FREE DVD

IoT TRICKS
Indoor climate control with MQTT

SPOTLIGHT ON SMALL DISTROS

LINUX PRO MAGAZINE
SEPTEMBER 2019

SPOTLIGHT ON SMALL DISTROS

Choose the best small-footprint Linux for your environment

Nftables
Exploring the new Linux firewall tool

Secure your LibreOffice PDFs

Robotics
Controlling a robot arm with PiXtend

wikit
Wikipedia from the command line

Network Sleuth
Searching for problems with ARP

Timeshift
Create and manage system snapshots

LINUXVOICE

- Scribus: Free desktop publishing
- Puppy Linux BionicPup
- maddog on FreedomBox

FOSSPicks
- Pick colors with Pick
- Name Generator
- Heaptrack

Tutorials
- WordPress
- 3D Printing

Issue 226
Sep 2019
US$ 15.99
CAN$ 18.50

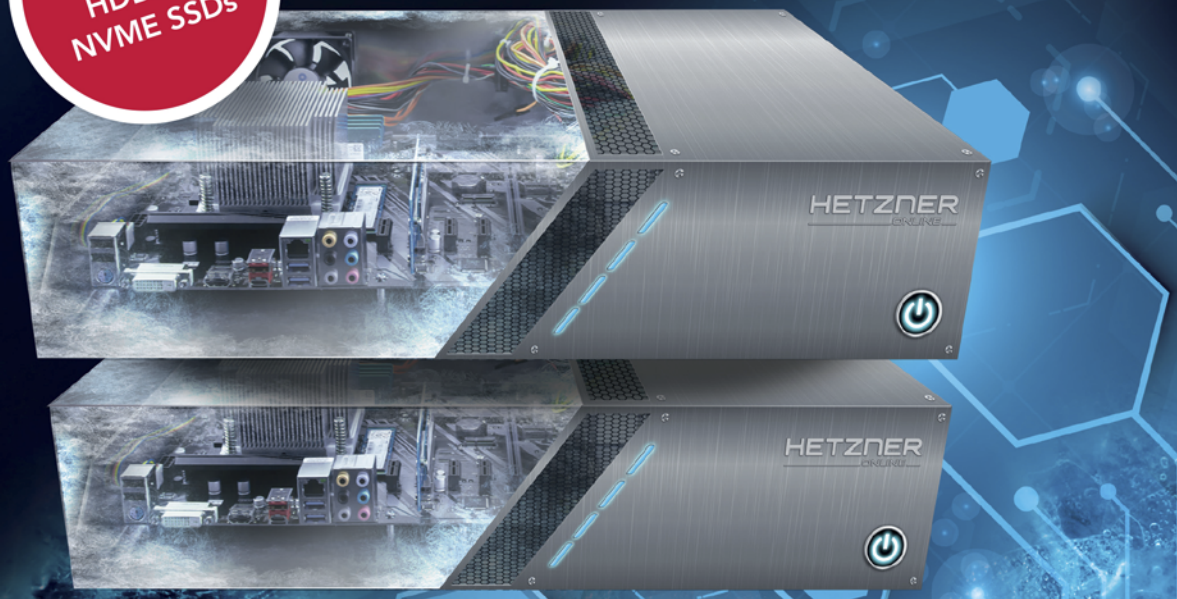
HETZNER ONLINE

AN EXTREME REFRESH WITH OCTA-CORE POWER!

DEDICATED ROOT SERVER EX62

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

ENTERPRISE HDDs or NVME SSDs



Dedicated Root Server EX62

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

monthly from $ **72.50**

Dedicated Root Server EX62-NVMe

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

monthly from $ **72.50**

www.hetzner.com

THE BIG PICTURE

Dear Reader,

IBM finally closed the deal to purchase Red Hat this month, writing a $34 billion check for the leading Linux company in hopes of restarting its cloud and network service endeavors. Red Hat was a really big fish in the open source scene, but they are tiny compared to IBM, and many observers are wondering how this story will unfold.

In the press release confirming the sale [1], IBM said all the right things, leading off the announcement with the following bullets:

- IBM preserves Red Hat's independence and neutrality; Red Hat will strengthen its existing partnerships to give customers freedom, choice, and flexibility.
- Red Hat's unwavering commitment to open source remains unchanged.
- Together, IBM and Red Hat will deliver a next-generation hybrid multicloud platform.

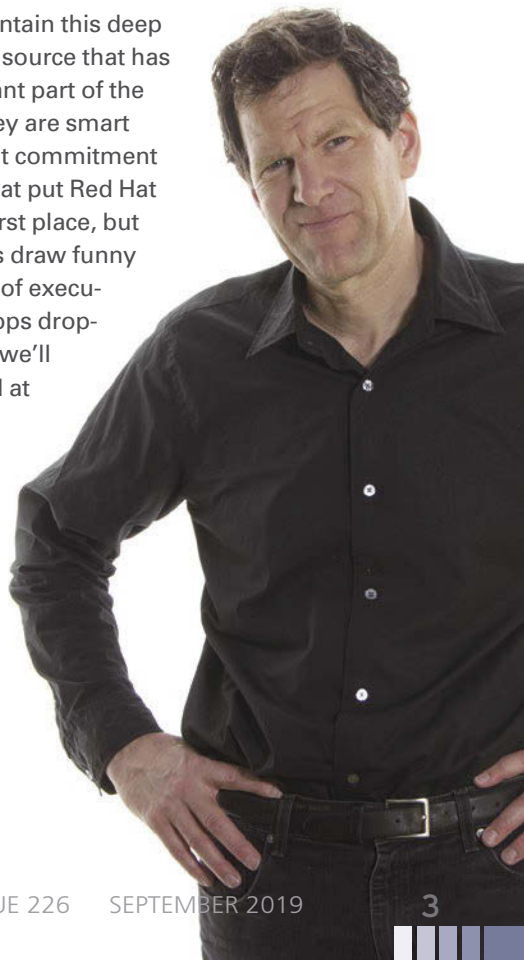
As you probably already know, this deal is all about the cloud, and about integrating hybrid cloud with IBM's existing business service portfolio. The quote from IBM CEO Ginni Rometty sounds like a manifesto, "Businesses are starting the next chapter of their digital reinventions, modernizing infrastructure and moving mission-critical workloads across private clouds and multiple clouds from multiple vendors. They need open, flexible technology to manage these hybrid multicloud environments. And they need partners they can trust to manage and secure these systems. IBM and Red Hat are uniquely suited to meet these needs. As the leading hybrid cloud provider, we will help clients forge the technology foundations of their business for decades to come."

It all looks good for the big picture, and IBM has always been good at the big picture. The company has a long history with the cloud. In fact, one could argue that IBM's Big Green initiative back in 2007 was one of the first, bold steps that got the cloud revolution off the ground [2]. It is an interesting lesson for all who will heed it that, despite their early interest and energy in cloud computing, IBM got left behind by the agility of smaller companies and the power of open source development. Twelve years after Big Green, Big Blue is looking for a way back in, and Red Hat offers tantalizing possibilities.

How will all this end? It is too early to know. Whatever they tell you, things are never exactly the same when a big company buys a smaller company, but that doesn't mean it will be a disaster. IBM has been working on infusing some startup culture and sensibility into their massive ranks for some time, so they are at least thinking about how to keep it fresh. And actually, IBM is pretty good at maintaining a long-term vision. Better IBM than Oracle or Microsoft.

I'm not too worried about what will happen with Red Hat Enterprise Linux; however, I do wonder what will become of all the little projects and infrastructure services that Red Hat helps to develop. The Red Hat website lists over 400 open source projects to which Red Hat contributes [3]. Some of the tools on the list are core Linux components you hear about every day. Others are lesser known tools that you never hear about, because they *just work* and don't cause a lot of bug reports. Of course, Red Hat is not the lead sponsor on all 400 of these projects, but they *are* a big part of many. Sage Weil, the creator of the Ceph storage system, works for Red Hat. So does Lennart Poettering, creator of systemd. Other FOSS projects, like Spacewalk, JBoss, and many more, carry the life blood of Red Hat.

In the short term, I'm sure nothing will change, but if the economy goes downhill, and the company starts looking for ways to cut costs and conserve resources, will IBM maintain this deep commitment to open source that has been such an important part of the Red Hat culture? If they are smart they will, because that commitment to free software is what put Red Hat ahead of IBM in the first place, but cost cutting directives draw funny pictures in the minds of executives. If the money stops dropping in their pockets, we'll see if IBM is still good at the big picture.



Joe

Joe Casad,
Editor in Chief

Info

[1] IBM Closes Landmark Acquisition of Red Hat: *https://newsroom.ibm.com/2019-07-09-IBM-Closes-Landmark-Acquisition-of-Red-Hat-for-34-Billion-Defines-Open-Hybrid-Cloud-Future*

[2] Big Green Initiative: *https://www.pcworld.com/article/131777/article.html*

[3] Red Hat Software Contribution: *https://community.redhat.com/software/*

LINUX MAGAZINE

WHAT'S INSIDE

Hardware resources keep expanding, and mainstream OS systems also keep expanding, sucking up the newfound space with more and better bloatware. But many users would rather escape the arms race and keep it simple. This month, we review some of the top resource-conscious small distros. Also inside:

- **Nftables** – this new tool for managing Linux firewalls is gradually replacing iptables (page 20).
- **Timeshift** – a user-friendly app for creating and managing system snapshots (page 28).

Check out MakerSpace for a homegrown IoT humidity monitor, and see our LinuxVoice section for tutorials on WordPress and 3D

SERVICE

NEWS

COVER STORIES

IN-DEPTH

IN-DEPTH

MAKERSPACE

SEE PAGE 6 FOR DETAILS

LINUXVOICE

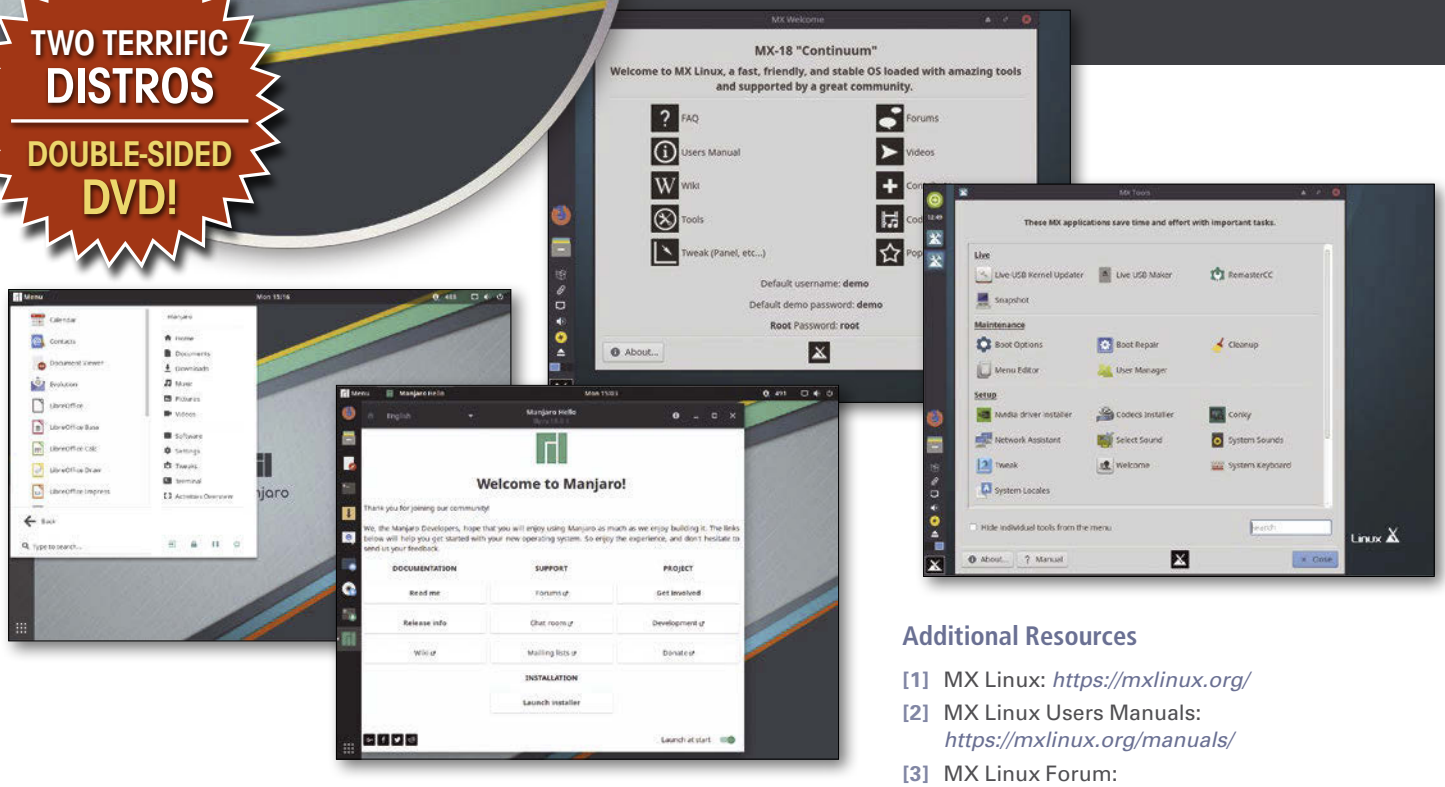
On the DVD

MX Linux 18

MX Linux calls itself a "midweight OS designed to combine an elegant and efficient desktop with simple configuration, high stability, solid performance, and medium-sized footprint." MX, which is a collaboration between the antiX and MEPIS communities, has gained attention throughout the Linux world and is now at the top of the DistroWatch list with over 4,000 hits per day. Inside you'll find a customized Xfce desktop and the innovative MX Tools – a collection of utilities for configuring and troubleshooting the boot process, sound system, video drivers, and more. MX is based on Debian and includes both SysVinit and systemd – with SysVinit as the default.

Manjaro Gnome 18.0

Manjaro is a desktop OS based on the iconic Arch Linux. Manjaro comes with classic Arch tools, such as the pacman package management system, but prides itself on a simple and user-friendly desktop environment.



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

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

Additional Resources

[1] MX Linux: *https://mxlinux.org/*
[2] MX Linux Users Manuals: *https://mxlinux.org/manuals/*
[3] MX Linux Forum: *https://forum.mxlinux.org/*
[4] Manjaro: *https://manjaro.org/*
[5] Manjaro User Guide: *https://manjaro.org/support/userguide/*
[6] Manjaro Forum: *https://forum.manjaro.org/*

openSUSE
Asia Summit 2019
Bali, Indonesia

Information Technology Department
Faculty of Engineering
Udayana University
Bali, Indonesia
October 5—6, 2019

- openSUSE
- Embedded System
- Containers
- Open Source Software
- Cloud
- ...and more

https://events.opensuse.org/conferences/summitasia19

NEWS

Updates on technologies, trends, and tools

THIS MONTH'S NEWS

Debian Buster Arrives

The Debian community has announced the release of Debian 10 "Buster" (*https://www.debian.org/News/2019/20190706*). Debian is one of the most popular GNU/Linux-based distributions. Buster will be supported for the next five years.

