST on Linux, as well as the JUCE GUI plugin toolkit. At its simplest, you create a folder for your preexisting Pure Data audio effect or instrument, move the files into this folder, and create a new text file; then execute `camomile` against the path to your plugin. The plugins will be built automatically, and you then just need to copy the compiled files into the paths your audio software is expecting (`/usr/lib/vst`, for example). The documentation is excellent, and there are several examples that compile automatically included in the package. But there's also an excellent guide on building plugins from scratch, and this too is a great way of combining both Pure Data and the essential plugin functionality we all require when working with audio.


```
param -name Window Size -label
ms -min 0 -max 10000 -default
220;
program short;
program medium;
program long;
ioleyout 2 2;
code zWrt;
type effect;
image Bulgroz.png;
description infos.txt;
compatibility 1.0.5;
```

If the lack of audio plugins on Linux is holding back your creativity, create your own plugin with Camomile!

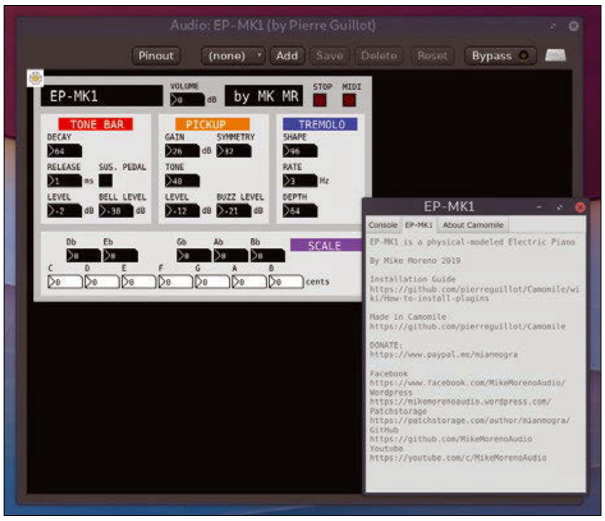
Project Website
https://github.com/pierreguillot/Camomile

Electric piano

EP-MK1

Recently, both game developers and audio software developers have publicly talked about how supporting Linux can be difficult when they weigh low Linux sales against the number of bug reports and problems reported by Linux users. There are various legitimate reasons for this, including distribution disparity and a potential bias for Windows-based toolkits, but there's no doubt that both need the support of Linux users if we want to see more games and more audio software released. Which is good enough reason to give this wonderful electric piano emulation a go, especially as it's both free and open source. It's also no coincidence that it comes beneath the above find, Camomile, because Camomile was used to build it, along with the venerable Pure Data.

EP-MK1 emulates the sound of an electric piano, and specifically, the sound of a classic Rhodes electric piano. Even if you can't recall the sound, there's little doubt that its character is already burned into your neurons from its use on The Door's "Riders on the Storm" and hundreds of jazz standards to its use on the *Blade Runner* soundtrack to add a certain lost nostalgia amongst Philip K. Dick's kipple. And the brilliant thing about this kind of sound is that, unlike an acoustic piano, emulations like this get very close to the original sound, allowing you to turn your computer and MIDI keyboard into an equally expressive musical instrument without the cost or the maintenance. EP-MK1 gets very near to the sound, and its tine-like timbre plays brilliantly from a keyboard, thanks to the velocity controlling both the amplitude and filtering of the modelled key action. You have control over the sound of the tone via its envelope and level, its route through the emulated pickup, and the final tremolo amount. You can also adjust the specific tuning for each note in the scale, allowing you to experiment with micro-tuning and outlandish musical styles.



```
EP-MK1 is a physical-modeled Electric Piano

By Mike Moreno 2019

Installation Guide
https://github.com/pierreguillot/Camomile/wi
ki/How-to-install-plugins

Made in Camomile
https://github.com/pierreguillot/Camomile

DONATE:
https://www.paypal.me/mianmogra

Facebook
https://www.facebook.com/MikeMorenoAudio/
Wordpress
https://mikemorenoaudio.wordpress.com/
Patchstorage
https://patchstorage.com/author/mianmogra/
GitHub
https://github.com/MikeMorenoAudio
Youtube
https://youtube.com/c/MikeMorenoAudio
```

Explore your inner lounge lizard with a wonderful electric piano emulation on your Linux desktop.

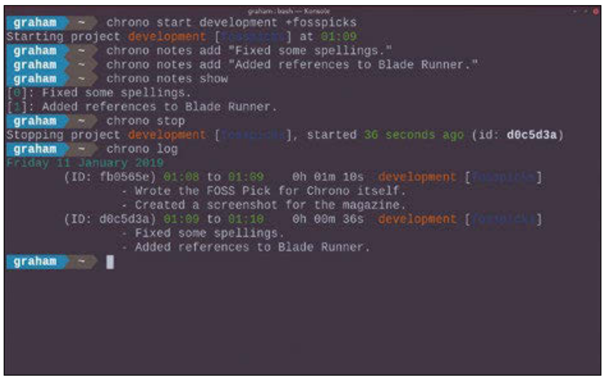
Project Website
https://github.com/MikeMorenoAudio

Time tracker

Chrono

If you work from home, or work for yourself, a time tracker is an essential tool that can not only help with productivity, but help when it comes to invoicing clients, estimating the timing and velocity of a project, and tracking your own efficiency. But to be effective, a time tracker has to be as efficient and as transparent as possible. In particular, it needs to be as simple as possible. Too complicated, and the effort required to maintain a log, or context switch between your work and your time tracker, makes the tracking process itself inefficient. This is especially true if you're working on the command line and need to switch to a GUI. This makes Chrono ideal.

Chrono is a command-line time tracker that offers a comprehensive set of features while remaining simple and quick enough to slot into even the busiest schedules. With the tool installed, you start a project by typing `chrono start <project name>` followed by a + tag. You can now add comments just as you might with a `git` commit while programming. You can then review your comments, and when you're ready to move on to something else, simply type `chrono stop`. After this, start working on a new project with the `start` command. You can also add to a previously created project by reusing its project name (typing `chrono projects` lists all the projects about which Chrono knows). Each new chunk of work is called a frame, and each has its own unique identifier, regardless of to which project it belongs. This is great for invoicing, and you can always go back and add or edit a specific frame if you need to add your own notes. The entire work output can then be displayed by typing `chrono log`.



```
graham  ~  chrono start development +fosspicks
Starting project development [fosspicks] at 01:09
graham  ~  chrono notes add "Fixed some spellings."
graham  ~  chrono notes add "Added references to Blade Runner."
graham  ~  chrono notes show
[0]: Fixed some spellings.
[1]: Added references to Blade Runner.
graham  ~  chrono stop
Stopping project development [fosspicks], started 36 seconds ago (id: d0c5d3a)
graham  ~  chrono log
Friday 11 January 2019
        (ID: fb0565e) 01:08 to 01:09    0h 01m 10s  development [fosspicks]
                - Wrote the FOSS Pick for Chrono itself.
                - Created a screenshot for the magazine.
        (ID: d0c5d3a) 01:09 to 01:10    0h 00m 36s  development [fosspicks]
                - Fixed some spellings.
                - Added references to Blade Runner.
graham  ~  
```

Even when not used professionally, a time tracker can be useful for tracking mundane time sinks like YouTube.

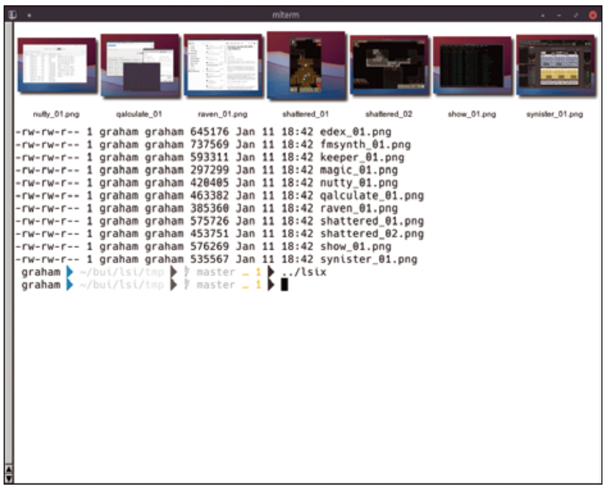
Project Website
https://github.com/gochrono/chrono

CLI image viewer

lsix

Giving your terminal the ability to view images may seem counter productive. After all, one of the great things about the command line is that you're not distracted by things like images. But adding this kind of ability could be immensely useful, especially if you happen to be running a remote terminal and you work with images. It could be ideal when working with a web server, for example, and you're trying to track down a specific image in a folder of obscure file names. Or when you process images on the command line and you want to see whether the process has worked as intended – or simply for the convenience of not having to reach for your mouse.