Buster ships with several desktop environments including Cinnamon 3.8, GNOME 3.30, KDE Plasma 5.14, LXDE 0.99.2, LXQt 0.14, MATE 1.20, and Xfce 4.12. In this release, GNOME will default to using the Wayland display server instead of Xorg. "The Xorg display server is still installed by default and the default display manager allows users to choose Xorg as the display server for their next session," according to a blog post from the Debian project.

The Reproducible Builds project enabled Debian developers to build bit-for-bit identical binary packages of the open-source packages available in Debian 10. "This is an important verification feature, which protects users against malicious attempts to tamper with compilers and build networks. Future Debian releases will include tools and metadata so that end-users can validate the provenance of packages within the archive," said the blog post.

To make Debian more secure, AppArmor is installed and enabled by default. Furthermore, all methods provided by the APT package manager (except cdrom, gpgv, and rsh) can optionally make use of seccomp-BPF sandboxing. The HTTPS method for APT is included in the apt package and does not need to be installed separately.

Buster supports a total of ten architectures, including 64-bit PC / Intel EM64T / x86-64 (amd64), 32-bit PC / Intel IA-32 (i386), 64-bit little-endian Motorola/IBM PowerPC (ppc64el), 64-bit IBM S/390 (s390x), ARMel, and more.

Debian 10 is available for free download (*https://www.debian.org/distrib/*).

IBM Acquires Red Hat

IBM has completed the acquisition of open source company Red Hat for approximately $34 billion (*https://www.tfir.io/2019/07/09/ibm-acquires-red-hat-for-34-billion/*). Founded in 1993 as a Linux vendor, Red Hat has evolved into an open-source infrastructure player. Red Hat has become synonymous with a successful open source business model that's often called the Red Hat Business Model.

Red Hat has become one of the most successful open source companies, clocking in at around $3.4 billion in annual revenues. The company will become an independent entity within IBM as part of IBM's Cloud and Cognitive Software segment.

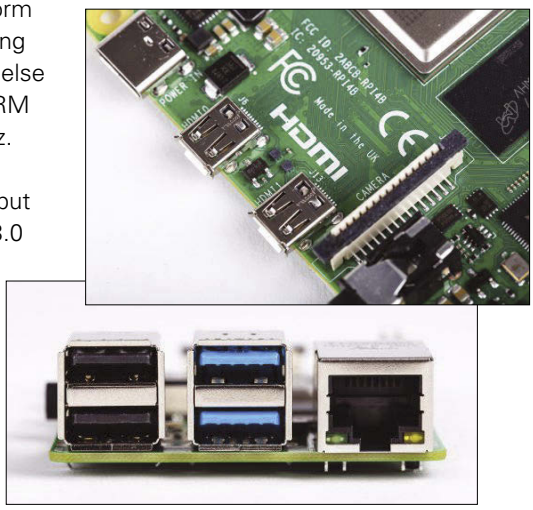
Red Hat will continue to be led by Jim Whitehurst and its current management team. Whitehurst will report to IBM CEO, Ginni Rometty.

IBM will maintain Red Hat's headquarters in Raleigh, North Carolina, its facilities, brands, and practices.

Raspberry Pi 4 Is Here

Eben Upton, the founder of the Raspberry Pi Foundation, announced the next major update to the single-board computer. Raspberry Pi 4 is here (*https://www.raspberrypi.org/blog/raspberry-pi-4-on-sale-now-from-35/*).

The new model has the same form factor and pricing. Yes! It's still being sold for $35. But it has everything else improved. Pi 4 features a 64-bit ARM Cortex-A72 CPU running at 1.5Ghz. Now you can get up to 4GB of LPDDR4 SDRAM and full-throughput Gigabit Ethernet. It has two USB 3.0 and two USB 2.0 ports. You can now connect two monitors running at 4K resolution. It also has 4Kp60 hardware decode of HEVC video. Instead of using the old micro-USB for power, it's now using USB-C.



Raspberry Pi foundation has also released a new Raspbian operating system for this device. "To support Raspberry Pi 4, we are shipping a radically overhauled operating system, based on the forthcoming Debian 10 Buster release. This brings numerous behind-the-scenes technical improvements, along with an extensively modernized user interface, and updated applications including the Chromium 74 web browser."

Ubuntu Takes A U-Turn with 32-Bit Support

Canonical, maker of the popular Ubuntu Linux distribution, has revived support for 32-bit libraries after feedback from WINE, Ubuntu Studio, and Steam communities.

Last week Canonical announced that its engineering teams decided that Ubuntu should not continue to carry i386 forward as an architecture (*https://ubuntu.com/blog/statement-on-32-bit-i386-packages-for-ubuntu-19-10-and-20-04-lts*). "Consequently, i386 will not be included as an architecture for the 19.10 release, and we will shortly begin the process of disabling it for the eoan series across Ubuntu infrastructure," wrote Will Cooke, Director of Ubuntu Desktop at Canonical.

Photo by Justin Luebke on Unsplash

However, the news was not received well. Canonical was criticized for the move. Responding to the uproar, Canonical decided to continue to support 32-bit applications.

As Steve Langasek, a Debian and Ubuntu developer wrote in a mailing list, maintaining support for 32-bit libraries is "a cost largely paid by Canonical (both in terms of infrastructure and in terms of engineering work to keep the base system working). It's not very compelling to say that Canonical should continue bearing these costs out of pocket on the grounds that some other companies are unwilling to update their software to an ISA from this millennium :)"

OpenSSH Fixes Side Channel Attacks

There is a rise in memory side-channel vulnerabilities like RAMBleed, Spectre, and Meltdown. OpenSSH is often at the center of attacks where a bad actor "exploits

MORE ONLINE

Linux Magazine
www.linux-magazine.com

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

Fail2ban: A Password Protection Service
Ken Hess
Fail2ban is a quick to deploy, easy to set up, and free to use intrusion prevention service that protects your systems from brute force and dictionary attacks.

Cross-Platform Database Management with DBeaver • Mayank Sharma
Save time and effort as you rummage through large quantities of data.

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

OpenMP – Coding Habits and GPUs
Jeff Layton
In this third and last article on OpenMP, we look at good OpenMP coding habits and present a short introduction to employing OpenMP with GPUs.

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

Jenkins Configuration as Code
Tomasz Szandala
The move from Groovy scripts to Jenkins Configuration as Code simplifies the initialization of Jenkins and Jenkins plugins.

Translation Tool with Version Control
Ulrich Bantle
The free web-based Weblate provides coordinated web-based localization with links to version control systems that propagate translations across components within projects.

Prepare, Analyze, and Visualize Data with R
Mira Céline Klein
The free R programming language and software environment for statistical computing and graphics is well supported, has great flexibility, and is easily automated.

memory read vulnerabilities to steal secret SSH private keys from the restricted memory regions of the system," according to The Hacker News.

The root cause of this issue is the fact that the OpenSSH agent stores a copy of the SSH keys in the memory (RAM of CPU), eliminating the need of entering a passphrase to log into the server via SSH. Since these keys are stored in either RAM or CPU in plaintext, they are susceptible to attacks.

The OpenSSH community is now fixing this issue through an update. OpenSSH will now encrypt private keys before storing them into the system memory.

"Attackers must recover the entire prekey with high accuracy before they can attempt to decrypt the shielded private key, but the current generation of attacks have bit error rates that, when applied cumulatively to the entire prekey, make this unlikely," said Damien Miller of the OpenBSD project on a mailing list (*https://marc.info/?l=thn&m=156109087822676*).

Firefox Fixes Error that Crashed HTTPS Pages

Mozilla has made changes to its Firefox browser that helps sysadmins fix TLS errors due to HTTPS (*https://blog.mozilla.org/security/2019/07/01/fixing-antivirus-errors/*). These errors are triggered by antivirus software that try to intercept secure connections over HTTPS.

The cause of the problem was the fact that Firefox trusts only those Certificate Authorities (CAs) that are listed in its own store, whereas antivirus software systems use their own CAs. "The antivirus products relying on other trusted CAs provided by the operating system (OS) are not allowed to intercept HTTPS connections on Firefox," said The Hacker News.

This conflict between Firefox and antivirus software led to users experiencing crashed HTTPS pages showing errors like "SEC_ERROR_UNKNOWN_ISSUER."

To fix the issue, Mozilla has created a mechanism to detect when a Firefox error is caused by a MITM. Users can enable the 'enterprise roots' preference that allows Firefox to import any root CAs that have been added to the OS by the user, an administrator, or a program that has been installed on the computer.

This option is available only on Windows and MacOS.

"It might cause some concern for Firefox to automatically trust CAs that haven't been audited and gone through the rigorous Mozilla process. However, any user or program that has the ability to add a CA to the OS almost certainly also has the ability to add that same CA directly to the Firefox root store," said Mozilla in a blog post.

Altair Releases Hyperworks 2019

Altair has announced the release of the Altair Hyperworks 2019 high-performance computing development platform (*https://www.altair.com/news/11632/*). The new version of the company's flagship modeling and development platform provides faster simulation and more efficient workflows.

According to Altair, the new release "...expands on the number of solutions available for designers and engineers, under a single, open-architecture platform, to speed decision-making and time to market."

Hyperworks 2019 features faster simulation for complex assemblies, better fatigue-life prediction, more efficient multiphysics workflows, superior modeling, and expanded non-linear solver functionality.

See the Altair website for more information on Altair Hyperworks 2019 (*https://www.altair.com/hyperworks-2019/*).

Zack's Kernel News

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

By Zack Brown

Adding git Documentation

Documentation is lovely. One especially lovely form of documentation identifies the reasons for using a particular tool in a certain way. Jonathan Corbet recently documented how maintainers should use `git` to do merges and rebases within the context of feeding patches from many developers up to higher-level maintainers and ultimately up to Linus Torvalds for inclusion in the official kernel tree.

It's no wonder that the situation is tricky. When Linus wrote the `git` tool, he created a situation in which one group of developers could "pull" an entire tree of development and treat their version essentially as the top of a whole other project. Then other groups of developers could pull that tree into their own sub-projects, and so on, like a spider plant budding little spider plants off of it. This hadn't really existed before, and it's a kind of weird topology, when you consider that earlier revision control systems had none of that. Developers simply contributed to one central repository, with no branching or merging of which to speak – Just one spider plant, but no others budding off it.

As owner of their own sub-projects, maintainers coordinate work among developers coding on that sub-project. To keep things clean, they might do things like edit individual commits after they've already been accepted into the sub-tree or reorder a bunch of commits to have a more logical-seeming history. This is all normal, especially in a large and complex project where the history of the work is crucial for identifying copyright ownership and finding the particular patches that introduced bugs. This editing and reordering is called "rebasing."

Likewise, sub-project maintainers may feed their `git` tree to an upstream maintainer, hoping that maintainer will "merge" all the patches from that tree into their own, and that they'll then submit the upstream tree to be merged yet further upstream and eventually into Linus's tree, where the patches will become part of the official Linus kernel.

However, merges can go in both directions. The maintainer of a sub-project may want to merge everything that's been going on in an upstream tree into their sub-project's tree, so they'll stay up-to-date with all the work going on in the wider world.

Jonathan's documentation talks about exactly when it's good and bad to merge trees either upstream or downstream, and when it's good to do a rebase – or at least when it should definitely be avoided.

One rule of thumb he identified is very simple, yet also perhaps annoying:

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**.

Don't rebase any `git` repository that has already been made public. In other words, only rebase your own private work. Otherwise, you set up all sorts of potential conflicts between trees that have already been merged, but then one tree is rebased and gets merged with the other again; you end up with a big mess instead of the lovely clean history you desired.

In fact, Jonathan said, you really should never rebase at all, if you can help it. Rebasing changes the context in which each patch entered the codebase and could actually introduce bugs. So you should weigh your desire to rebase against your willingness to test all your rebased patches again, even if you had already tested them thoroughly before rebasing.

Merging is a different thing entirely. There's no way to avoid merging – it's one of the main points of using `git` in the first place. Fork and merge; fork and merge. That's the development process in a nutshell, not just for Linux, but for any project that uses `git` and has more than a single developer.

As Jonathan said, there were thousands of merge requests just during the Linux 5.1 development cycle. Almost nine percent of all kernel commits were merges.

He also said merges were good for another reason. Linux development history would be clearer if related patches were grouped together in a merge, as opposed to feeding each individual patch into the tree separately. He admonished developers not to bother trying to rebase their trees to avoid merges. A merge was the good case, he said.

This wasn't only true for the top-level tree maintained by Linus. Downstream trees should also accept merges from developers of yet-further-downstream trees, Jonathan said. They should simply add a commit message describing the purpose of the merge and the patches that went into it, just as if it were the commit message for a single patch.

Likewise, he said, for bug hunting, the "Signed-off-by" tags generally used in kernel development should be used in trees going all the way downstream. This would help keep track of who did what and help resolve claims of copyright violation or other legal matters that might arise. As he put it, "Failure to do so threatens the security of the development process as a whole."

Jonathan also cautioned against merging from an upstream tree into a downstream tree that already had unmerged work in it. This could be tempting, if for example you've been working on a new driver for awhile, and your tree has become out of date with all the patches pouring into Linus's official tree; you want to merge all of those changes into your tree, to make sure none of the work you're doing will conflict with anything anyone else has been doing.

It's a perfectly legitimate desire, but Jonathan recommended against it. It would essentially bring a huge X factor of all those many unknown patches into your nice isolated little development branch, which could have a big impact on the behavior of your development kernel; you might have no way of knowing which changes were introduced by problems in your own code versus problems in someone else's.

You might think it would be okay to do a merge from an upstream tree right at the tail end of your development cycle, just to check for conflicts with anyone else's work and resolve those conflicts before submitting your own merge request.

But Jonathan said that wouldn't be good either. As he put it, Linus wanted to see those conflicts himself, so he could go over those parts of the code more carefully before accepting the patches. Conflicts, Jonathan said, identified problem areas that should receive more attention, and not just from the people submitting the patches.

However, Jonathan said it was perfectly fine for you to do a "test merge" (i.e., copy your own branch, merge the upstream tree into that branch, and then check for conflicts yourself). But instead of trying to resolve those conflicts, you should simply discuss their existence when sending your merge request to Linus. That way, he'll know that you know there are issues, and he won't think they somehow slipped by you while he works on resolving them himself.

He closed the doc by saying that, really, these were general guidelines and not laws. Sometimes violating them would be the right thing to do.

There was general approval, especially from Linus, and a few people had some constructive criticism to offer. But the doc is sailing into the kernel tree. It's excellent, because this kind of documentation represents opinion rather than cold hard fact. It documents the preferences and policies of the project leaders and the reasons behind them. Other projects could have different preferences and different reasons.

Untangling the System Call Situation

It's sort of a pain in the neck to change the kernel's system calls. Each supported architecture needs to be updated, and then the lookup tables need to be updated – so the number of each system call can be paired with the actual kernel function being called – and then of course all the other system calls may need to be renumbered. Since there are thousands of people all pouring patches into the kernel at lightning speed, it's likely someone else has also renumbered the system calls, which will cause conflicts between your two patches.

David Howells had the perfectly reasonable impulse to write a script that would handle all the nightmare elements of this process for you. It would update the tables and renumber the other system calls. It would even resolve patch conflicts with other developers. It had all the fixins.

Arnd Bergmann gave his mixed reaction very clearly, when he said that the script looked great – except for the fact that it was needed at all.

Linus Torvalds went the other way, saying the script would have been great – except that what was really needed was something completely different. He said:

"I really think the solution is not this kind of helper script, but simply that we should work at not having each architecture add new system calls individually in the first place.

IOW, we should look at having just one unified table for new system call numbers, and aim for the per-architecture ones to be for 'legacy numbering'.

Maybe that won't happen, but in the hope *that it happens, I really would pre-*

fer that people not work at making scripts for the current nasty situation."

And that was that. Linus generally makes no bones about the fact that he'll happily inconvenience developers, even while going to tremendous lengths not to inconvenience users. Binary interfaces that might be in use by a single piece of existing compiled code, for example, are sacrosanct. Interfaces that developers use all the time, however, might just go out the window if there's a decent reason. And in the current case, interfaces that are a vicious trial to update will remain a vicious trial, until someone comes up with a better underlying design.

Bit or Bitmap?

One way to get clarification on a feature requirement is to do it wrong and wait for someone to squawk. Thierry Reding may or may not have done that very thing recently, when he posted a patch to update the core driver code so it wouldn't give a warning that he felt might confuse users.

The warning came about when the kernel failed to probe a given driver before reaching the end of its init process during boot time. The `probe()` function was supposed to initialize all devices and get information about the resources it offered to the system. Without that initialization and information, the system wouldn't be able to use the device. Ordinarily, it would be appropriate for the kernel to give a warning in that circumstance.

However, in some cases, a device couldn't be initialized, because it was a hot-plug device or for various other legitimate reasons. And in that case, a driver that depended on that device might need to defer its `probe()` call until the device came online – which could be at any point after bootup.

In that case, Thierry said, it would be better to let the driver tell the kernel that was deferring its `probe()` call until further notice that no warning would be needed – this would be business-as-usual as far as the user was concerned.

Thierry's patch added a flag to some kernel functions, to allow the calling routine to specify that `probe()` was being deferred. This flag was in the form of a bitmap (i.e., a string of 1s and `0`s). And this, it turned out, was the cue for a squawk.

Rob Herring and Rafael J. Wysocki had no real objection to the patch, and all seemed well.

Greg Kroah-Hartman, on the other hand, had already spoken to Thierry about this particular feature and had given what he felt had been clear instructions about how the patch should be implemented. Instructions that, in his opinion, Thierry had simply ignored.

Greg had told Thierry to avoid "odd flags," after Thierry's first version of the patch had used a Boolean instead of a bitmap. The bitmap, Greg now said, "did not make the API any easier to understand at all." And he added, "do it correctly please, like I asked for the first time."

Thierry pointed out that Greg had really only specified "no Boolean flags" and had not really been clear about what would be better. He suggested, "to avoid further back and forth, what exactly is it that you would have me do? That is, what do you consider to be the correct way to do this?"

As an alternative to his flagging solution, he proposed some new code that relied on return values and clearly named function calls to indicate that `probe()` was being deferred. And after some modifications suggested by Rafael, Greg was mollified. Mostly he wanted the interface to be readable and clear and not to require anyone to "just know" the meaning of some binary (or bitmapped) flag. He said, "Yes, that's much more sane. Self-describing APIs are the key here; I did not want a Boolean flag, or any other flag, as part of the public API as they do not describe what the call does at all."

The culture of kernel development is full of personalities, with many different approaches to how to get things done. Sometimes knowing how to get the full attention of the upstream maintainer can be the quickest way to get your patch into the kernel. ■■■

We round up some top small Linux distributions

Small Is Beautiful

Beyond the realm of mainstream, enterprise-ready desktop giants lie a handful of miniscule Linux distributions that barely leave a noticeable footprint on a computer's resources. Here's a look at some of the leading small distros. *By Mayank Sharma*

There is no universal definition of the word "small" when it is used to define a Linux distribution. For some users, a small distribution is one that has a minimal footprint on a computer's resources. For others, a small distribution is the one that is distributed in a nifty little package and can be lugged around in a portable USB key. Both types of distributions have their uses. Small-footprint distros help save old and resource-strapped computers from ending up in a landfill, while USB-style distros give their users a safe, secure, and private environment on a public computer, like in a library or hotel lobby. The candidates we've selected in this article do a bit of both (see also the "Not Considered" box).

A lot of hardware is good enough to run the mainstream Linux distributions without any issues. However, the modern Linux desktop is a fairly resource hungry beast as well. Coupled with the fact that mainstream Linux distros are designed to appeal to a large number of users, they also have become too bloated. This has fueled the demand for lightweight options both for individual apps (see box titled "Lightweight Apps") and for complete distributions.

Just as it is difficult to define a distro as "small," defining hardware as "older" is also tricky. In addition to users who have hardware that's been outdated fairly recently, there's another kind who are holding on to their workhorses from the last decade. They usually just use their computer to browse the web, do some text editing, and watch some videos. It's a shame to force these users to dump their trusted machines and get the latest multicore computers loaded with several gigabytes of RAM or even a dedicated graphics card. However, chances are their workhorse isn't supported by the latest kernel, which keeps dropping support for older hardware that is no longer in vogue.

Carbon Dating Hardware

Forget about decades-old hardware; many popular distributions don't think it's viable to even support 32-bit architecture any more. Everything from new user projects such as Solus and elementary OS to the privacy-centric Tails Linux has dropped support for the 32-bit platform. Even mainstream projects like Arch and Ubuntu no longer offer installation ISOs for 32-bit machines. Despite the fact that 32-bit machines disappeared from the shelves quite a while back, they still adorn a lot of desks all over the world. Thankfully, you can still put these old machines to good use, and it really isn't a surprise that a large number of open source developers are working hard to make obsolete hardware usable again.

New software is always leveraging on the pace of hardware developments and rendering even relatively newer hardware obsolete. Examples of these relatively recent attic-ready hardware would be single-core or dual-core AMD Athlons and Intel Pentiums with about 2GB of RAM that they share with onboard graphics. A couple of years ago, mainstream distros would perform adequately on these machines, but not anymore. Then there are the hordes of netbooks that were pitched as light and portable replacements for laptops, but their under-powered hard-

Author

Mayank Sharma is a technology writer, and you can read his scribblings in various geeky magazines on both sides of the pond.

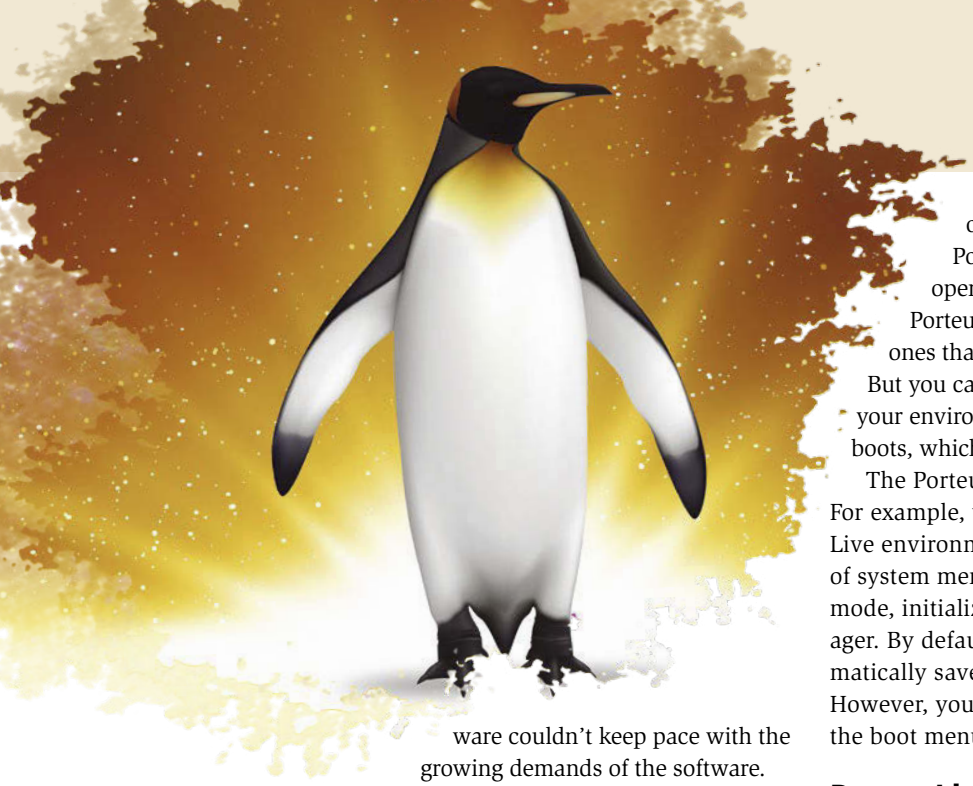
Not Considered

Several distributions use lightweight apps, but aren't necessarily small enough to qualify for this article. Prime examples include the official Ubuntu spins based on the LXDE and Mate desktops, which are designed for hardware that doesn't have the power to run the main edition but aren't small in size. The Trinity Desktop Environment based Q4OS, the JWM-powered Star Linux, antiX, and 4MLinux miss out for the same reasons.

Lightweight Apps

Two traits are common to every lightweight distro; a lightweight desktop environment and lightweight apps. Lightweight alternatives exist for every type of software. For instance, office apps like the AbiWord word processor and the Gnumeric spreadsheet are a popular replacement for mainstream suites like LibreOffice and Calligra. Other popular small contenders include the Midori web browser, Xpdf PDF viewer, Fotoxx image editor, CMPlayer media player, and App Grid software center, among others. Some distros, most notably Puppy Linux, also replace the stock apps with custom ones that are optimized for the minimal environment under which they are running.

Some desktop environments have a lighter footprint on resources as compared to mainstream desktop environments like Gnome 3 and KDE 4. LXDE and Xfce are two of the popular options that are used by many distros. There's also Mate, which is a continuation of Gnome 2. Then there's Enlightenment, which includes some bling as well. Other distros don't even use a full-fledged desktop environment and opt instead for a simpler window manager, such as Openbox. Other lighter but esoteric window managers include IceWM, Fluxbox, FLWM, and Joe's Window Manager (JWM).



ware couldn't keep pace with the growing demands of the software. Using the distributions in this article, you'll be able to put your trusty old workhorse back into active service and even ensure you have a miniscule portable environment that you can safely use on a public computer. Whatever your needs, you'll be able to walk away with a distribution that meets your requirements.

Porteus

Porteus [1] is a portable distribution designed and optimized to run from removable media such as a USB flash drive, SD card, or even optical media like a CD. If the medium is writable, Porteus will save all changes inside a folder and load them on subsequent boots. The distribution is based on Slackware Linux using a modified version of the Linux Live scripts that help cut down bootup and shutdown times. The latest release supports newer UEFI machines and also includes the latest Intel firmware fix for the Spectre vulnerability.

Instead of a single ISO, Porteus is available as seven separate ISO images, each with a different desktop – from the heavyweight KDE and Cinnamon to the lightweight LXQt, LXDE, Openbox, and others. All seven flavors of Porteus support both 32- and 64-bit architectures, and the ISOs of most editions weigh in around 300MB each. This is because the distribution exists in a compressed state on the storage media, and the familiar Linux directory structure is created on-the-fly during boot.

Another unique aspect of Porteus is its use of modules, which are precompiled packages that you activate and deactivate as per your requirements. If you move these app modules to a designated directory inside the removable storage, the app is available in subsequent boots as well. The modules infrastructure works well in conjunction with the traditional binary packages. Porteus uses the Unified Slackware Package Manager (USM), which lets you fetch apps from across five different Slackware repositories. When you fetch packages via USM, you also get the option to convert them into modules (Figure 1). You can then activate the modules to use the app and make them survive reboots as well.

Porteus has a unique mechanism for getting onto a writable medium. Instead of `dd`'ing its ISO file on to a USB stick, you need to loop mount it and copy its contents onto an ext4-formatted disk. Then run a script from inside the USB to make it bootable, and you're good to go. The process is explained in the official installation guide [2]. You can also install Porteus to an internal hard drive, though the developers don't recommend this option. Out of the box, Porteus doesn't ship with very many apps besides the ones that come with the desktop environment by default. But you can use USM and its repository to easily flesh out your environment and have them available on subsequent boots, which makes it very convenient (Figure 2).

The Porteus boot menu offers some interesting options. For example, you can use the boot menu to copy the entire Live environment to RAM, which requires more than 768MB of system memory. You can also opt to boot Porteus in text mode, initialize a PXE server, or access the Plop boot manager. By default, the Live session is persistent, which automatically saves all changes made to the Live environment. However, you do get the option to start a fresh session from the boot menu.

Puppy Linux

One of my all-time favorites, Puppy Linux [3], was one of the first of the miniscule distributions. Puppy had its first release way back in 2003. The distribution is built from the ground up and has grown beyond its original mandate of resurrecting older hardware that had been rendered useless due to lack of support in the mainstream distributions.

The project currently has four official versions. Three are based on Ubuntu's Bionic, Xenial, and Tahr releases, and one is based on Slackware 14.1. Despite their different base and package repositories, all variants use JWM, which is one of the lightest window managers.

You can transfer Puppy Linux to a rewritable medium like a USB disk or anchor it to your hard disk. The distribution's custom installer will help you with both these installation targets. The distribution supports two installation mechanisms. The default *frugal* mechanism copies Puppy files inside a partition that holds another distribution, while the *full* option takes over the entire partition and is recommended only for older machines. Puppy's installation process is rather unconventional and might even intimidate first-time users, but every step is well documented and easy to follow.