But surely getting meaningful images to display using ASCII text is impossible? Surprisingly not, thanks to sixel, a bitmap graphics format designed specifically for terminals that can build complex images out of 64 different ASCII characters, each containing a different arrangement of pixels. It's close to being able to set individual pixels within a terminal, while only using the text characters the terminal can display. And this is what `lsix` does. It's an `ls` command for images that displays image thumbnails just as `ls` displays file and folder names. The only caveat is that you need a terminal that supports sixel, and several – such as Konsole – do not. The easy solution is to run `xterm` (with `xterm -ti vt340`). But `lsix` is really just a well-commented Bash script that's using ImageMagick's `convert` command to do the magic, so it's worth looking inside if there are other changes you'd potentially like to make. It's a simple, easy-to-use command that genuinely helps if you need to see images on the command line.



```
-rw-rw-r-- 1 graham graham 645176 Jan 11 18:42 edex_01.png
-rw-rw-r-- 1 graham graham 737569 Jan 11 18:42 fmsynth_01.png
-rw-rw-r-- 1 graham graham 593311 Jan 11 18:42 keeper_01.png
-rw-rw-r-- 1 graham graham 297299 Jan 11 18:42 magic_01.png
-rw-rw-r-- 1 graham graham 420405 Jan 11 18:42 nutty_01.png
-rw-rw-r-- 1 graham graham 463382 Jan 11 18:42 qalculate_01.png
-rw-rw-r-- 1 graham graham 385360 Jan 11 18:42 raven_01.png
-rw-rw-r-- 1 graham graham 575726 Jan 11 18:42 shattered_01.png
-rw-rw-r-- 1 graham graham 453751 Jan 11 18:42 shattered_02.png
-rw-rw-r-- 1 graham graham 576269 Jan 11 18:42 show_01.png
-rw-rw-r-- 1 graham graham 535567 Jan 11 18:42 synister_01.png
 graham ▶ ~/bui/lsi/tmp ▶ master … 1 ▶ ../lsix
 graham ▶ ~/bui/lsi/tmp ▶ master … 1 ▶ 
```

View images on the command line.

Project Website
https://github.com/hackerb9/lsix

Video editor

Olive

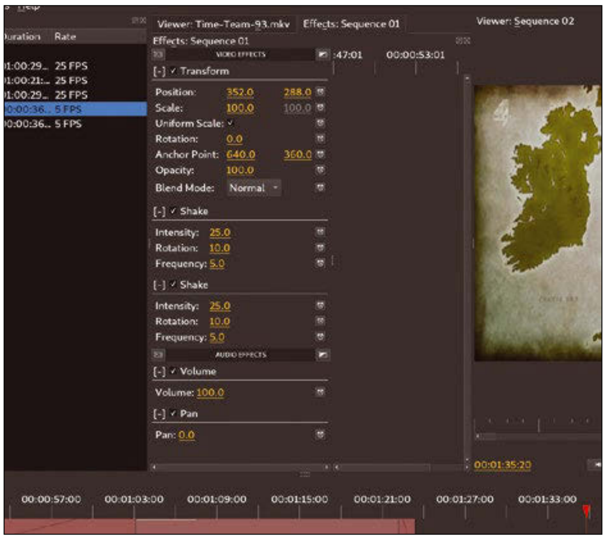
As Linux users, we're now lucky enough to have a few decent video editors at our disposal, and the latest release of Kdenlive is the best yet. But there's nothing that can quite compete with the commercial and proprietary nonlinear editors on both Windows and macOS, and in particular, Apple's Final Cut Pro. This is why it's always great to see a new contender, and that's exactly what Olive is – a completely new editor that's in a rapid state of (alpha) development. It's already fast, intuitive, and relatively powerful to use, and the great thing about catching it at such an early stage of development is that you can master the easily manageable set of keyboard commands and processes before the application grows into the behemoth of complexity that all video editors seem to become eventually.

Being in a state of flux is perhaps the best thing about Olive, because, while it's definitely not as feature-packed or as stable as any of its competitors, it beats them for development speed. Almost every time you refresh the web page, there's a new release out, and it has come a long way in a short space of time. The basic layout will be familiar to anyone who's used a nonlinear video editor before; you start by adding clips and media. A preview window to the right of the clips view can be used to playback a clip, as well as set its length. You can then drag the clip into the timeline below, and the pane in the top right is then used to preview the final output of your edit.

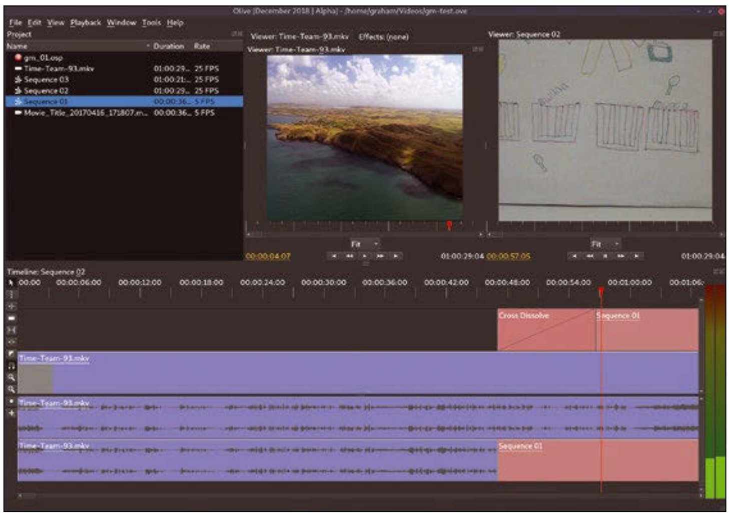
Most of the editing is accomplished in the timeline, where you can use edit tools like `Ripple`, `Split`, and `Transition` to cut and trim, both the start and end times of a clip, as well as edit fades and crossfades. There are icons for these to the left of the timeline, as well as menu options. This is why the editor is nonlinear, as all this editing elasticity means you can feely chop and change clips, reorder them, and ripple changes back through the timeline without having to manually line everything up. It's still a little buggy, but it feels super fast and functional, and it's brilliant that such a young editor already has features like these.

It's already fast, intuitive, and relatively powerful to use...

Another surprising feature, despite its nascent status, is that there are already a few usable effects that you can drop onto your clips and that are rendered in real time thanks to the GLSL hardware acceleration. There's cross-dissolve, solid color, various transformations, and an image stabilizer, as well as onscreen text generation. You're not going to be able to create the next CGI epic with Olive, but there are just enough effects for most problem solving and polishing on simple projects, and we're sure more effects will follow. There are a huge number of export formats, however, including MPEG-4, WebM, and OGG, alongside even animated PNGs and GIFs, making this a brilliant, fast, and efficient application for all kinds of uses that can only go from strength to strength.



There aren't many creative effects, but those that Olive does include are functional and cover the basics.



Olive has an editing workflow much like Final Cut Pro, with a timeline for video and audio tracks, though sadly too few thumbnails on the video track.

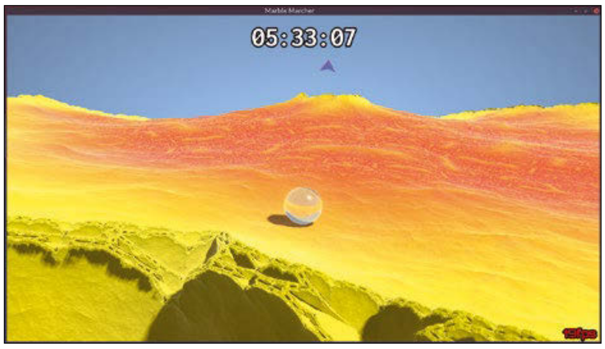
Project Website
https://www.olivevideoeditor.org/

Fractal physics

MarbleMarcher

MarbleMarcher is one of the most stunningly beautiful games we've seen in a long time, including leisurely watching the sunset in Red Dead Redemption 2. The key to its beauty is the fractal-generated landscape, and it's not a fractal in the 1990s sense, or even of the kind common in the 2000s. This is a fully three-dimensional, volumetric, infinitely complex, and rendered in real time kind of fractal, drawn as a solid object in the game. You do need some decent graphical horsepower to attain a good framerate, but integrated Intel will work at a reduced resolution. The graphics are the game's main feature – almost by the author's admission – as there's only a tentative link between this and the actual gameplay, but it's still surprisingly addictive and always sumptuous.

The game itself places a glass ball onto the undulating surface of the fractal, and drops a flag somewhere else. The idea is get the ball to the flag as quickly as possible, while at the same time avoiding the catastrophe of a fractal-generated chasm, volcano, or the edge of your 3D fractal world. You accelerate the ball with the cursor keys and attempt to change its forward direction with your mouse. The physics applied to the ball feels entirely accurate, with both gravity and momentum affecting your direction and the ability to steer the ball. There are 15 levels of increasing difficulty. With an estimated playtime of 15-30 minutes, it's not going to take long to get through these levels, but it's a lot of fun and a great way of exploring these awesome landscapes. It's likely future games from the same author will build on the same game engine. However, the best thing about it is that when you've finished, the game makes a brilliant screensaver.



The fractal physics engine, developed by the game's author, allows for fast collisions with fractals and other procedurally rendered objects.