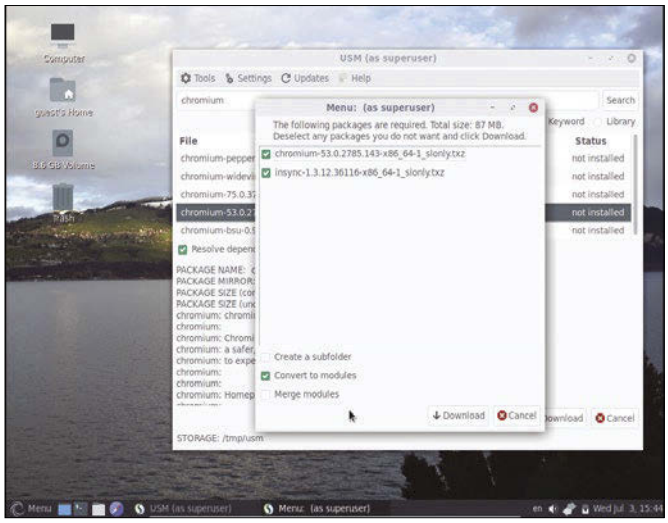


Figure 1: Depending on the desktop you choose, it can easily require up to 1GB of RAM to run Porteus.

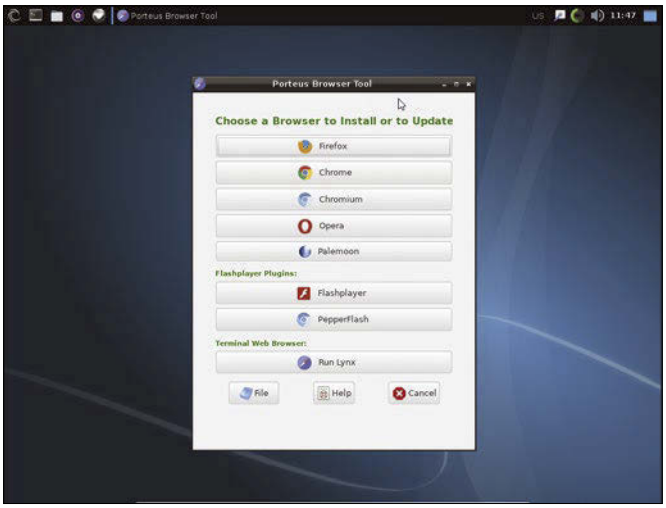


Figure 2: Porteus v5.0, which is currently available as RC1, includes a browser installation script.

Once you're done with the installation, you'll be amazed by the distribution's cache of apps. There's no beating Puppy for out-of-the-box functionality. You'll find an app for virtually every task that you can perform with a desktop computer. Puppy's menu includes popular lightweight options such as Claws Mail, the Pale Moon web browser, the AbiWord editor, the mpv multimedia player, and more (Figure 3). Interspersed with them are Puppy's own custom apps – primarily to ease administration tasks. For a miniscule distribution, Puppy has some apps that you wouldn't find even in full-fledged distributions, including apps to block online ads, a softphone for Internet telephony, a secure downloader, a DVD burning app, and lots more.

Puppy Linux packages are called pets and have a `.pet` extension. You can install packages using the distribution's custom Puppy Package Manager tool. By default, the package manager will fetch apps from the repositories of the Puppy variant you are using, but you can configure it to download packages from other Puppy repositories as well. You can also download SquashFS (SFS) files for some popular apps like Kodi, KdenLive, LibreOffice, WPS Office, and more (Figure 4). SFS files are compressed environments that package an app and all its dependencies. Think of them as modules in Porteus that can be activated and deactivated as well.

When you shut down Puppy for the first time, the distribution offers to save all changes inside a file that can optionally be encrypted for added security. Speaking of security, it should be noted that the distribution runs everything as the root user. The Puppy developers consider this to be a safe option but also give you the option to run the distribution as a non-privileged user, although the non-privileged option is currently considered experimental. Interestingly, Puppy Linux can also save changes from a Live session onto rewritable optical media as well. Puppy also scores highly for its ample documentation. The distribution bundles help documentation on several topics, such as working with Microsoft Office files, how to add codecs, installing software, and more. Help pages contain links to the documentation pages for most applications.

Slax

For a long time, Slax [4] was the go-to distribution for bootable USB disks. Back then, it was based on Slackware but was still relatively easy to customize. But then, like many one-man projects, it stopped receiving updates and was dormant for several years. Now in its second incarnation, Slax has undergone a dramatic change with its move to Debian, yet it retains all the features that made it stand out from the field of Live USB distros.

Slax is available for both 32- and 64-bit machines, and its ISO weighs in under 300MB. The distribution is getting regular updates and is gradually taking on new features, a majority of which are behind the scenes and focused on improving its compatibility with modern hardware.

Despite the fact that the projects are now heading in different directions, Porteus started out as a Slax remix, and they still

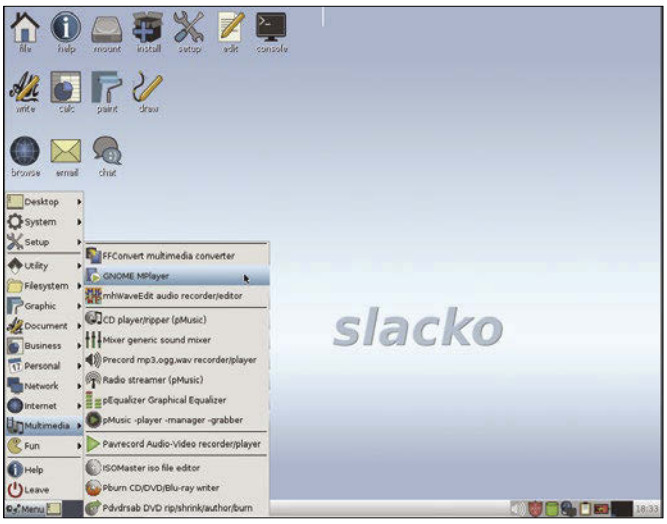


Figure 3: Not all Puppy variants ship with the same set of apps by default.

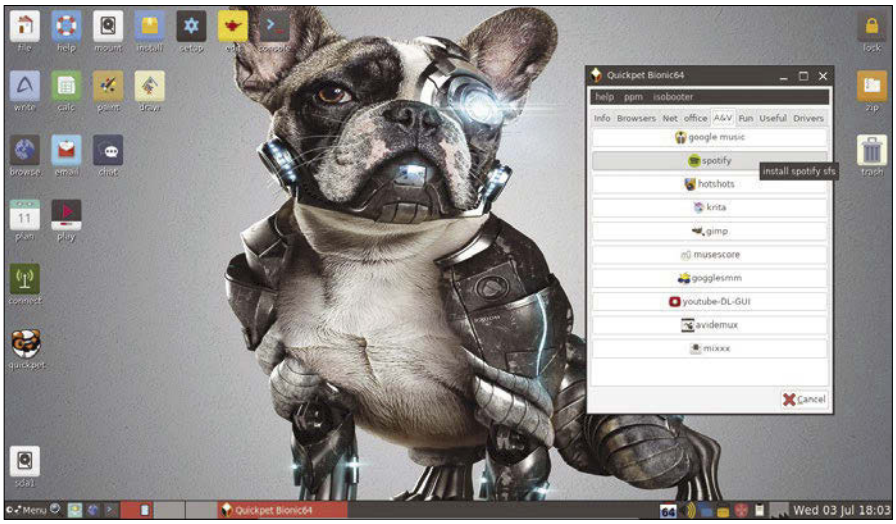


Figure 4: Some Puppy variants like Bionic Puppy include the custom Quickpet app to easily install popular applications.

share several traits. Slax, like Porteus, has an offbeat procedure [5] for getting it onto a USB disk. You'll have to loop mount it and copy its contents onto an ext4-formatted disk. Instead of the traditional installation process, you need to run a script from inside the USB to make it bootable, and you're good to go.

The distribution offers three boot options. By default, Slax will boot with the persistence option to save any changes made in the Live environment. You can, however, choose to boot into the factory environment by disabling persistence. The third option will run the Live environment entirely from RAM. Running Slax in memory will lead to the best performance, but it will only work on machines that have an adequate amount of RAM. Slax's website mentions that 512MB of RAM is required to hold all the Slax data, so you need about 1GB of RAM for a smooth functioning environment.

Despite its miniscule size, Slax boots to a graphical desktop that sports the lightweight Fluxbox window manager. The desktop runs Slax's homegrown and very minimalist Xlaunch application launcher that weighs only 25KB. The distribution ships with only a handful of apps, and there's no graphical package management app preinstalled (Figure 5). Slax is based on Debian, so you can use the Apt package management tool to pull in any number of apps.

Another trait Slax shares with Porteus is its modular architecture. Everything in Slax, from the core components to the web browser, is a module that users can enable or disable on-the-fly. The modular architecture makes it fairly simple to customize the distribution to suit your needs. Any changes you make to the distro, like changing the wallpaper or acquiring new apps with `apt-get`, will automatically be saved if you're running Slax from a writable medium like a USB disk.

On the other hand, if you're running Slax from a non-writable medium like a CD or have disabled persistence, you can still save your customizations by rolling them into a module. A simple `savechanges` command [6] will save all changes into a custom module (Figure 6). You can then place this module inside the USB, along with the other modules or use it to generate a new Slax ISO. Slax's developer has written scripts to reduce the process of creating a custom ISO down to a single command.

Slax, like Porteus, includes a PXE server that can again be activated with a single script. You can test this feature in VirtualBox by attaching the NIC of the Slax virtual machine (VM) to the internal network. Once the PXE server is running, any other VM that's connected to the internal network and set to boot from the network will boot into Slax by grabbing the Slax modules over the network.

SliTaz

SliTaz [7], which stands for "Simple, Light, Incredible, Temporary Autonomous Zone," has been chugging along for over a decade now. The rolling release distribution maintains two branches and is available in a stable and a developmental version, both of which are available for both 32- and 64-bit platforms.

The recommended image is about 53MB. Besides the official flavors, there are many other downloadable images for SliTaz, because its developers and community provide many variations to address different use cases and system limitations. For instance, there's a low RAM version for systems with as little

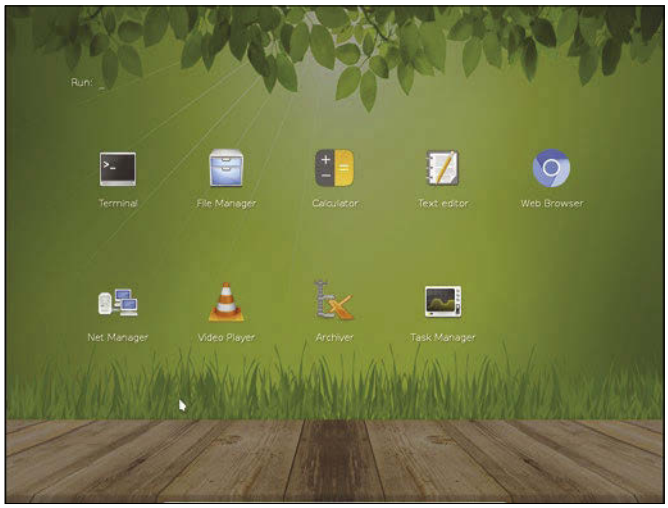


Figure 5: Slax doesn't have a graphical package manager, but you can pull in Synaptic from the repositories.

as 24MB RAM, a version with Firefox instead of Midori, a version with no extra applications, and so on.

The distribution has an extensive boot menu with options to boot into any of the five environments. The `gtkonly` and the `fulldesktop` options take you to a graphical desktop, while the `base` and `justx` options boot into a restricted environment to help you create your own customized version of SliTaz. There's also the web boot option (Figure 7) that will boot SliTaz straight from its mirrors. First time users should stick to the default option that will take you into a graphical desktop.

The distro uses the Openbox window manager and, despite its size, allows you to enable some desktop effects as well. Its menus are flush with all the regular desktop apps, including web browsers, audio players, media editors, several development tools, and more.

SliTaz is built with a set of home-brewed tools known as Cookutils and uses BusyBox for many of its core functions. The highlight of the distribution are the bunch of custom tools, such as TazPanel, which is the distribution's control center. You can use TazPanel to administer all aspects of the system. It's an all-in-one app that provides access to system configura-

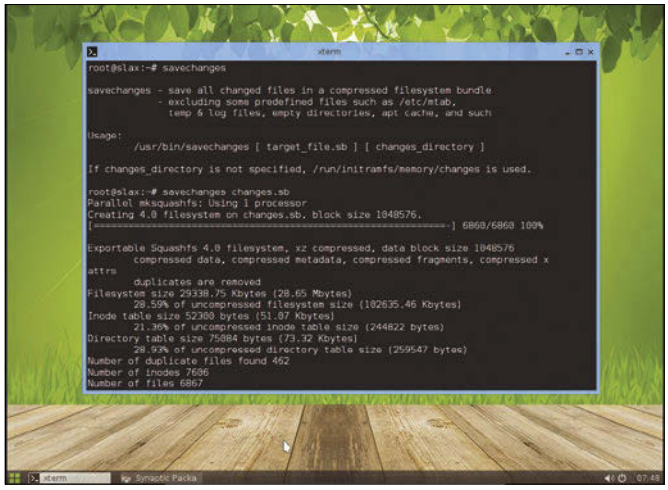


Figure 6: The Slax custom scripts have enough documentation to orient first-time users.

Figure 7: The Web Boot option helps you try the various SliTaz editions without writing different media.

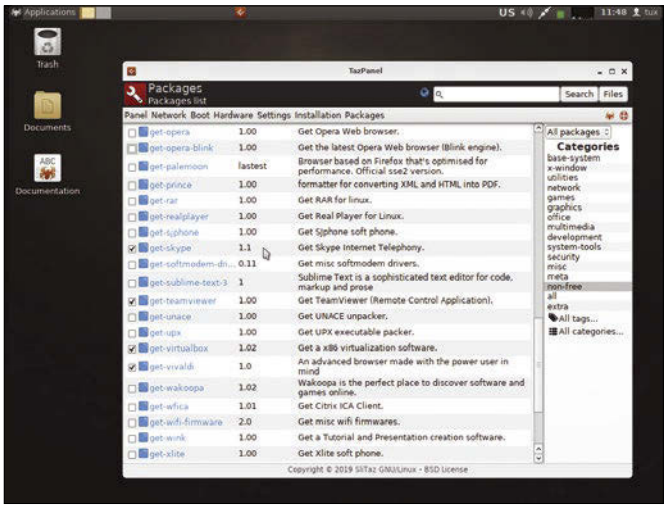


Figure 8: SliTaz uses the same installer irrespective of whether you wish to install it inside a dedicated partition or inside a folder in Windows.

tion, hardware detection, user and group management, system updates, and application installation. You can also anchor SliTaz to your hard disk, and Windows users can run it from inside a directory. The process involves creating a partition in your hard disk via GParted. The installer also enables you to create a separate `/home` partition along with a non-root user.

The distribution's custom repositories hold about 5,000 packages that you can fetch via the TazPanel app (Figure 8). In addition to open source apps, you'll also find several non-free apps as well, including Skype, Google Earth, Dropbox, and more. If you need an app that isn't listed, use Slitaz's command-line `tazpkg` package manager [8] to create SliTaz-compatible packages from popular packaging formats, including `.deb`, `.rpm`, `.tgz`, and even from other small distributions, such as Puppy Linux, Slax, and Tiny Core.

Another useful custom tool is TazLiTo, which helps you roll customized spins of the distribution with additional packages and custom wallpaper. TazLiTo and the other SliTaz custom tools are very well-documented and fairly intuitive to operate, even by first-time users.

Tiny Core

One of the smallest distributions in this feature, Tiny Core [9] is available in three flavors. There's a miniscule 13MB Core edition that will help advanced users craft a custom installation. The recommended Tiny Core edition weighs in at 19MB and boots to a graphical desktop. At the top end is the 232MB CorePlus edition that has additional drivers for wireless cards, a remastering tool, and localization support (Figure 9).

Download Tiny Core and transfer it to USB with `dd` or Ether. Due to its minuscule size, Tiny Core boots blisteringly fast. True to its name, Tiny Core bundles just a terminal, a text editor, and an app launcher on top of the lightweight FLWM window manager. It has a control panel to manage bootup services and configure the launcher. If you need anything else, you'll have to pull it in using the distribution's package manager, including the installer, if you want to install Tiny Core to your hard disk.

Installing Tiny Core is easy, once you download the *tc-install* or *tc-install-GUI* application. The distribution offers several installation mechanisms, including a frugal installation to the hard disk, as well as an option for installing within an existing Linux partition. Also, Tiny Core has a unique package installation process (Figure 10). When installing a new application, you can choose between having the package load into the distribution automatically at boot time or on demand. Choosing to load a package at boot makes it available to you immediately after a reboot. On the other hand, choosing to load it on demand will speed up bootup times as the app isn't loaded or mounted in Tiny Core parlance. To use an on-demand app, you'll first have to load it manually.

But the distro's stellar performance comes at the price of usability. The distribution is designed for advanced users, and you'll need to spend some time fleshing it out using its package manager, which isn't the most intuitive in the business. You'll also have to browse through the documentation to familiarize yourself with the distribution's peculiarities, irrespective of your experience with Linux. One highlight of Tiny Core is its remaster tool, which lets you create your own remix of the distribution. Like everything else in the distribution, the remaster tool is fairly powerful and feature rich, but it isn't the most intuitive.

Conclusion

As I mentioned at the start, all the small distributions in this article have their uses. Some will help revive an old unused computer, and some will help you get the maximum performance out of an adequately equipped machine, while others will help you carry a familiar environment in a portable medium. (See Table 1 for a comparison.)

Tiny Core Linux is the leanest of the lot. You can use Tiny Core on the most resource-strapped computer, but it does have its shortcomings. For starters, the distribution doesn't ship with any real applications, and you'll have to spend time with its quirky tools, converting the basic installation into a usable desktop. You'll have to download the installer as well, which doesn't ship with the recommended Tiny Core variant. You'll also have to familiarize yourself with Tiny Core's way of doing things.

SliTaz is slightly less esoteric. The distribution does ship in a more usable shape straight out of the box; you'll still have to

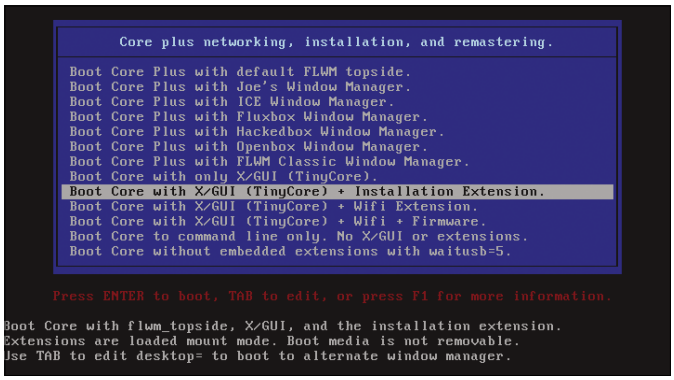


Figure 9: Tiny Core Plus has a lot more bootable options and is still relatively tiny as a 106MB image.

spend some time with the package manager to tailor the environment to your requirements. Also, in order to get the most from the distribution you'll have to read through the documentation and spend some time familiarizing yourself with some of the custom tools.

In my opinion, Puppy Linux is still the top dog when it comes to pressing old computers back into active duty. On the plus side, I particularly like Puppy's default software collection, which is simply unrivalled by any other distribution in this article. Unless you have very particular needs, it'll be quite a while before you fire up Puppy's package management app. On the flip side, Puppy might offer too many options for a new user.

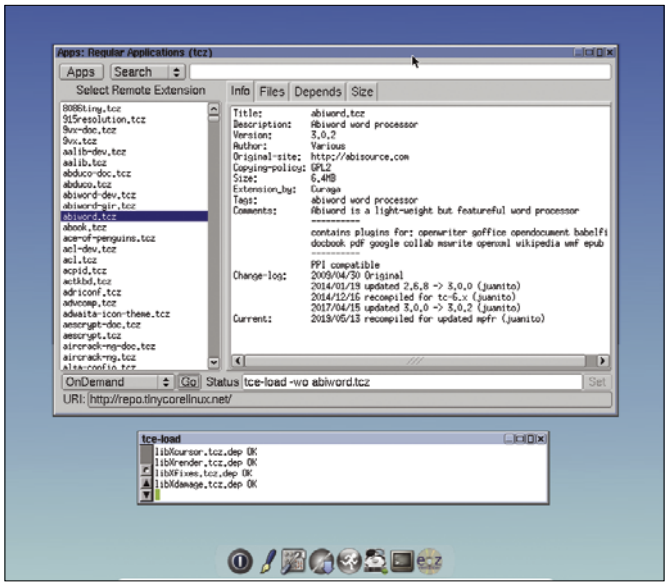


Figure 10: Tiny Core's sole developer, Robert Shingledecker, was previously involved with the once-popular but now-dormant Damn Small Linux project.

Although all the options in the article can run from a USB, I personally recommend Porteus and Slax as the best options if you're looking to take a familiar environment with you. Porteus scores over Slax, because it ships in a more ready-to-use state than Slax's bare-bones environment. Another deciding factor between the two can be your familiarity with the underlying base distro. Debian users might prefer Slax, but with the graphical USM package manager, Porteus' use of Slackware will appeal to many users. You can, of course, use the customization tools to spin custom versions.

Note however that while you'll have no issues getting both Porteus and Slax to boot on older machines with BIOS, getting them to work on newer UEFI-equipped computers is a little tricky. Porteus now supports UEFI firmware, but to get it to boot on such machines, you'll have to disable the SecureBoot feature. On the other hand, the developer of Slax has promised to add support for UEFI firmware once the project gets 50 patrons on Patreon. Several users on the forum boards of both distros have suggested workarounds, but I haven't been able to get them to work reliably on all my machines. I would caution against disabling the SecureBoot security feature unless you know what you're doing. ■■■

Info

[1] Porteus: *http://porteus.org*

[2] Porteus installation: *http://www.porteus.org/component/content/article/26-tutorials/general-info-tutorials/114-official-porteus-installation-guide.html*

[3] Puppy Linux: *www.puppylinux.com*

[4] Slax: *www.slax.org*

[5] Slax USB: *https://www.slax.org/starting.php*

[6] Slax savechanges command: *https://www.slax.org/customize.php*

[7] SliTaz: *www.slitaz.org*

[8] SliTaz package manager: *http://doc.slitaz.org/en:handbook:packages*

[9] Tiny Core Linux: *www.tinycorelinux.net*

Table 1: Small Distro Roundup

	Porteus	Puppy Linux	Slax	SliTaz	Tiny Core
Based on	Slackware	None	Debian Stretch	None	None
Installation	Involved	Straightforward	Involved	Involved	Involved
Desktop Environment	Various	JWM	Fluxbox	Openbox	FLWM
Package Management	Graphical	Graphical	Text	Graphical	Graphical
Ratings					
Bundled Apps	2	5	2	4	2
Usability	4	5	4	4	4
Support	3	4	3	4	3
Documentation	3	4	3	4	4
Ratings are based on a five-point scale with 1 being the lowest and 5 the highest.					

Exploring the new nftables firewall tool – a successor to iptables

Traffic Rules

The nftables firewall utility offers a simpler and more consistent approach for managing firewalls in Linux. *By Frank Hofmann*

If you are training to become a network administrator, or even if you just want to get better at Linux, you cannot avoid dealing with the topic of firewalls, including the rules for filtering packets on the network.

The iptables firewall tool [1] is slightly long in the tooth, and the program code in particular has become more and more complex. Small changes in the project core affected all the tools associated with the project. Iptables, ip6tables, ebtables, and arptables all originate from the same codebase – not in the form of modules, but by code duplication. Accordingly, the four tools drifted apart over time. Iptables was best maintained, and ebtables was neglected. Bugs patched in iptables still existed in Ebtables years later.

The problems with maintaining the iptables code base prompted the development of a successor called nftables [2] back in 2009 by the netfilter project [3]. The first two letters of nftables are derived from the project; nftables simply means "netfilter tables." The stated development goals include higher data throughput, greater scalability with a view to changing requirements, and, in particular, a modular structure that is easier to maintain. Since Linux 3.13 (January 2014), nftables has existed directly in the kernel [4]. The nftables firewall tool uses internal, proven components of the netfilter project.

As of the release of Debian 10 "Buster," which is planned for summer 2019, Debian will completely rely on nftables [5], which will also affect derivatives like Ubuntu and Linux Mint. Red Hat-based distros are also moving to more reliance on nftables. The current releases of most Linux distributions already contain nftables – not enabled, but ready for use.

Extensions and Conversions

You can create rules for firewalls with the command-line tools `iptables` (IPv4), `ip6tables` (IPv6), `arptables` (ARP packets), and `ebtables` (Ethernet frames). Nftables replaces all four with a single command-line tool named `nft`, which now sets all the rules for accepting, forwarding, modifying, or rejecting packets from the network on the system.

Iptables uses different filters and three processing chains named `INPUT`, `FORWARD`, and `OUTPUT` to handle packets. The nftables framework does not have any similar built-in chains – you'll have to define the chains yourself.

Nft uses two libraries: *libmnl*, a minimalist Netlink library [6], and *libnfnetlink*, a Netlink library in userspace [7]. As a result, the size of the code in the Linux kernel is smaller, and small changes to `nft` do not mean that you need to customize the kernel [8].

To make sure that the appropriate kernel module is loaded in your system kernel, work with the output from the `modinfo` (Figure 1) and `lsmod` (Figure 2) commands. The feedback in the exam-

Author

Frank Hofmann works on the road, preferably in Berlin, Geneva, and Cape Town, as a developer, trainer, and author. He is co-author of the *Debian Package Management Book* (*http://www.dpmb.org/index.en.html*).

Acknowledgements

The author thanks Axel Beckert and Werner Heuser for their suggestions and criticism during the preparation of the article.

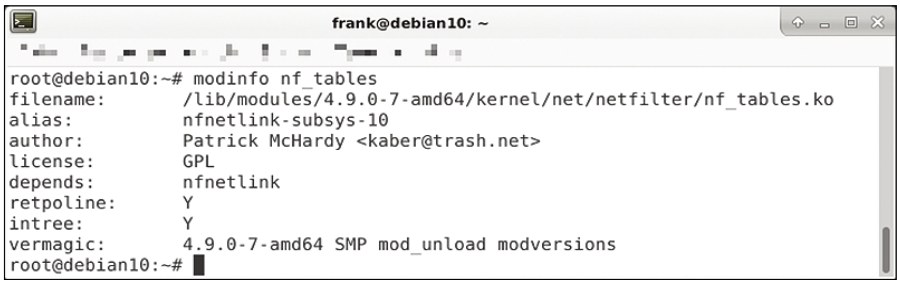


Figure 1: The output from the `modinfo` command returns information about the kernel module.

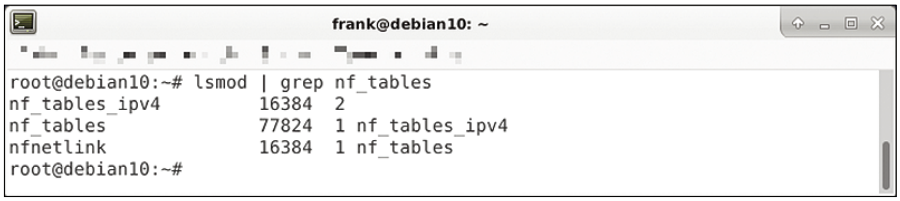


Figure 2: `lsmod` tells you whether the kernel module has been loaded by the system.

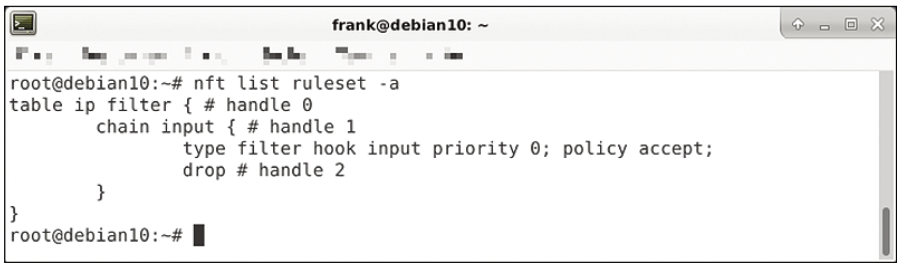


Figure 3: The command `nft list ruleset -a` lists all set rules.

ples shown is positive and lets you get started with `nft` directly.

Basic Configuration

Nftables starts with a completely empty ruleset; there are no predefined tables, chains, or rules. As the user (or admin), you first create the tables, add chains that latch into the Linux kernel as netfilter hooks, and then populate them with appropriate rules. All these steps are performed using the `nft` command, which you execute as root.

Listing 1 demonstrates how to define a firewall that does not (yet) let packets through. The first command (line 1) creates a table for IP packets of type `filter`. Line 2 adds a chain to the `filter` table. In line 3, a rule is added to the chain to discard all packets (`drop`).

The command in line 4 gives an overview with all the rules on the firewall (Figure 3). In addition to the entries, there are comments in the form `# handle handle_number`, which you use to reference the entries. This option is of particular interest if you want to delete or change existing specifications, or insert new ones before or after them. For example, the command from line 5 deletes the drop rule.

Basic Approach

`nft`'s developers chose the Berkeley Packet Filter (BPF) [9] to define the nomenclature of their rules, and they orient their work on the classic `tcpdump` [10], so that you don't have to relearn everything [11].

`nft` provides a number of address families: `arp` (ARP), `bridge` (previously provided by ebtables), `inet` (includes IPv4 and IPv6), `ip` (for IPv4), `ip6` (for IPv6), and `netdev` are predefined. `netdev` is used to filter incoming packets before they reach Layer 3, according to the ISO/OSI specification.

`nft` translates the rules and keeps them in a small virtual machine (`nftables core`) for communication with the Linux kernel.