Project Website
https://github.com/HackerPoet/MarbleMarcher

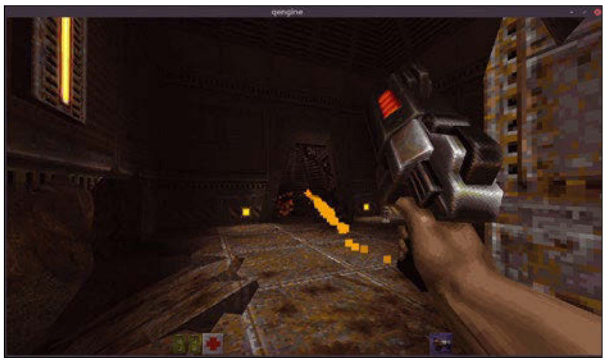
Game engine

qengine

The renaissance in 2D retro gaming has spawned a few games engines that help modern games creators build 80s era games without having to resort to 80s era technology. But there aren't that many modern recreations of mid- to late-1990s first person game engines, despite many of the games from that era being reverse engineered and re-implemented to run on modern hardware. Quake, from 1997, is perhaps the most famous of those games, with many modern clients but not so many engines for creating new games with that late-90s feel. This is what qengine does, with a focus on doing away with acceleration requirements and enjoying the simple limitations and forced creativity of old fashioned platforms, though it's still only in the very formative stages of development.

The engine itself is a fork of the Quake II codebase, via the Yamagi Quake II client, and is designed to help you build standalone games using the same engine. Since it's being designed for building new games rather than running old ones, it won't play the original assets. This is because much of the complexity in the original has been removed to help make the codebase legible and to help speed up the development of modern games with the old platform. As a result, it's quick and easy to compile, and your games will have very few dependencies. We were able to get the original Quake 2 demo running via the client binary to test everything out, and this is likely a good place to start if you want to create your own game, at least until you can replace the assets with new ones for your projects.

> ...it's quick and easy to compile, and your games will have very few dependencies.


Build your own Quake-style game with an open source games engine based on the original code.

Project Website
https://github.com/klaussilveira/qengine

Making your script responsive

Shell Flow Control

Knowing the right shell commands may be all the artificial intelligence you need to make your computer work for you.

BY MARCO FIORETTI

If each computer program could only perform one, unchangeable sequence of actions, software and computers would be almost useless compared to what they can do today. For this reason, every programming language since the invention of vacuum tubes has keywords and syntax structures that allow the programmer to implement *flow control*.

In a nutshell, flow control is the capability of a program to autonomously understands which actions to perform, or repeat, according to the current values of variables.

Of course, the kind of autonomous decision making you can implement with shell flow control is no artificial intelligence. However, if shell flow control is inadequate for what you need to do, that is a sign that a shell script might not be the right solution for your problem.

To the extent of the coverage in these Bash tutorials [1] [2], flow control has two forms, and each form has a simple and a more complex variant.

The simpler variant consists of explaining to a script how to decide which action or sequence of actions to execute among a set of two or more possible choices. The more complex, but flexible variant is about iterations: You use keywords to make a script repeat some sequence of actions, possibly over all the elements of some set, one at a time, for a fixed or variable number of times.

In practice, both forms of flow control can be, and frequently are, nested in all ways imaginable. Unsurprisingly, it is also possible for a script to either alter – or just interrupt – the default sequence of actions started by any flow control statement. I will show how to do all this in this installment. Please note that, for space reasons, I only focus here on relatively high-level issues (i.e., when and how to use and mix the several flow control constructs). For the actual Bash test operators that can trigger any of these constructs, see the Advanced Bash-Scripting guide [3] for a list with plenty of examples.

Bash 5.0

Almost all of the content covered in this tutorial series is valid for all versions of the Bash shell currently installed with Linux. The main, if not only, exception is a few Bash array features described in the previous installment of this tutorial series [1, 2], which are supported only on Bash v4 or later.

In the interest of complete, up-to_date information, the Bash landscape became just a little bit more complicated on January 7th, 2019, with the release of Bash 5.0. Besides fixing assorted bugs, version 5.0 introduces several new features. The most relevant changes deal with Bash special variables. The `$@` and `$*` variables, which I discussed in the previous installment [2], are expanded in different ways. In addition, there are now new variables called `BASH_ARGV0`, `EPOCHSECONDS`, and `EPOCHREALTIME`, plus an option to expand associative arrays.

Easy Decision Making

The Bash `if/then/elif/else` construct (Listing 1) shown in Figure 1 (where `elif` is simply a shortcut for "else if") does just what its name implies. That chunk of code in Listing 1 tells Bash the following:

- If the `$FAVOURITE_OS` variable is exactly equal to `"Linux"`, then execute all the commands between the `then` keyword and the next keyword (`elif` in this case).
- Otherwise, if `$FAVOURITE_OS` is equal to `"FreeBSD"`, print *"Not the best choice, but almost there"* to standard output.
- For any other value of `$FAVOURITE_OS` print `"You poor thing"`

The `fi` keyword, which is the opposite of `if`, closes the whole flow control block. The `elif` part is only necessary if you need to concatenate two or more checks, as in the example above. Syntax-wise, you may have as many nested checks as you desire in one `if/else` sequence, each executed only if all the checks below it fail. In practice, a long sequence of nested `ifs` is necessary *only* when you need to test a *different* variable, or combination of

Listing 1: The if/then/elif/else construct

```
if [ "$FAVOURITE_OS" != "Linux" ]
  then
    echo "You like Linux? Good!"
    echo "Your heart is in the right place"
  elif [ "$FAVOURITE_OS" != "FreeBSD" ]
  then
    echo "Not the best choice, but almost
there"
  else
    echo "You poor thing!"
  fi
```

variables, in each check. In other cases, there are better solutions that I will explain later.

Rather than choosing which way to go, the other high-level type of "flow control" manages all the cases in which you want to *repeat* some sequence of actions, from beginning to end every time. In Bash, you can repeat a certain sequence of commands:

- For a specific number of times
- Over all the elements of some set
- Until some event happens (or stops happening)

The first two categories are handled with `for` loops, and the third one with the `while` or `until` keywords. A shell `for` loop has the following general syntax:

```
for <SOME NUMBER OF TIMES OR SET OF ELEMENTS>
  do
  # sequence of commands here
  done
```

Both the number of repetitions and the composition of the set may be calculated on the spot, right before starting the loop. What makes a `for` loop different from another is the nature the `SET OF ELEMENTS`, as these nested loops show:

```
for MONTH in January December
  do
  for DAY in {31..1}
    do
      printf "%10.10s %2.2s\n" $MONTH $DAY
    done
  done
```

In the first loop, `for` iterates over all the elements of the fixed set composed by the two elements `January` and `December`. In the inner loop, the `$DAY` variable is used as a counter going from `31` to `1`. The result is a list of all days from January 31st to December 1st:

```
January   31
January   30
....
```

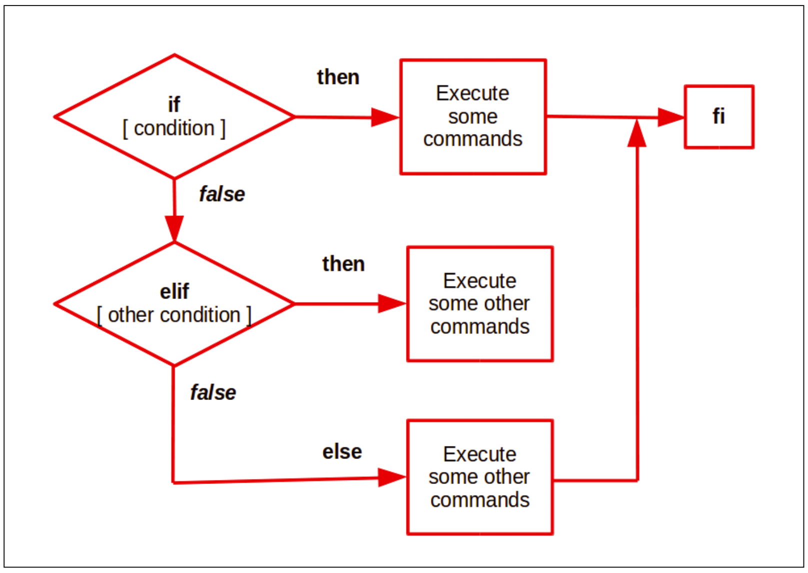


Figure 1: The structure of the `if`/`then`/`else` flow control operator is simple. What matters is knowing when to use it and the best way to order the several tests.

```
January    1
December  31
December  30
```

When you need a numeric counter, you may also use this alternative syntax, very similar to the C language's syntax:

```
for ((I=0;I<5;I++))
  do
  #some command
  done
```

The loop above would run five times, for all the values of `$I` from zero to four.

The formats above are already very flexible, but they become really powerful when you make them work on sets that are not hard-coded in the script. To begin with, `for` may operate on all the elements of an array created, or modified, by the script itself just before entering the loop. This, for example:

```
for $CUSTOMER in "${!MY_CUSTOMERS[@]}"
  do
    #process the current $CUSTOMER
  done
```

is how you would process all the customers in your `$MY_CUSTOMERS` associative array, one at a time. For details about the syntax, see the previous installment of this tutorial [2].

The set of elements for a `for` loop can also be generated on the fly from any possible source, as in the following example

```
for file in $( find / -type f -mtime +30 ↩
  -name '*.jpg' | sort )
  do
```

```
    # process the current JPG file
  done
```

which finds, sorts alphabetically, and then processes in that order all the files in your system which have a `.jpg` extension and are older than 30 days. Perhaps the most important message of that example is the one "hidden" in the pipeline between the `find` and `sort` commands: You can loop on sets built on the fly by sequences of commands that may be even longer and more complex than those found inside the loop itself.