The example in Listing 2 demonstrates how to enable port 22 for incoming packets as needed for access via SSH. Thanks to `nft`, the overhead is reduced to a single command, and the syntax is simpler.

If you want to add two ports, 80 and 443 (i.e., HTTP and HTTPS), you need two extra lines per port for iptables. With `nft`, however, you just need to extend the existing line to combine all three protocols at once. Port 22 is added in braces, followed by the two ports, 80 and 443 (Listing 3), separated by commas.

Listing 1: Setting Up a Chain

```
01 # nft add table ip filter
02 # nft add chain ip filter input {type filter hook input priority 0\;}
03 # nft add rule ip filter input drop
04 # nft list ruleset -a
05 # nft delete rule ip filter input handle 2
```

Listing 2: Enabling Port 22

```
### Allow incoming packets on port 22.
### With Iptables:
# iptables -A INPUT -p tcp --dport 22 -m conntrack --ctstate NEW,ESTABLISHED -j ACCEPT
# iptables -A OUTPUT -p tcp --sport 22 -m conntrack --ctstate ESTABLISHED -j ACCEPT
### With Nft:
$ nft add rule inet filter input tcp dport 22 ct state new,established accept
```

Listing 3: Adding Ports

```
# nft add rule inet filter input tcp dport { 22, 80, 443 } ct state new,established accept
```

Please note that the spaces within the brackets must be exactly as shown – otherwise Bash will swallow them up and complain. Users of Zsh run into the same problem, which you can resolve with suitable quoting.

Save and Restore

As with iptables, the nftables configuration can be saved to a file. The first command from Listing 4 writes the current ruleset to the `firewall.config` file, while the second command reads the configuration again.

To make sure that there are no other (possibly interfering) rules in the cache before initializing the firewall, add a line that reads `flush ruleset` at the beginning of the `firewall.config` configuration file.

Translation

Humans are known to be creatures of habit who have difficulty with change. The `iptables-translate` and `ip6tables-translate` commands are two tools that help you get used to the new environment. These commands convert the nomenclature of iptables firewall rules to nftables format (Listing 5). You can convert both for individual statements and complete rulesets.

Listing 4: Writing the Ruleset to a File

```
# nft list ruleset > firewall.config
# nft -f firewall.config
```

Conclusions

Nftables helps you accommodate several complex tools under one roof, thus making it easier to secure the network. You'll want to try out the new firewall rules before you deploy them on your network. One option is to create a virtual test network using VirtualBox or the smart application Mininet [12]. Or, you could set up a pack of Raspberry Pis on a small, isolated network to simulate a real-world configuration. ■■■

Info

[1] Iptables: *https://netfilter.org/projects/iptables/index.html*
[2] Nftables: *https://netfilter.org/projects/nftables*
[3] Netfilter project: *https://netfilter.org*
[4] "Why you will love nftables": *https://home.regit.org/2014/01/why-you-will-love-nftables*
[5] Nftables in Debian: *https://wiki.debian.org/nftables*
[6] libmnl: *https://git.netfilter.org/libmnl*
[7] libnfnetlink: *https://git.netfilter.org/libnfnetlink*
[8] "Iptables vs. nftables": *https://digitalglarus.ch/en-us/cms/blog/2018/08/19/iptables-vs-nftables*
[9] Berkeley Packet Filter (BPF): *https://en.wikipedia.org/wiki/Berkeley_Packet_Filter*
[10] tcpdump: *https://www.tcpdump.org*
[11] "Differences between iptables and nftables explained": *https://linux-audit.com/differences-between-iptables-and-nftables-explained/*
[12] Mininet: *http://mininet.org*

Listing 5: Converting Iptables Rules

```
$ iptables-translate -A INPUT -p tcp --dport 22 -m conntrack --ctstate NEW -j ACCEPT
nft add rule ip filter INPUT tcp dport 22 ct state new counter accept
$ ip6tables-translate -A FORWARD -i eth0 -o eth3 -p udp -m multiport --dports 111.222 -j ACCEPT
nft add rule ip6 filter FORWARD iifname eth0 oifname eth3 meta l4proto udp udp dport { 111,222} counter accept
```

Unit testing Go code with mocks and dependency injection

Better Safe than Sorry

Developers cannot avoid unit testing if they want their Go code to run reliably. Mike Schilli shows how to test even without an Internet or database connection, by mocking and injecting dependencies. *By Mike Schilli*

Not continuously testing code is no longer an option. If you don't test, you don't know if new features actually work – or if a change adds new bugs or even tears open old wounds again. While the Go compiler will complain about type errors faster than scripting languages usually do, and strict type checking rules out whole legions of careless mistakes from the outset, static checks can never guarantee that a program will run smoothly. To do that, you need a test suite that exposes the code to real-world conditions and sees whether it behaves as expected at run time.

Ideally, the test suite should run at lightning speed so that developers don't get tired of kicking it off over and over again. And it should be resilient, continuing to run even while the Internet connection on the bus ride to work occasionally drops. So, if the tests open a connection to a web server or need a running database, this is very much out of line with the idea of fast independent tests.

However, since hardly any serious software just keeps chugging along by itself without a surrounding infrastructure, it is important for the test suite to take care of any dependencies on external systems and replace them with Potemkin villages. These simulators (aka "mocks") slip into the role of genuine communication partners for the test suite, accepting its requests and returning programmed responses, just as their real world counterparts would.

What Can Go Wrong?

Listing 1 [1] shows a small library with the `Webfetch()` function, which expects a URL for test purposes and returns the content of the page hiding behind the URL. What could possibly go wrong when you're just fetching a website? First of all, the specified URL may not comply with the standardized format. Then there could be problems contacting the server: errors in DNS resolution, network time outs, or the server might just be taking a powernap. Or maybe the given URL does not refer to a valid document on the server, which then responds with a `404`, or it requests a redirect with a `301`, for example.

The code in Listing 1 checks for all of these potential errors and returns an `error` type if it finds one. Once the client has finally tapped into the stream of incoming bytes from the network as of line 23, it can happen that the stream is suddenly interrupted because the network connection breaks down. A good client should field all of these cases, and a good test suite should verify that the client does so in all situations.

No Fuss

Now the test-suite might run on systems that don't have a reliable Internet connection – nothing is more annoying than a program that sometimes works properly and sometimes doesn't. To avoid such dependencies, test suites often replace external systems with small-scale responders. In what is known as mocking, simple test frameworks mimic certain capabilities of external systems in a perfectly reproducible manner. For example, a simplified local web server is used that only delivers static pages or only reports error codes.

Programs in Go can even run a server in the same process as the test suite, thanks to its quasi-simultaneous Go routines. With no need to start an external

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.

process, this is incredibly convenient, because the whole time-consuming and error-prone fuss about starting and, above all, shutting down external processes properly even in erroneous conditions is no longer required.

Conventions

Listing 2 checks if the `Webfetch()` function from Listing 1 works and if the server delivers a text file as advertised. To do this, it defines the `TestWebfetchOk()` function as of line 18. The function expects a pointer to Go's standard testing data structure of the `testing.T` type as a parameter, which it later uses to report errors to the test suite.

It should be noted that Go insists on sticking to conventions and stubbornly ignores anything that deviates from them. The names of all test files must end with `_test.go`. So, it must be `webfetch_200_test.go` and not, say, `webfetch_test_200.go`, because otherwise the command `go test` would silently ignore the file and therefore wouldn't find any tests to execute. On top of that, the names of the test suite's test routines must start with `func Test<tXXX>`, otherwise Go won't recognize them as unit tests and won't run them at all.

And, finally, the "one package per directory" rule always applies in Go: All three Go programs, the `webfetch.go` library and the two files with the unit tests, all show the `package webfetcher` package at the beginning of their code.

The properly defined `TestWebfetchOk()` function in Listing 2 becomes part of the test suite, and the error checks in the `if` statements in lines 23 and 27 become its test cases. Line 23 verifies that the server sent an OK status code of `200`, and line 27 compares the received string with the mock server's hard-coded string, specified in line 10 (`"Hello, client."`). In both cases, the test suite says nothing if everything runs smoothly and only reports any errors that may occur.

If you prefer your test routines to be a bit more talkative, you can use the `t.Logf()` function before each test case to display a message hinting at the test about to be performed; you can also generate some output in case of success. As implemented in Listing 2, the test suite called in verbose mode with

```
go test -v
```

does not give you any details on the individual test cases if everything passes, except that it reports the executed test functions (Figure 1). But if something were to go wrong, the error messages output with `t.Errorf()` would rear their ugly heads.

The `Always200()` handler in line 12 of Listing 2 defines the behavior of the built-in test web server. No matter what the incoming type `*http.Request` request looks like, it simply returns a status code of `http.StatusOK` (that is, `200`) in the header of the HTTP response and adds the `"Hello, client."` string as the page

Listing 1: webfetch.go

```
01 package webfetcher
02
03 import (
04   "fmt"
05   "io/ioutil"
06   "net/http"
07 )
08
09 func Webfetch(url string) (string, error) {
10   resp, err := http.Get(url)
11
12   if err != nil {
13     return "", err
14   }
15
16   if resp.StatusCode != 200 {
17     return "", fmt.Errorf(
18           "Status: %d", resp.StatusCode)
19   }
20
21   defer resp.Body.Close()
22
23   body, err := ioutil.ReadAll(resp.Body)
24   if err != nil {
25     return "", fmt.Errorf(
26               "I/O Error: %s\n", err)
27   }
28   return string(body), nil
29 }
```

Listing 2: webfetch_200_test.go

```
01 package webfetcher
02
03 import (
04   "fmt"
05   "net/http"
06   "net/http/httptest"
07   "testing"
08 )
09
10 const ContentString = "Hello, client."
11
12 func Always200(w http.ResponseWriter,
13                 r *http.Request) {
14   w.WriteHeader(http.StatusOK)
15   fmt.Fprint(w, ContentString)
16 }
17
18 func TestWebfetchOk(t *testing.T) {
19   srv := httptest.NewServer(
20            http.HandlerFunc(Always200))
21   content, err := Webfetch(srv.URL)
22
23   if err != nil {
24     t.Errorf("Error on 200")
25   }
26
27   if content != ContentString {
28     t.Errorf("Expected %s but got %s",
29       ContentString, content)
30   }
31 }
```

```
$ go test -v
=== RUN   TestWebfetchOk
--- PASS: TestWebfetchOk (0.00s)
=== RUN   TestWebfetch404
--- PASS: TestWebfetch404 (0.00s)
PASS
ok      go-testing/eg   0.003s
$
```

Figure 1: The test suite checks both success and failure conditions of the `Webfetch` library.

content. Line 19 starts the actual web server from the `httptest` package, converts the previously defined handler to the `http.HandlerFunc` type, and hands it over to the server as a function.

The server's `URL` attribute specifies the host and port on which the new server is listening, which Line 21 hands over as a URL to the `Webfetch()` client function to be tested. From the client's point of view, a status code 200 and the previously set string are returned as expected, and the test suite does not raise an error.

No False Positives

But `Webfetch()` also needs to handle error cases correctly. To check this, Listing 3 defines the `Always404()` handler, which tells the web server to send a `404` status code to the client for each request, plus a page of empty content. Quickly added to the new web server starting in line 15, `Webfetch()` now receives "Not Found" messages from the server and ensures this is actually happening in the `if` conditions starting at line 19 of Listing 3.

Independence

Alas, elegant inline servers with configurable handlers are not always available for all use cases. What do you do, for example, if a system needs a database? It is important to make sure that the dependency of the main program on the database is not hard-wired somewhere inside the system, but can be tinkered with from the outside. The procedure is known as "Dependency Injection" and feeds new objects with structures that define external targets while constructing the objects. This can lead to real mazes of dependencies with larger software architectures, which is why Uber [2] and Google [3] have already written packages to deal with it at scale.

Injection: This Isn't Going to Hurt

In order to avoid headaches for a library's end user, many developers would try to abstract as many details as possible in their software designs on first consideration. An example storage service for names, `namestore`, with a connected database as shown in Listing 4, would not initially reveal that it uses a database at all, but would simply create and manipulate the necessary scaffolding behind the drapes of the `NewStore()` constructor.

But this has fatal consequences for unit tests, which can no longer use the `namestore` package's interface for their tricks: For example, using a test-friendly SQLite database or even a driver for CSV format as the back end instead of a MySQL database that `namestore` might be using by default, which the test suite would then need to install and start in a complex process.

From a test-friendly design point of view, it makes more sense to have the library user pass the dependencies (such as the database used) to the constructor as shown in Listing 5, where the user opens the database (in this case SQLite) first and then passes the database handle to the `namestore` object's constructor, which then uses it for its inner workings.

The implementation of a library that is unit test-friendly is shown in Listing 6. As a data container to pass the database handle and potentially other items to the library, line 8 defines the `Config` type structure, which the `NewStore()` constructor expects in line 12. The constructor then returns a pointer to it to the caller, and object methods like `Insert()` from line 16 can then be called by using

Listing 3: webfetch_404_test.go

```
01 package webfetcher
02
03 import (
04 "net/http"
05 "net/http/httptest"
06 "testing"
07 )
08
09 func Always404(w http.ResponseWriter,
10   r *http.Request) {
11   w.WriteHeader(http.StatusNotFound)
12 }
13
14 func TestWebfetch404(t *testing.T) {
15   srv := httptest.NewServer(
16     http.handlerFunc(Always404))
17   content, err := Webfetch(srv.URL)
18
19   if err == nil {
20     t.Errorf("No error on 404")
21   }
22
23   if len(content) != 0 {
24     t.Error("Content not empty on 404")
25   }
26 }
```

Listing 4: main-wrong.go

```
01 package main
02
03 import (
04   ns "namestore"
05 )
06
07 func main() {
08   nstore := ns.NewStore()
09   nstore.Insert("foo")
10 }
```

Listing 5: main.go

```
01 package main
02
03 import (
04   "database/sql"
05   _ "github.com/mattn/go-sqlite3"
06   ns "namestore"
07 )
08
09 func main() {
10   db, err :=
11     sql.Open("sqlite3", "names.db")
12   if err != nil {
13     panic(err)
14   }
15
16   nstore := ns.NewStore(ns.Config{Db: db})
17   nstore.Insert("foo")
18 }
```

them as "receivers," as in `nstore.Insert()` in line 17 of Listing 5. This way, `Insert()` in line 16 in Listing 6 gains access to the database connection previously defined by the user in `config.Db`.

As you can see, the unit-test-friendly design ensures now that tests can be carried out by injecting different dependencies, either mocks or alternative databases, avoiding additional installation overhead, and making sure the tests are running super fast.

Embedded Examples

Who reads manual pages?! Most users dread working through abstract descriptions of what a system does and prefer working examples of applications that can be copied directly and quickly adapted to individual needs. These example sections are usually hidden in the depths of the instructions, prompting many users to first scroll all the way to the bottom. Unfortunately, it's quite common that these application examples were neglected during development and don't work (anymore), because the developer changed the code and forgot to add the changes to the matching section on the manual page. This is remedied by Go's automatic manual page creator with examples that can be added to be continually tested in the unit test suite.