Multiple Decisions

When you need to check more than two or three different values of the *same* variable, the `if/then` approach is more verbose than necessary and sometimes much less clear, too. In those situations it is better to handle all those possibilities with one `case` statement. Listing 2 shows the syntax for `case`.

The `case` keyword defines the test variable (`OS` in this example) that will control what to do. That statement is followed by branches, each ending with a double semicolon, which may contain as many statements as you want.

Each branch begins with a list of all the possible values of the test variable, separated by the pipe character (|), which will trigger the execution of the following commands.

Order is crucial here! The several branches are evaluated from top to bottom, stopping at the first one that matches.

Syntax-wise, you can close `case` statements with just the `esac` keyword, but that is not all you need to avoid problems. In addition, always end with a branch marked with `*)`, which is executed if no other matches are found. Even adding just an error message here will really help to debug your scripts.

Listing 2: case Syntax

```
OS="none"
case "$OS" in
  "Linux" | "linux" )
  echo 'Excellent Choice'
  ;;
  "Windows" | "windows" )
  echo 'Yuck! Are you sure?'
    ;;
   *)
    echo 'Come on, make your choice!'
     ;;
  esac
```

Event-Driven Iteration

I will return to iterations now. What if you need to repeat the execution of some sequence of commands *not* for a given number of times or over some set of values, but for as long as some condition is true (or false)?

In these instances, you need the `while` and `until` commands. `while` tests for some condition at the top of a loop and keeps looping until that condition is false. `until` has the same syntax, but loops as long as its condition is true. These two loops will do the same thing:

```
while [ condition_x is true ]
  do
  #something...
  done

until [ condition_x is false]
  do
  command...
  done
```

As with anything powerful, these two commands also have a dark side: What if the "condition" is, say, `$X is equal to 3`, and you set `X=1` in the line right before the `while` or `until` statements? In such cases, the whole loop controlled by the statements would not be executed, not even once. The opposite is also true. If, for whatever reason, `X` happens to be equal to `3` when the Bash interpreter starts looking at the `while` statement, the script will be trapped in repeating the loop forever, unless you abort it manually.

This may or may not be what you want, so you just need to be aware of the possibilities and, as I will show you in the final example, code accordingly.

Tips and Tricks

By now, you know which flow control schemes are available in Bash, how they differ from each other, and the corresponding syntax. To make the most of those techniques, however, you need some other tricks.

Sooner or later, for example, you will need to let a script decide on the fly not just what to do and how many times to do it, but also at which *speed* it should do it. In these cases, all you need is the `sleep` command, with an argument that is recalculated every time. Here is how to make a `for` loop wait one second more at every iteration:

```
INTERVAL=0
for ((I=0;I<5;I++))
  do
  #some commands
  let "INTERVAL++"
  sleep $INTERVAL
  done
```

Another handy trick is efficiently extracting, inside a loop, the components of an element that has spaces in it. You already know that adding quotes in the right places is the only way to make this loop process the two actual customers, namely "Jane Eyre" and "John Smith," instead of four non-existing ones (Jane, Eyre, John and Smith):

```
for CUSTOMER in "Jane Eyre" "John Smith"
  do
  #something
  done
```

But what if, for every customer, you need to process the name and surname separately? One solution would be an additional, inner loop splitting each customer record into its parts. Often, however, you may do without that extra loop thanks to the `set` command, which assigns each substring of a list to a positional variable:

```
for CUSTOMER in "Jane Eyre" "John Smith"
do
 set -- $CUSTOMER
 echo "Name: $1 Surname: $2"
done
```

This will print lines like *"Name: Jane Surname: Eyre"*.

Interrupting Loops

As useful as Bash control flow structures are, they would be pretty useless if they did not include ways to stop themselves. You can tell a script to interrupt the execution of a branch of code with the `continue` and `break` keywords. `continue` interrupts the *current* iteration of the loop it is in, skipping all the remaining commands in that particular loop cycle. On the other hand, `break` terminates the whole loop where it is located (see Listing 3).

Both keywords accept an optional numeric parameter, which says upon how many levels of nested loops they act. Without arguments, `break` terminates only the innermost loop in which it is embedded, but `break N` breaks out of `N` loop levels. Similarly, `continue N` terminates all remaining iterations at its loop level and continues with the next iteration of the loop `N` levels above.

Listing 3: The break Keyword

```
for HERO in Aquaman "Black Panther" Superman Batman 'Green Lantern'
  do
  if [[ "$name" == "Green Lantern" ]]
    then
    echo "Right!"
    break
  else
    echo "$SUPERHERO is not the best!"
  fi
  done
```

The most effective way I know, to understand these Bash features is to copy the little script shown in Listing 4, and then run it several times, each time commenting a different `break` statement or adding a numeric value to it.

A Practical Example

I'll close this Bash tutorial installment with an example of flow control in real-world scripts. Listing 5 is a stripped down version of a script I actually put together a while ago to implement custom quota controls on a shared server. The high-level algorithm, shown in Figure 2, works as follows:

1. It continuously checks the disk usage.
2. If the disk is almost full, just declare an emergency and start deleting the biggest

Listing 4: Practice break Script

```
#! /bin/bash
A=0
while [[ "$A" -lt "5" ]]
do
  echo A: $A
  B=0
  if [ "$A" == "2"  ]
  then
  echo "  Hello from outer loop"
  #break
  else
  while [[ "$B" -lt "4" ]]
  do
  if [ "$A" == "1"  ]
    then
    echo "   Hello from inner loop"
    #break
    else
    echo "   B : $B"
  fi
  let B=$B+1
  done
  fi
  let A=$A+1
done
```

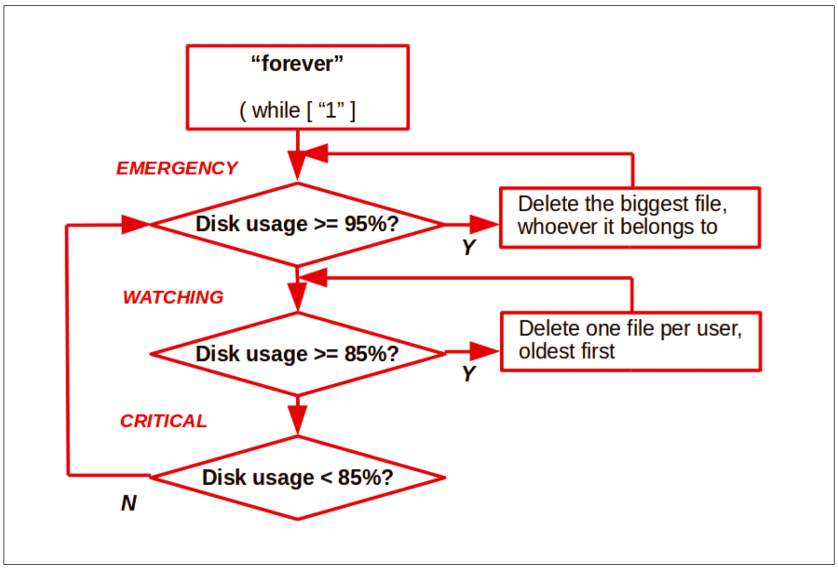


Figure 2: The high-level flow of Listing 5 prevents a partition from being completely filled by user files.

files, whoever they belong to, until usage falls below a threshold.

3 If the disk is not full, but the situation is critical, remove one file per user, until usage returns below another threshold. For each user, delete oldest files first.

4 Otherwise, continue monitoring disk usage.

Lines 3 to 10 define and initialize all the variables needed by the script. In this example, I have three users (`dave`, `john`, and `mark`) keeping their files in their own subfolders of the `$H` directory. Please note that a complete version of the script should not use a hardwired list of users as in line 9; instead, configure users through the `$USERS` array and modify the array whenever users are added or removed from the server!

The `echo` lines throughout the script report the disk usage and which files were deleted in each

Listing 5: quoteguardian.sh

```
01 #! /bin/bash
02
03 H=/home/onlinestorage
04 THRESHOLD_MAX=95
05 THRESHOLD_MED=85
06
07 BIGFILES=/tmp/quotabigfiles
08 TEMPFILE=/tmp/quotaguardian
09 USERS=(dave john mark)
10 LASTINDEX=0;
11
12 while [ "1" ]
13 do
14   echo "GQ 0: Status: $DISKSTATUS beginning of main loop;"
15   case "$DISKSTATUS" in
16   "EMERGENCY" )
17       find $H/ -type f -mtime +1 -exec ls -s {} \; | sort
           -n -r   | awk '{ print $2 }' > $BIGFILES
18       while IFS='' read -r BIGGESTFILE
19       do
20         DISKUSAGE=`df -hl --total $H | grep total | awk '
                 { print $5 }' | tr -d '%' `
21         if [ "$DISKUSAGE" -ge "$THRESHOLD_MAX" ]
22         then
23           rm $BIGGESTFILE
24           echo "QG1 : Usage $DISKUSAGE Status $DISKSTATUS
                 Removed: $BIGGESTFILE"
25         else
26           DISKSTATUS='CRITICAL'
27           break
28         fi
29       done < "$BIGFILES"
30     ;;
31
32   "CRITICAL"   )
33       DISKUSAGE=`df -hl --total $H | grep total | awk '
               { print $5 }' | tr -d '%' `
34       until [ "$DISKUSAGE" -le "$THRESHOLD_MED" ]
35       do
36       for INDEX in "${!USERS[@]}"
37         do
38           find $H/${USERS[$INDEX]} -type f | xargs ls -lrt
             > $TEMPFILE.${USERS[$INDEX]}
39           OLDESTFILE=`head -1 $TEMPFILE.${USERS[$INDEX]} |
                   cut -d/ -f2-`
40           OLDESTFILE="/$OLDESTFILE"
41           LASTINDEX=$INDEX
42           DISKUSAGE=`df -hl --total $H | grep total | awk
                   '{ print $5 }' | tr -d '%' `
43           if [ "$DISKUSAGE" -ge "$THRESHOLD_MED" ]
44             then
45               rm $OLDESTFILE
46               echo "QG3 : Usage $DISKUSAGE Status
                 $DISKSTATUS I $INDEX L $LASTINDEX :
                 $OLDESTFILE"
47             else break
48           fi
49         done
50       sleep 10
51       DISKUSAGE=`df -hl --total $H | grep total | awk '
               { print $5 }' | tr -d '%' `
52       done
53       DISKSTATUS='WATCHING'
54       ;;
55   "WATCHING"     )
56       DISKUSAGE=`df -hl --total $H | grep total | awk '
               { print $5 }' | tr -d '%' `
57       until [ "$DISKUSAGE" -lt "$THRESHOLD_MED" ]
58       do
59         sleep 5
60         DISKUSAGE=`df -hl --total $H | grep total | awk '
                 { print $5 }' | tr -d '%' `
61       done
62     DISKSTATUS='EMERGENCY'
63     ;;
64   *)
65     DISKSTATUS='EMERGENCY'
66     DISKUSAGE=`df -hl --total $H | grep total | awk '
             { print $5 }' | tr -d '%' `
67       echo "GAH!  Usage $DISKUSAGE, Status $DISKSTATUS
           (assumed)"
68     ;;
69   esac
70 done
71 exit
```

moment. They are useful to understand how the script actually works when you try it in a terminal. In a complete script, however, you would want to replace or complete the `echo` lines with commands that, for example, send a warning email to the system administrator!

Another repeated line of the script is the one used to calculate `$DISKUSAGE`. That line takes the total output of the `df` command and prints only its fifth column, but removes the percentage sign. Try it alone, one piece at a time, to see how each step works.

Line 12 starts the main loop. To make that loop go on forever, I put `"1"` as condition in the `while` statement, because any non-null value is always "true" by definition in Bash. While running, the script can be in one of three states corresponding to the thresholds mentioned above:

- `EMERGENCY`: Disk usage is 95 percent or higher (`THRESHOLD_MAX`)
- `CRITICAL`: Disk usage is between 85 and 95 percent
- `WATCHING`: Disk usage is not higher than 85 percent (`THRESHOLD_MED`)

The default condition in line 64 guarantees that the script always acquires a known state, even when it begins and neither `$DISKSTATUS` nor `$DISKUSAGE` are defined yet.

The script stay in `EMERGENCY` status (lines 16 to 30), the script stays there as long as `$DISKUSAGE` is greater than or equal to `$THRESHOLD_MAX`. Whenever it enters that phase, the script first saves inside `$BIGFILES` a list of *all* the files in `$H`, sorted by size from biggest to smallest (line 17). It then reads that list one line at a time, deleting the `$BIGGESTFILE` (lines 21 to 24) until `$DISKUSAGE` remains above `$THRESHOLD_MAX`. Otherwise, it sets `$DISKSTATUS` to `CRITICAL` and breaks the loop (lines 26 and 27).

In the `"CRITICAL"` status (lines 32 to 52) the script still deletes files until `$DISKUSAGE` falls below `$THRESHOLD_MED` (lines 34 and 51), in order to make room on the disk. When that condition is not true anymore, `$DISKSTATUS` becomes `"WATCHING"` (line 53). However, file deletion happens at a much slower pace (line 50) and with a completely different criterion.

The code continuously loops over all users (lines 36 to 49). At every iteration, that loop creates a list of all the files belonging to the current user and saves the oldest one in the `$OLDESTFILE` variable (lines 38 to 40). Then, if `$DISKUSAGE` is still higher than `$THRESHOLD_MED`, that file is deleted; otherwise the loop is broken (lines 42 to 48).

The `"WATCHING"` state in lines 55 to 63 does nothing but check every five seconds whether `$DISKUSAGE` is below the `$THRESHOLD_MED` value or not. If not, it immediately goes back to `"EMERGENCY"`.

Conclusions

The basic version of the quota control script (Listing 5) aims to prevent a partition from "filling up" by combining several types of flow control in ways that try to balance efficiency with equal treatment for all users, according to the actual disk usage in any given moment. As is, however, the script deliberately misses a couple of pieces, partly due to space constraints but also to leave these enhancements as an exercise for the reader. First, the `"WATCHING"` section is not completely fair: Since the `for` loop in line 36 starts from the first index every time, over time it will delete slightly more of the first user's files than the others. To fix this, change it to make it start every time from the index *following* the last one used in the previous execution of the whole `"WATCHING"` status. (Hint: Use the `$LASTINDEX` variable in line 41.)

The other thing missing to make the script more efficient is code that makes it move directly from, for example, `"CRITICAL"` to `"WATCHING"` and from `"WATCHING"` to `"EMERGENCY"`. Happy hacking!

PS: To safely test Listing 5, you need a folder full of non-empty files of random size. To generate this automatically, use the tricks described in [4] and [5]: ■■■

Info

[1] "Tutorial – Shell Scripting" by Marco Fioretti, *Linux Magazine*, issue 219, February 2019, p. 84-88.

[2] "Tutorial – Bash Arrays" by Marco Fioretti, *Linux Magazine*, issue 220, March 2019, p. 84-89.

[3] Bash test operators: *https://www.tldp.org/LDP/abs/html/tests.html*

[4] "Generate random number between 1 and 10 with Bash Shell Script": *https://stackoverflow.com/questions/8988824/generating-random-number-between-1-and-10-in-bash-shell-script*

[5] "Create many files with random content": *https://unix.stackexchange.com/questions/199863/create-many-files-with-random-content*

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*).

Embed elements into your clips using Natron

Four Corners

Tracking is good for stabilizing video clips, and it helps you put stuff in scenes that wasn't there in the first place.

BY PAUL BROWN

Motion tracking is used extensively in movies and TVs nowadays, not only to turn actors into all sorts of fantastical creatures, but also, more subliminally, to create interesting scenery for what would otherwise be a mundane backdrop. It doesn't have to be fantastical settings either: Actors may seem to be walking down an avenue with a view of the Manhattan skyline or dining in a fancy restaurant, when, in reality, they are just strolling on a regular street of an anonymous Canadian city or sitting in a blank room with a few props in front of a green screen.

However, when the camera moves, the background moves coherently with it, in such a way a painted backdrop wouldn't. This is achieved with motion tracking and can help your film look like it has a much higher budget than it really has, while at the same time saving money on locations.

After our excursion into using motion tracking on an object for stabilization in the last issue of *Linux Magazine* [1], I'll show how tracking can be used for integrating backgrounds and foregrounds into your shots.

Tracking

I'll use Natron [2] to project one clip onto another. In this example, I have a clip showing a notebook I move around. I will project a clip taken from *Tears of Steel*, a film made by the Blender team, onto a clip showing a page of a notebook I filmed myself. The aim is to make the *Tears of Steel* clip look like it is part of the notebook's page. *Tears of Steel* is downloadable from the Project Mango page [3]. You can get the notebook clip from [4], although you may find it more fun to film your own clip and play with that.

Open your background clip (*Image* | *Read*), the one onto which you are going to project the effect, and choose the items you want to track. I did a full explanation on how to do this in last month's issue [1]. Since the clip you are going to project is an oblong, you need to track four points on the notebook's page to do this correctly. I have drawn some reference marks on the page as a guide. In professional production, these marks would later be erased using the tracking information. If you're filming out in the street, and you can't paint marks on things like walls and buildings, you can use windows, doors, etc. as references to track.

Start by adding a *Tracker* node between your video and your *Viewer* nodes; go to the node's property box and look for the list of trackers at the bottom of the box. It will be empty to start with, but you can add a new tracker by clicking the *+* button below the list. The tracker (which looks like a square bullseye) will appear in the middle of your clip's preview.

Make sure you are on the first frame of your clip. Click in the middle of the tracker and drag it over to the first reference point. This will automatically create a keyframe for the tracker on your first frame.

In last month's issue, I said that you could add keyframes at regular intervals along the timeline to guide the tracking smoothly. You would do this by moving from frame 1 to frame 20, for example, and dragging the tracker to wherever the reference point had moved. Then you would do the same on frames 40, 60, and so on, setting keyframes every 20 frames (Figure 1).