Figure 2 shows a web version of the manual page, featuring automatically generated and continually tested examples, which were embedded in the original Go code by the developer. The following command:

```
godoc -http=:6060
```

launches a webserver on port 6060. If you point a browser at `http://localhost:6060/pkg`, you can maneuver to the man page for the package. When a user clicks on the down arrow next to the word *Example*, this opens a section showing both the sample code and the results expected on the standard output.

The highlight is that this example was generated automatically from the test code of the package in Listing 7 by the `godoc` command above. And, on top of that, using `go test`, the sample code is actually run in the test suite, and its outcome is compared with what is documented – so the documentation always remains up to date!

The way this whole thing works is that Go can identify the `ExampleSayHello()` function in Listing 7 as an application example useful for the documentation. How? It simply sees the `Example` prefix of the function in the test file (suffix `_test`). The web version of the `godoc` command then puts it in place with the surrounding documentation, partly extracted from programmer comments and automatic code introspection. By convention, Go interprets the comment in line 5 of Listing 7 (`"// Output: hello"`) as the example function's expected output, and `go test` executes `ExampleSayHello()` and checks whether `hello` actually appears on the standard output.

Along with the documentation automatically generated from code and comments in the library file in Listing 8, this results in self-checking documentation. It's a clear benefit for programmers who don't like to read instructions but just go directly to cut and paste. But software maintainers also benefit, as these checks make sure that embarrassing errors, such as the first application example in the documentation failing to run, no longer occur. ■■■

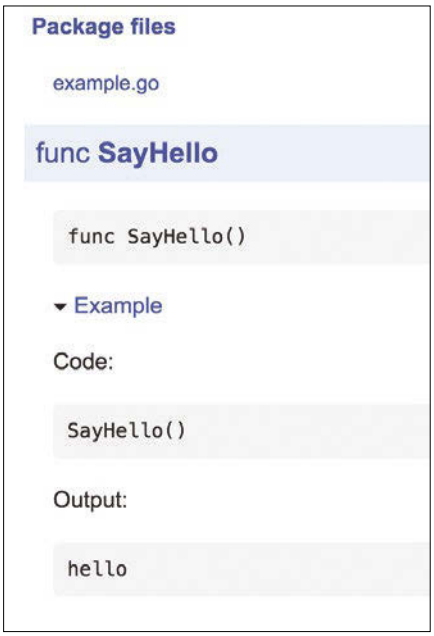


Figure 2: Examples show both sample code and standard output.

Listing 6: namestore.go

```
01 package namestore
02
03 import (
04   "database/sql"
05   _ "github.com/mattn/go-sqlite3"
06 )
07
08 type Config struct {
09   Db *sql.DB
10 }
11
12 func NewStore(config Config) (*Config) {
13     return &config
14 }
15
16 func (config *Config) Insert(
17     name string) {
18   stmt, err := config.Db.Prepare(
19       "INSERT INTO names VALUES(?)")
20   if err != nil {
21     panic(err)
22   }
23
24   _, err = stmt.Exec(name)
25   if err != nil {
26     panic(err)
27   }
28
29   return
30 }
```

Listing 7: example_test.go

```
01 package myhello
02
03 func ExampleSayHello() {
04     SayHello()
05     // Output: hello
06 }
```

Listing 8: example.go

```
01 package myhello
02
03 import (
04     "fmt"
05 )
06
07 func SayHello() {
08     fmt.Println("hello")
09 }
```

Info

[1] Listings for this article: *ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/226/*

[2] Uber's dig: *https://github.com/uber-go/dig*

[3] Google's wire: *https://github.com/google/wire*

Create snapshots with Timeshift

Time Jumping

Timeshift lets users easily create, manage, and restore system snapshots. *By Ferdinand Thommes*

Ideally, you should design a data preservation strategy in such a way that you only have to think about it once. A system snapshot is a popular component of the backup strategy for many organizations.

A snapshot copies the system state at a moment in time. Snapshots are very fast, so they don't tie up the system for long, but experts warn that a snapshot does not replace the need for a backup. Snapshots depend on pointers and other properties of the filesystem to preserve the system state, but if the underlying filesystem is damaged, you won't be able to restore the snapshot.

For many scenarios, however, a snapshot is a fast and easy way to bring the system back. If you make a change to your configuration or install a new driver and the system ceases to function, you can restore it to a previous state with a snapshot. Linux has several tools for creating system snapshots, including the uncomplicated Timeshift [1], as well as the command-line-only solutions CYA [2] and Snapper [3]. You can think of these Linux tools as similar to System Restore on Windows and Time Machine on macOS.

Timeshift was first released six years ago and has two options for creating snapshots in a graphical interface. If the installation of the operating system is based on a conventional filesystem, Timeshift relies on the powerful rsync protocol. In addition, the software also gives you the option of using Btrfs to handle snapshots. The operating system needs to support a layout with Btrfs subvolumes [4]. We tested both approaches.

Timeshift is available for RPM- or DEB-based operating systems, as well as for Arch Linux and its derivatives. The installation and setup are quite simple. If the distribution you are using does not offer a Timeshift package, use the distribution-independent installer [5] with the `.run` extension.

Configuration

After launching the application for the first time, go to the Settings menu for configuration. You will find five tabs: *Type*, *Loca-*

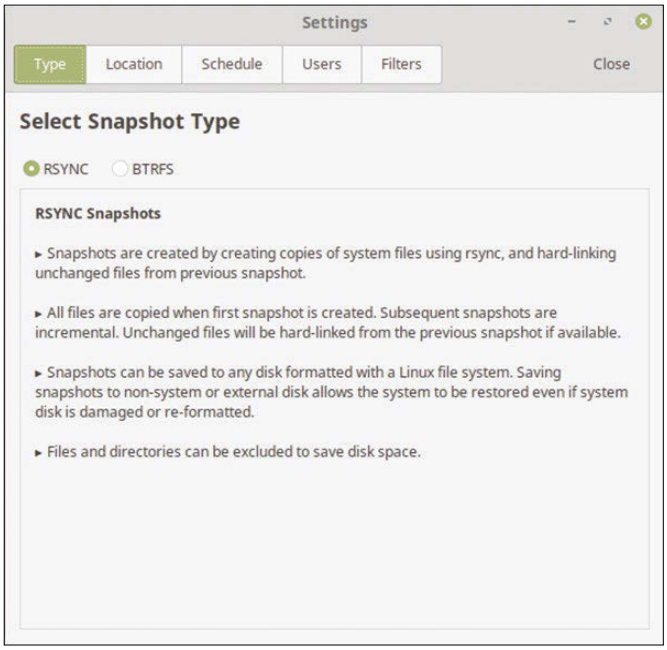


Figure 1: In the *Type* tab, select *RSYNC* for all filesystems except Btrfs.

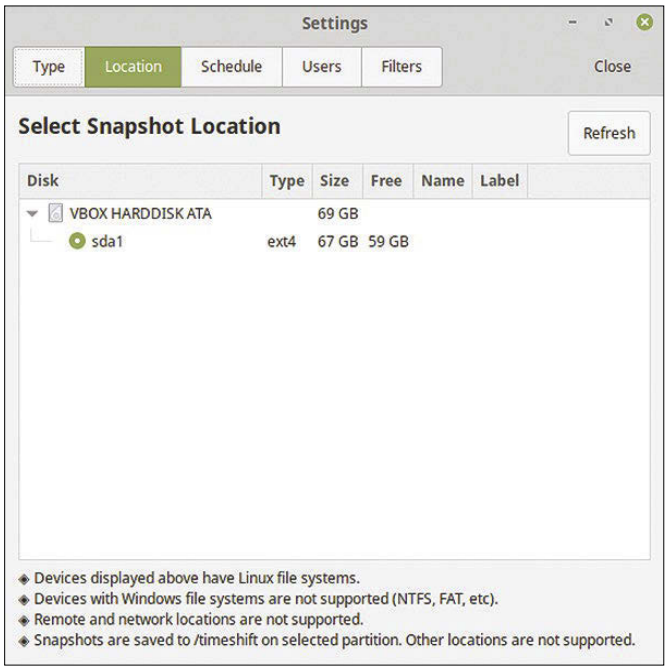


Figure 2: The *Location* tab only shows you usable Linux partitions. If possible, you should specify a storage location other than the system and home partitions.

tion, *Schedule*, *Users*, and *Filters*. First, you have to decide if you want to use rsync or Btrfs to create the snapshots; in the majority of cases, rsync is used.

In the *Type* tab, click on the arrow below the selection field to find information on both methods (Figure 1). Looking at the more conventional method with rsync first, Timeshift uses rsync to create snapshots in the form of copies of modified files and hard links [6] to unchanged files from previous snapshots. So only the first snapshot will match the data stored in it in terms of size; all the following snapshots are incremental.

Following the rsync method, Timeshift creates its snapshots on the system drive by default. You should change this by selecting another internal or external disk with a Linux-style filesystem and sufficient space. This allows the system to be restored even if the system drive has been damaged or accidentally formatted.

LUKS and LVM

Timeshift can handle LUKS encrypted systems, even if you set them up with LVM. The software also supports UEFI, but requires the GRUB2 bootloader. Docker or other containers are not supported, nor are the directories of Ubuntu's Snap package management system, which reside below `/dev/loop` by default. Snapshots must also not be located on network drives or remote devices.

After selecting the desired method, proceed to the *Location* tab, where you specify the storage location (Figure 2). It must have enough free disk space. However, you can only calculate the space you need once you have determined how often you want to take snapshots and how many you want to keep.

A test system that is updated daily occupies about 120GB of disk space after two years of operation. The initial snapshot weighs in at 55GB, and the system contains a total of seven snapshots. A system that is rarely updated or not used very often will probably require less space.

Next, the *Schedule* tab lets you define when to create snapshots and how many of them you want to keep for emergencies. You can choose between hourly, daily, weekly, and monthly snapshots (Figure 3). Timeshift fires up every hour and takes care of the upcoming tasks.

Not Home

The *Users* tab lists the system users. This always includes *root* and one or more

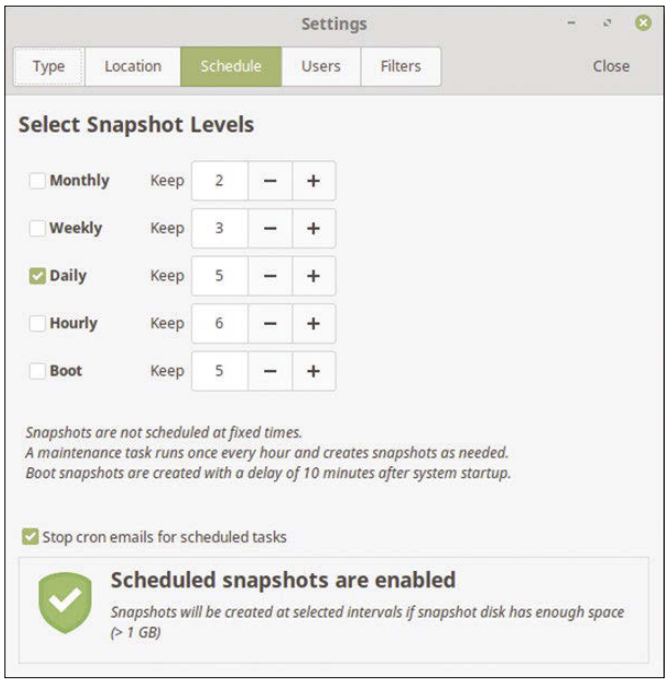


Figure 3: In the *Schedule* tab, you can define when Timeshift creates snapshots and how many it keeps.

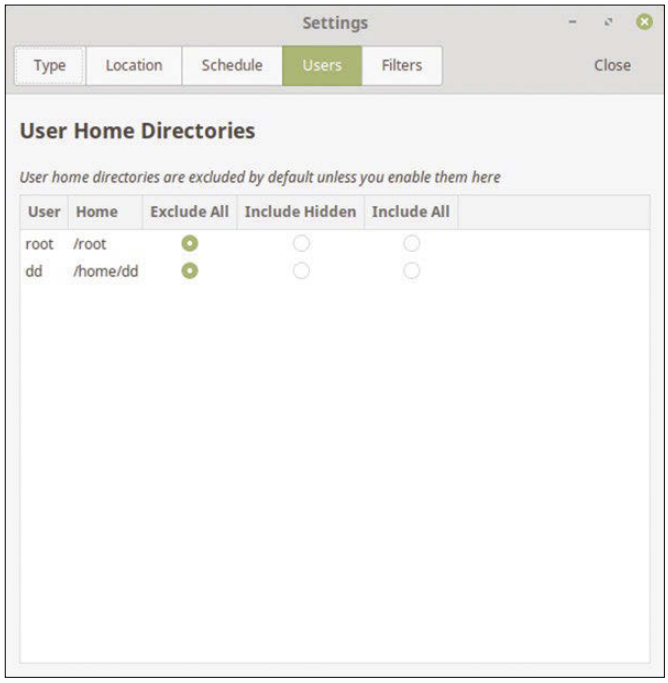


Figure 4: The *Users* tab lists the system users and supports generic inclusion or exclusion of the home directories. Use the filters in the next tab to fine-tune the settings.

unprivileged users (Figure 4). This is where you decide if you want to include home directories in the snapshot, and which ones. The *root* home does not normally need to be saved.

In the case of home directories for unprivileged users, you will want to back up the hidden files. Backing up the entire home directory would be counterproductive. If you had to restore an older snapshot, all the personal data would be overwritten with older variants when restoring the home directory. That's why it's better to use a traditional backup solution for this.

The last tab, *Filters*, influences the size of the snapshots. This is where you can decide to exclude file types, directories, and files from the snapshot. The preset filters result from the *Users* tab's settings. You can now decide on further candidates for exclusion (Figure 5).

The protocols in `/var/log` are generally best excluded, as is `/var/apt`. There are also exclusion candidates among the hidden files in the home directory. In our case, this includes the extensive configuration files of the installed browsers, which are already included in the daily backup. Directories like `dev/`, `proc/`, `sys/`, `media/`, `mnt/`, `tmp/`, and `run/` are automatically excluded. All other exclusions are shown in the *Summary* at the end.

Defining Filters

You have to think through your use of filters. Suppose you have selected the middle option (*Include Hidden*) for home under *Users*. Then Timeshift will only back up hidden files. However, you want to back up the `/home/<User>/foo/` directory, apart from the `bar/` subdirectory. Listing 1 shows the corresponding filter.

Logically speaking, the exclusion line actually belongs at the end. However, this would not work, because filters act on subsequent filters, but do not affect them retroactively. If the exclusion of `bar/` was at the end of the list, it would still be backed up, since the filter `/home/<User>/foo/ *` already includes it, and it cannot be excluded retroactively.