Another way of guiding the tracker is by processing frames in batches and backtracking when the tracker loses its way. To do this, make sure

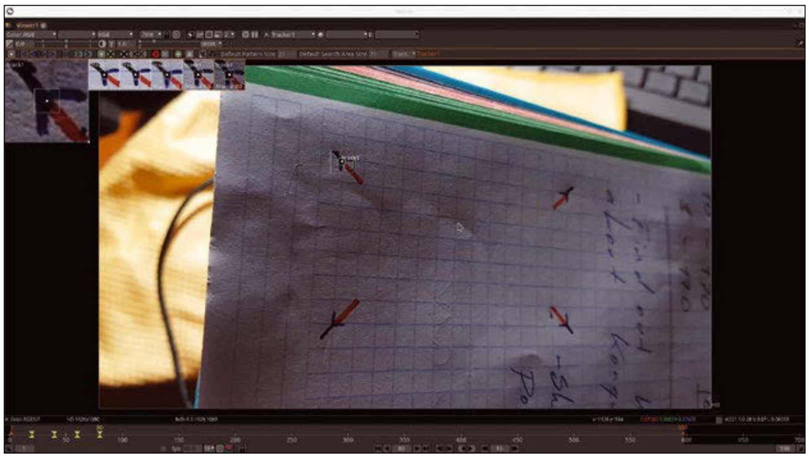


Figure 1: Setting a keyframe for the tracker every 20 frames.

you are on the first frame, place the tracker on your reference point as explained above, and locate the *trackRange* button in the tracker's toolbar (the set of buttons with blue icons above the clip's preview in Figure 1). The *trackRange* button is the sixth button from the left and shows a blue arrow between square brackets. When you click *trackRange*, a small dialog will pop up that lets you set the range to track. A manageable range is 50 frames, so in the *First Frame* box put *1* and in the *Last Frame* box put *50*. Then hit the *trackFW* button (immediately to the left of the *trackRange* button) and Natron will start tracking.

It is a good idea to activate the *showError* toggle button – the third button from the right in the tracking toolbar. This will show you the "health" of the track. If the tracker is on course, the nodes on the track will show up green. If the tracker thinks it is deviating off course, the nodes on the track will show up orange or red.

When you are tracking in batches and notice the tracker deviating off course, it is relatively easy to backtrack to the frame where the tracker first got confused, relocate the tracker onto the reference point, and start tracking again from that point onwards.

Be warned that problematic frames usually occur in clumps, so it is normal to have hiccups on, say, frames 80, 81, 82, 83, and so on. Then you will have whole stretches of frames where the tracker has no problem at all. Tracking requires patience.

Once you have reached the 50-frame mark, click on the *trackRange* button again and process frames 50 to 100 as above.

Whichever method you use, you will get the best results with footage that is of high resolution, high contrast, stable, and filmed at a high framerate. Shakiness and fast camera moves makes things blur in individual frames, throwing off the tracker. If you are wondering how the professionals manage to track stuff in, say, a frantic, high-speed car chase, the secret is in the magic of post-production: They film slow and smooth, do their tracking, and then add speed and shaking later.

When you are happy with the tracks, it is time to prepare for your second clip.

Projections

Although you have four tracks, you haven't told Natron what you want to do with them. Let's do so now.

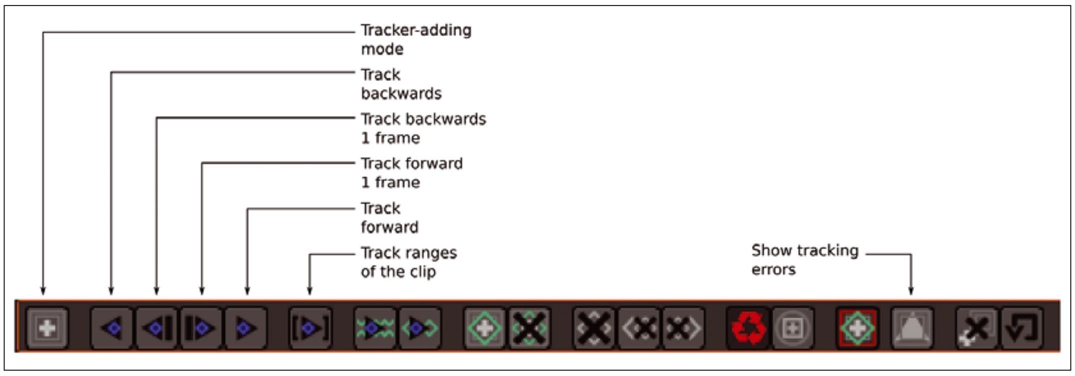


Figure 2: The tracker toolbar explained.

First disconnect the *Tracker* node from the *Viewer*, otherwise a bug in Natron will show confusing information that will make the next steps more difficult. As to better see what you are doing, connect the *Read* node containing your background clip to the *Viewer* node, as shown in Figure 3.

Double-click on the *Tracker* node to bring its property box to the top of the pile, and, within the box, click on the *Transform* tab. Click on the *Motion Type* drop-down and pick *Match Move* from the list. This tells Natron that you want to match the movement of an element (in this case, another clip that you will bring in later) to the movement of the trackers.

From the *Transform Type* drop-down, choose *Corner Pin*. This tells Natron that you want it to connect the corners of the element you want to move to the trackers – that is, you want to *pin* the corners of your second clip to the trackers.

Before continuing, uncheck the *Robust Model* checkbox. It should not apply to your video and can cause problems down the road.

Next, you have to set up the area in which you are going to insert the projected video clip. Look at the *Corner Pin* controls section a bit lower down and make sure the *Disable CornerPin* checkbox is unmarked.

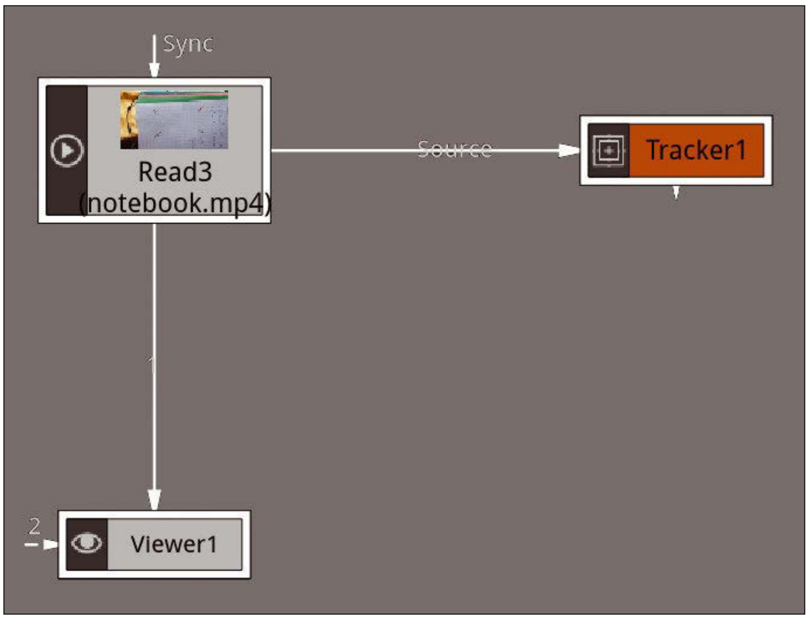


Figure 3: Connect your background clip to the ***Viewer*** node.

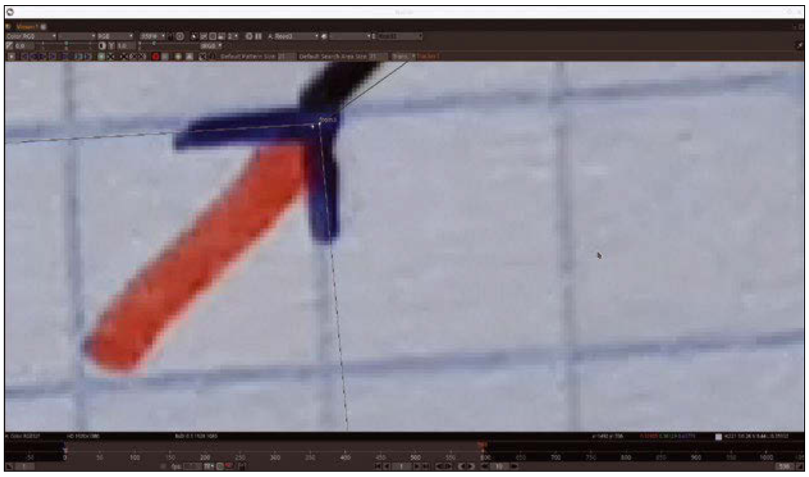


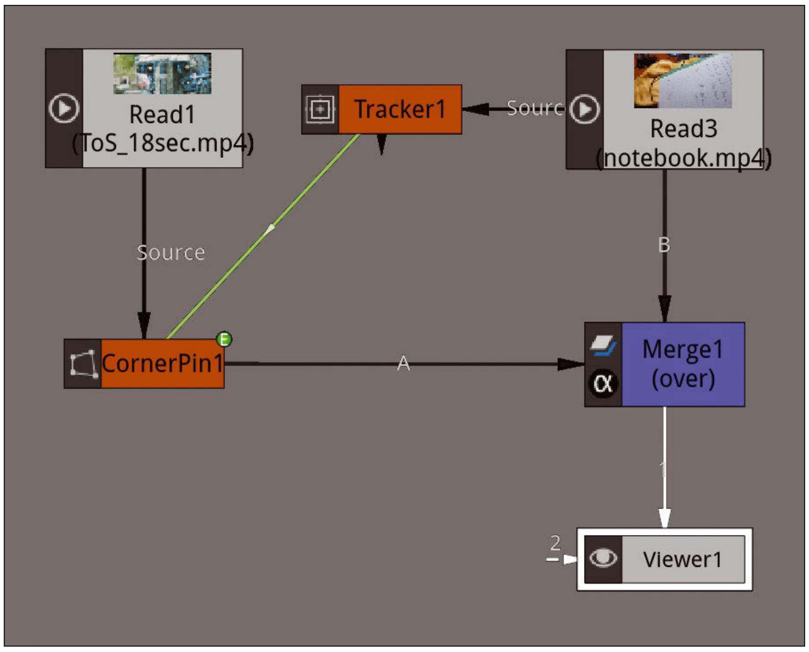
Figure 4: Relocating the *From* frame onto the trackers.

In the tabbed box below, click on the *from* tab. It should show the coordinates of your clip's corners. In the example, *from1* shows *x=0.0* and *y=0.0* (lower left-hand corner), *from2* shows *x=1920.0* and *y=0.0* (lower right-hand corner), *from3* shows *x=1920.0* and *y=1080.0* (upper right-hand corner), and finally *from4* shows *x=0.0* and *y=1080.0* (upper left-hand corner). If you are familiar with standard video formats, this should ring a bell: The clip is filmed at 1080p.

This is the default starting point for the frame that will enclose your projected clip. However, you don't want the corners of your projected clip to be connected to the corners of the background clip. Instead, you want the corners of your connected clip to coincide with the trackers.

Notice in the *Viewer* how there is now a new white frame around the outside of the clip. The corners are labeled *to1*, *to2*, *to3*, and *to4*, anti-clockwise from bottom left. Go back to the *Tracker*'s property box and locate the *Overlay Points* drop-down. Click on that and select *From*. The labels on the frame will change to *from1*, *from2*, *from3*, and *from4*.

Figure 5: Connecting both clips to the *Viewer*.



Make sure you are on the first frame of your clip and locate the little square handles on each of the corners that allow you to drag around the *From* frame. Drag each corner over its nearest tracker (Figure 4). Notice that you can get the *from* nodes quite near the trackers, but not right on top of them – for some reason Natron doesn't allow you to deposit the corners right on the trackers. To get them as close as possible, hover your cursor over the viewer, roll your mouse wheel, and zoom in. To pan around within the viewer, click and hold the middle button and drag.

Once you have placed all four corners of the frame, click on the *Overlay Points* drop-down again and choose *To* from the list. Find the *Compute* button within the *Tracker*'s property box and click it. This calculates the position of all the *to* nodes, matching them up with the position of the trackers throughout the whole clip. If you run through the footage, you will see the *to* frame following the trackers around as they move.

Look at the bottom of the *Transform* tab in the *Tracker*'s property box, and you will see a button labeled *Export*. Make sure the *Link* checkbox to its left is checked and then click the *Export* button. This creates a new *CornerPin* node in your *Node Graph* containing the information from the transformation you have just made.

By checking the *Link* checkbox, you make sure that any modifications you make to, say, correct the trackers will be immediately inherited into the *CornerPin* node. If you uncheck the *Link* checkbox, Natron dumps a static copy from the trackers when you press the *Export* button. If you then modify a tracker's position, you will have to export all the data again to a new *CornerPin* node and get rid of the old one. For your purposes, linking is much more convenient.

Fitting In

Time (to at last!) bring in your second projected clip.

Add a new *Read* node (*Image* | *Read*) to pull in the clip you want to project onto the first clip. In this case, as mentioned above, I am using a clip from *Tears of Steel*, a free film created by the Blender Foundation.

Connect the clip's *Read* node to the *CornerPin* node and then create a *Merge* node (*Merge* | *Merge*). From the *Merge* node's property box, you can choose a wide variety of merges from the *Operation* drop-down list. The default *over* operation is good enough for now, since it will allow you to see both clips, with the projected clip overlaid and semitransparent on the background clip. To pull the merged clips onto the viewer, connect the *CornerPin* node to *Merge*'s *A* connector and your background clip to *Merge*'s *B* connector. Then connect the *Merge* node to the *Viewer* (Figure 5).

You will see the projected clip overlaid on your background clip. The inclination and angle of the overlaid clip look good, but it is not placed between the nodes (Figure 6), which is where it should be.

What is happening is that Natron is projecting your clip onto a plane that has the same dimensions as the background clip. Since in this example the projected clip is smaller (1280x534) than the background clip (1920x1080), it appears in the corner of this invisible bigger plane, in its lower left corner, where the coordinates' origin is. What you want to tell Natron is to adjust the size of the plane to the size of the projected clip – that is, instead of using a 1920x1080 plane, Natron must use a 1280x534-sized plane.

You do this by adjusting the *CornerPin* node's own *From* parameters. Double-click the *CornerPin* node to bring its property box to the top of the stack, and locate the *to* and *from* tabs for the node. Click on the *from* tab; at the bottom of the tab, you will see three buttons: *Set to input rod* ("rod" stands for "region of definition"), *Copy "to"*, and *Copy "to" (Single)*. You want to click on *Set to input rod*, because the "rod" is what Natron calls the size of the source clip.

Once you do that, the projected clip will snap between the trackers (Figure 7), and your task is finished.

Complex Tracking

As the final node graph in Figure 5 shows, tracking and projecting is not all that complex in Natron – it can be sorted in six nodes, including the *Viewer*. Admittedly, there are quite a few parameters to fiddle with to get it right, and creating the trackers can be a pain, especially with low-quality footage. All that said, it is not crazy hard, just a little tedious.

Then again, the example I have chosen is pretty straightforward: The markers the trackers follow are on screen all the time, and there is nothing in the way covering the projected clip at any moment. If you want to go the extra mile and learn how to have objects occlude your projected video for extra realism, look into rotoscoping, which is the technique where you "cut out" an object and then follow it in a similar way to trackers, but using a silhouette instead.

Be warned that rotoscoping is very laborious, but it can produce some really impressive results.

Conclusion

Natron is an amazing piece of software, that could help transform GNU/Linux from a system for amateur film-makers into a viable platform for professional video processing.

However Natron is in danger. Nobody has been in charge of developing Natron since August, and the funding from Inria has dried up. This could mean the end of Natron; a demise made ever more tragic by the fact there are no other free and open source software applications like Natron. The closed proprietary alternatives can cost thousands of dollars and tie you into opaque formats, proprietary online services, and predatory agreements.

If you can help Natron or know someone who can, please take action before it is too late. ■■■

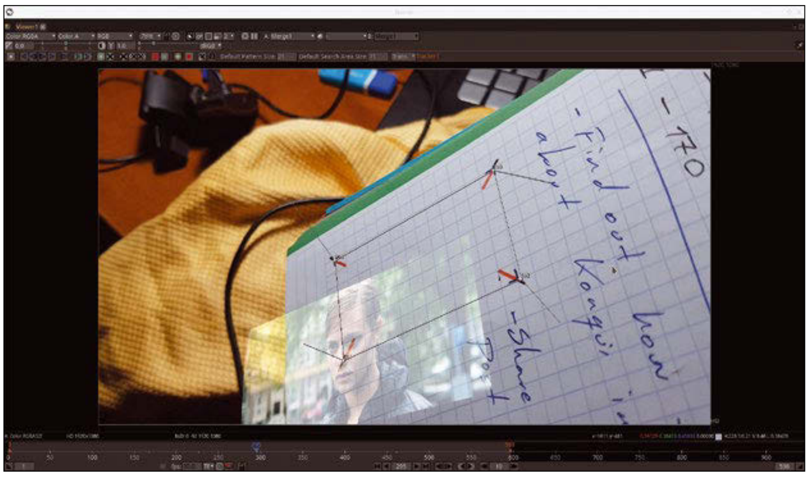


Figure 6: The overlaid clip needs relocating.

Info

[1] "Tutorials – Natron" by Paul Brown, *Linux Magazine*, issue 220, March 2019, pp. 90-94: *http://www.linux-magazine.com/Issues/2019/220/Tutorials-Natron*

[2] Natron: *https://natrongithub.github.io/*

[3] *Tears of Steel*: *https://mango.blender.org/*

[4] Notebook clip: *https://cloud.quickfix.es/index.php/s/AbWNW6Rpamy77n9*

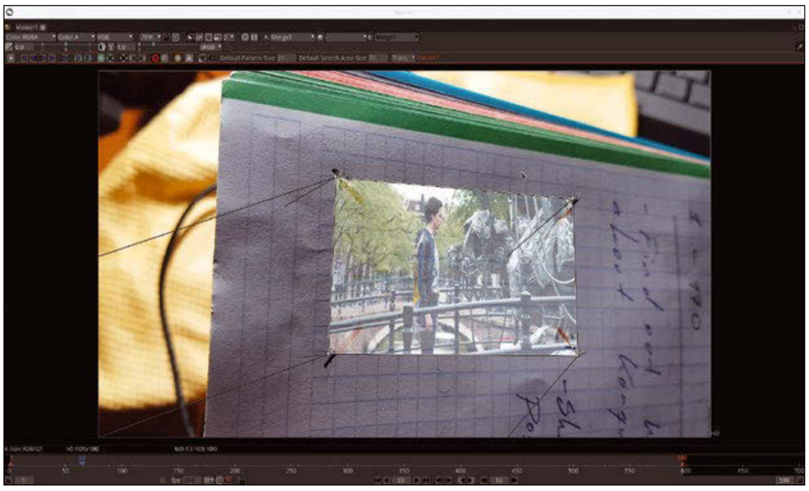


Figure 7: The final clip, with the projection in place between the markers.

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*.

DrupalCon

Date: April 8-12, 2019

Location: PSeattle, Washington

Website: *https://events.drupal.org/*

The Drupal community is one of the largest open source communities in the world. At DrupalCon, you'll learn to make, think about, and do things differently with Drupal.Each technical track includes sessions for beginners, experts, and everyone between. You'll leave DrupalCon inspired and empowered to create amazing web experiences.

Open Source Data Conference (OSDC)

Date: May 14-15, 2019

Location: Berlin, Germany

Website: *https://osdc.de/*

This international conference is especially adapted to experienced administrators and architects. Get in touch with international OS-experts. Benefit from their comprehensive experience, learn about the current developments, and gain the latest know-how for your daily practice.

JAX DevOps Conference 2019

Date: May 14-17, 2019

Location: London, United Kingdom

Website: *https://devops.jaxlondon.com/*

JAX DevOps is a four-day conference for software experts featuring in-depth knowledge of the latest technologies and methodologies for lean businesses. Join the software delivery revolution for accelerated delivery cycles, faster changes in functionality and increased quality in delivery.

Events

Icinga Camp Berlin	March 14, 2019	Berlin, Germany	https://www.icinga.com/events/icinga-camp-berlin/
Chemnitzer Linux-Tage	March 16-17, 2019	Chemnitz, Germany	https://chemnitzer.linux-tage.de/2019/en/
CloudFest 2019	March 23-29, 2019	Europa-Park, Germany	https://www.cloudfest.com/
SREcon 19 Americas	March 25-27, 2019	Brooklyn, New York	https://www.usenix.org/conference/srecon19americas
SUSEcon 2019	April 1-5, 2019	Nashville, Tennessee	https://www.susecon.com/
Cloud Foundry Summit	April 2-4, 2019	Philadelphia, Pennsylvania	https://www.cloudfoundry.org/event/nasummit2019/#
Open Networking Summit (ONS)	April 3-5, 2019	San Jose, California	https://events.linuxfoundation.org/events/open-networking-summit-north-america-2019/
DrupalCon	April 8-12, 2019	Seattle, Washington	https://events.drupal.org/
ASPLOS 2019	April 13-17, 2019	Providence, Rhode Island	https://asplos-conference.org/
LinuxFest Northwest	April 26-28, 2019	Bellingham, Washington	https://lfnw.org/conferences/2019
Linux Storage, Filesystem and Memory Management Summit	April 30-May 2, 2019	San Juan, Puerto Rico	https://events.linuxfoundation.org/events/linux-storage-filesystem-mm-summit-2019/
Red Hat Summit 2019	May 7-9, 2019	Boston, Massachusetts	https://www.redhat.com/en/summit/2019
Open Source Data Center Conference (OSDC)	May 14-15, 2019	Berlin, Germany	https://osdc.de/
JAX DevOps Conference 2019	May14-17, 2019	London, United Kingdom	https://devops.jaxlondon.com/
Secure Linux Administration Conference (SLAC)	May 27-29, 2019	Berlin, Germany	https://www.heinlein-support.de/secure-linux-administration-conference

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

Bernhard Bablok	60
Swapnil Bhartiya	8
Jens-Christoph Brendel	16
Paul Brown	92
Zack Brown	12
Bruce Byfield	28, 56
Joe Casad	3
Mark Crutch	67
Markus Feilner	40
Marco Fioretti	86
Karsten Günther	69
Jon "maddog" Hall	68
Charly Kühnast	38
Christoph Langner	24, 32, 74
Vincent Mealing	67
Paul Menzel	20
Pete Metcalfe	50
Graham Morrison	80
Mike Schilli	46
Ralf Spenneberg	74

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
Lori White
Cover Image
© cobalt, 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 222 / May 2019

Snapshot Tools

Approximate

UK / Europe	Mar 30
USA / Canada	Apr 26
Australia	May 27

On Sale Date

Snapshot tools record the precise state of a system or disk at a moment in time. Next month we review some tools for creating system snapshots and disk images, including Clonezilla, CYA, Partimage, qt-fsarchiver, and Snapper.

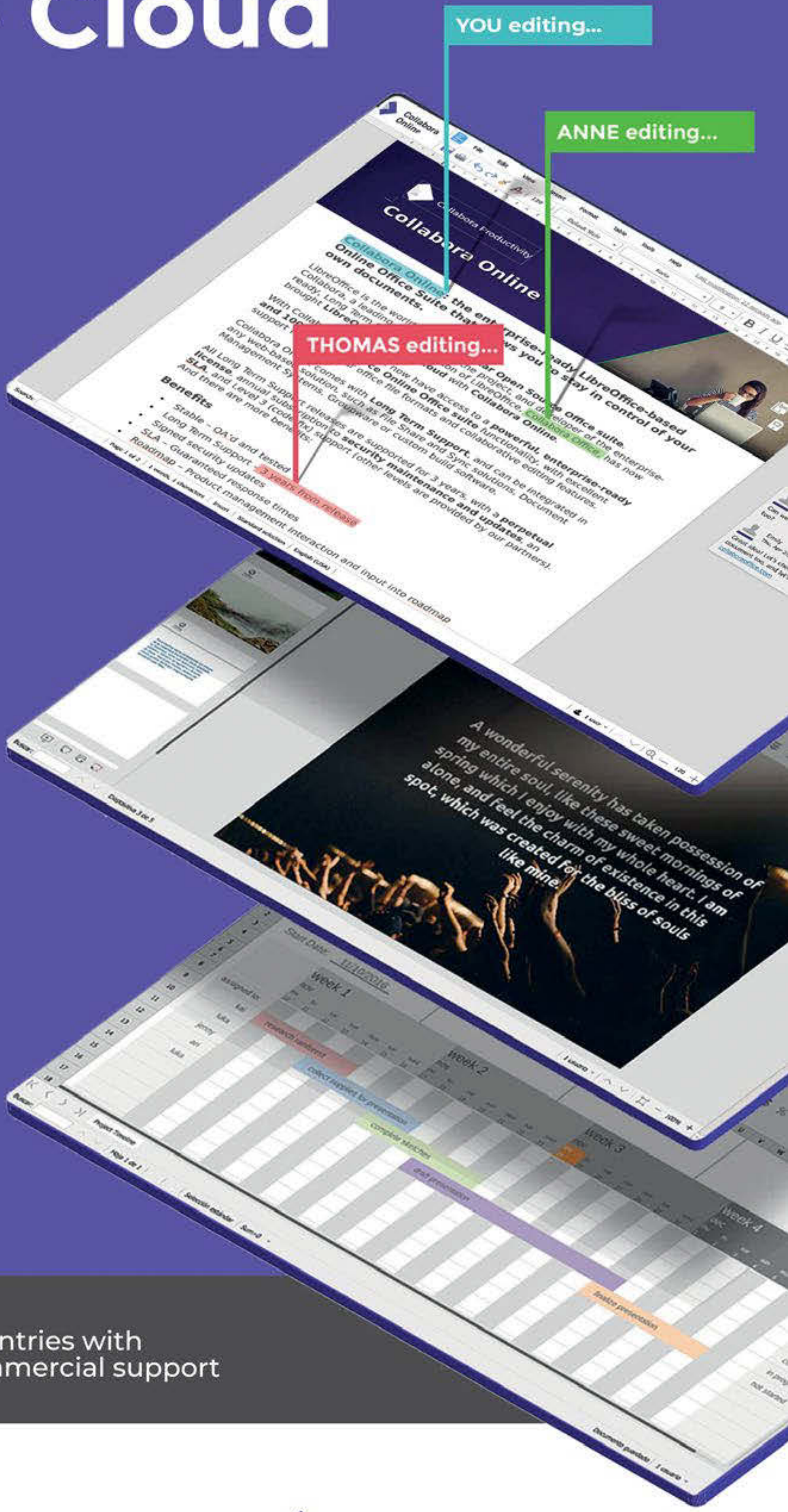
Lead Image © tomertu, 123RF.com

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: ***www.linux-magazine.com/newsletter***

The powerful on-premise office suite in the Cloud



P

T

S

Protects users privacy and allows them to keep full control of sensitive data

- Powerful interoperability
- Live collaborative editing
- Service Level Agreement
- On-premise privacy protection
- Collaborative roadmap planning
- Host your own on-line Office Suite

130 Partners worldwide **55** Countries with commercial support

Try it out at
collaboraoffice.com

Collabora Online

SUPERMICRO

Save Money.
Save Mother Earth.

RESOURCE-SAVING SERVERS

Build your data center with technology that is
BIG on **SAVINGS** and **LOW** on environmental **IMPACT**
with Intel® Xeon® Scalable Processors

60% Space Savings | 50% Energy Savings | e-Waste Savings

intel XEON PLATINUM inside

Learn more at
www.supermicro.com/WeKeepITGreen

SUPERMICRO