Listing 1: Fine-Tuning Home Directory Filters

```
/home/User/
- /home/User/foo/bar/
+ /home/User/foo/ *
```

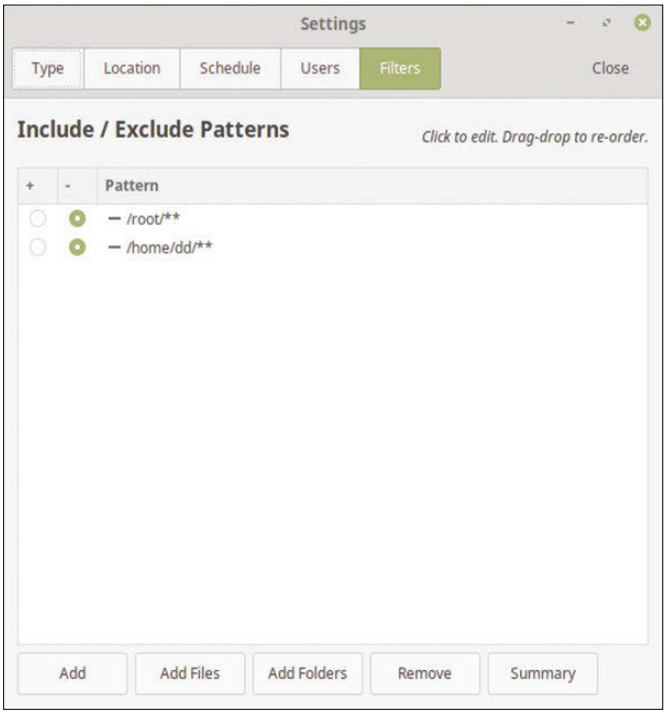


Figure 5: The filters for precise exclusion of file types, folders, and directories have their own logic, which is not described in the documentation.

Simple Btrfs

When using Timeshift with Btrfs, the configuration is even easier, because the snapshots generally end up on the system partition. In addition, a snapshot always covers the entire system, except for Home.

In the tabs *Type*, *Location*, *Schedule*, and *Users*, there is very little to configure. *Type* and *Schedule* are identical to the rsync configuration tabs. As the *Location*, Timeshift specifies the system partition as the default. In *Users*, you can include the home directory, and you can enable quota support [7] in *Btrfs groups*.

If you are wondering what the *Wizard* menu item does in addition to *Settings*, you are not the only one. The only difference besides the *Summary* at the end is the absence of the *Filters* menu item. Thus, the wizard seems to be intended for users who do not want to set filters.

First Tests

After completing the configuration, create an initial manual snapshot as a test in the main window. During the first few days, you will want to check whether Timeshift is working to your satisfaction (Figure 6).

You should also start the tool as root once only in the terminal; this shows you all the available options. For example, `timeline --list` tells you about the method used, the storage location, and the individual snapshots. You should be familiar with this function if you need to perform a restore later without a graphical user interface.

Simply Restore

Timeshift uses a directory structure that matches the filesystem. Likewise, in the event of damage, the folder or file can be restored simply using copy

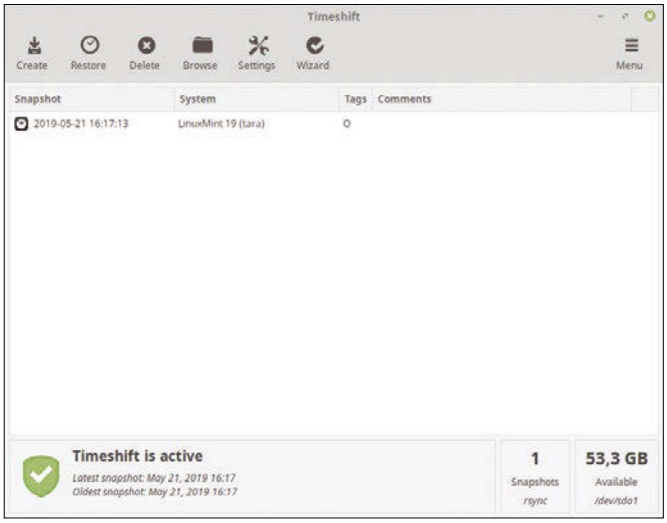


Figure 6: Clicking on a snapshot lets you enter comments.

and paste (Figure 7). When it comes to rolling back the entire system, this is done either from the active system with a subsequent reboot, from another installed distribution, or via a live medium.

If there are several installed distributions, the restore will be successful even from the neighboring system – provided that Timeshift can be installed on it. If the snapshots are not on the system disk, you could theoretically install a different distribution and, if you don't like it, then restore the old system using a live medium. All you have to do is adjust the UUID manually, as the partition has been formatted in the meantime.

On the Running System

To restore from the running system, select the snapshot you want to restore and click *Restore* in the menubar. In rsync, then select the target drive (Figure 8). You can also optionally adjust the bootloader in this case if the subsequent reboot fails with the default set-

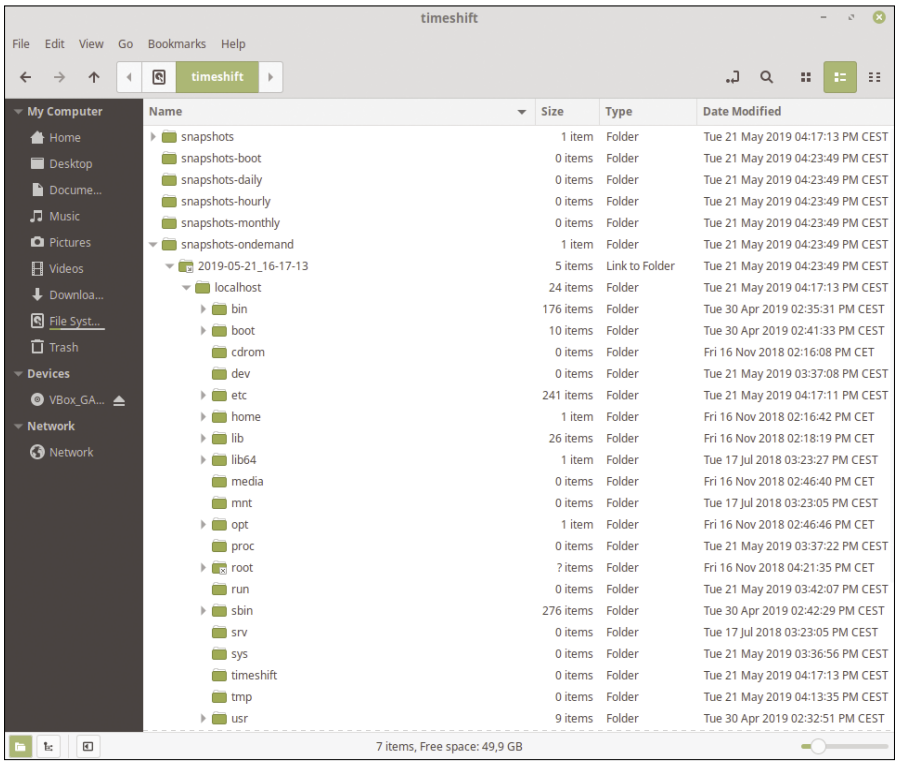


Figure 7: Unlike some backup applications, Timeshift does not use a proprietary file format, but lets you manually restore individual files and directories.

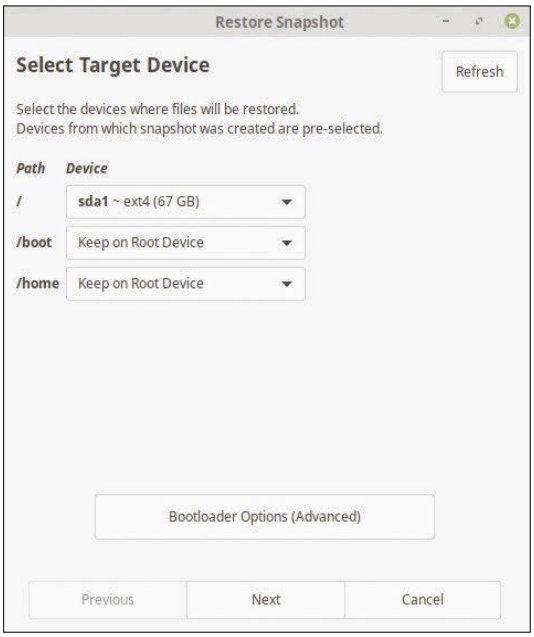


Figure 8: In general, the defaults that Timeshift specifies for the target device are a good fit. Nevertheless, you should take a close look at them.

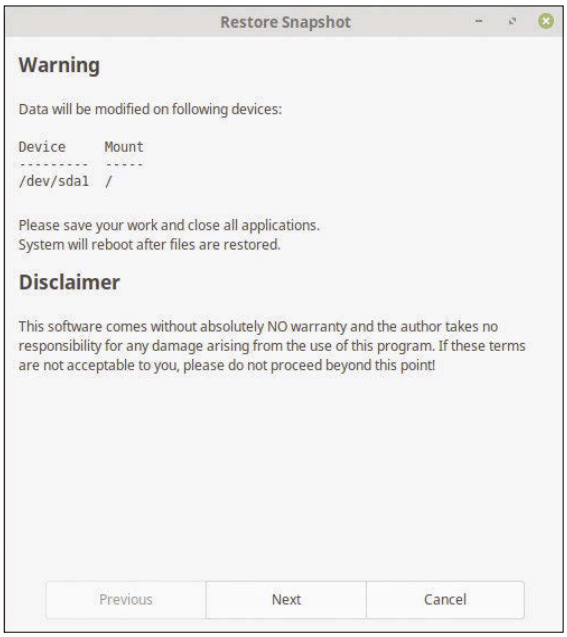


Figure 9: The last window, before the restore starts, shows what happens when you press *Next* (in an example with Btrfs). After that, there's no going back.

ting. For Btrfs, the source and target drives are normally identical.

In the next window, Timeshift performs a test run and then lists all the files it will be restoring. If required, you can also search for packages here. The last window displays the partitions involved once again (Figure 9). Click *Next* here to start the recovery. To be on the safe side, you should close running applications beforehand.

If you start the process on the running system, a terminal opens to let Timeshift restore the files. Do not interrupt the operation in the terminal – this could cause irreparable damage to the installation. After the recovery is complete, the system automatically restarts, reboots the selected snapshot, and then displays the current state of the system.

From the Outside

If you restore the system from a remote system, Timeshift will display the progress graphically instead. When the process is complete, restart the system. If this does not work, test another snapshot or modify the bootloader accordingly. However, in our test, problems of this type did not occur in any case.

If you are using Timeshift with Btrfs, you can continue working immediately after the restore. The desired subvolume then changes to the default at the next restart. Restoring Btrfs snapshots only takes seconds, whereas rsync takes 10 minutes or more, depending on the data volume.

If you test rsync on a Btrfs system first before switching to Btrfs snapshots later, Timeshift will seem to remove the rsync configuration. However, this is only a visual glitch: As soon as you switch back to rsync, you will see that all the snapshots are still available.

Tests Passed

Timeshift passed our tests on various systems with rsync and Btrfs with flying colors. No matter whether default, encrypted, or LVM – recovery succeeded at the first attempt every time. In addition, I have been using Timeshift on a system for about two years without any trouble so far.

Having said this, Timeshift could be better documented. The developer's wiki on GitHub [8] is quite terse. It lacks not only an explanation of the filters, but also instructions for Btrfs on creating the required *Ubuntu subvolume layout*. Not all distributions install Btrfs with the `@` and `@home` subvolumes. For example, openSUSE uses a completely different layout, which is oriented to Snapper and is not suitable for Timeshift. There is also no man page for Timeshift or, at least, no mention of the possibility to use the tool at the command line.

Conclusions and Outlook

It seems that Timeshift is tailored for Linux Mint users. You thus have to glean information about advanced use cases from the Internet. However, this does not detract from the high functionality of Timeshift.

As an extension, integration with the update and boot manager would be desirable, in the way openSUSE does this with Snapper. In addition, the way snapshots are presented could be more meaningful, so that you can view the status of the individual snapshots. Having said this, the integrated comment function does help here. For Btrfs, the grub-btrfs [9] script lets you provide snapshots with the boot manager. ■■■

Info

[1] Timeshift: *https://github.com/teejee2008/timeshift*

[2] CYA: *https://github.com/cleverwise/cya*

[3] Snapper: *http://snapper.io/*

[4] Subvolumes: *https://btrfs.wiki.kernel.org/index.php/Manpage/btrfs-subvolume*

[5] Timeshift installer: *https://github.com/teejee2008/Timeshift/releases*

[6] Hard links: *https://en.wikipedia.org/wiki/Hard_link*

[7] Quotas: *https://btrfs.wiki.kernel.org/index.php/Quota_support*

[8] Wiki: *https://github.com/teejee2008/timeshift/wiki*

[9] grub-btrfs: *https://github.com/Antynea/grub-btrfs/blob/master/41_snapshots-btrfs*

Editing PDF Structure with QPDF

Reconstruction

Use QPDF to easily make structural changes to your PDFs, including reorganizing pages, creating watermarks, setting encryption options, and changing permissions. *By Bruce Byfield*

QPDF [1] is a structural editor for PDF files. This description places it in a very specific niche. In its usual output method, it does not edit the content of PDF files – to the extent that editing content is possible, opening a PDF in LibreOffice is generally the easiest way to work. Nor does QPDF import PDFs to different formats – the repositories of major distributions like Debian are full of scripts for that, like `pdf2htmlEX` and `pdf2svg`. However, if you need to change how a PDF is put together, QPDF is a toolkit that is both comprehensive and more convenient than the assorted scripts that only perform a single function. In fact, by adding options, you can make an entire series of structural edits with a single command. QPDF is especially handy if you no longer have the file from which a PDF was generated and are therefore unable to make a new one with different settings.

QPDF is available in many distributions. If it is not in your distribution's repository, you can download the source code from the project site and build it with the usual trio of commands: `configure`, `make`, and `make install`. The syntax, too, is simple:

```
qpdf OPTIONS ORIGINAL-FILE OUTPUT-FILE
```

The output file is not needed for some options, such as those for information. Commands complete without any confirmation except the return to the prompt.

The original file is kept untouched when the command is run, so any errors will not leave you with a corrupted file. Detailed help is available from the command `qpdf --help` rather than the man page. QPDF's options are numerous, but the most generally used options can be divided into four main categories: those for general operations, information, page selection, and encryption. In addition, for the adventurous, QPDF can create a file in QDF mode, which will create an output file that can be opened in a text editor.

Options for General Operations

These options determine how QPDF runs, and most can be used alongside other options. They include several unusual features. For example, QPDF's own completion tool can be enabled for either the Bash or Zsh shells with the command

```
eval $(qpdf --completion-bash)
```

or

```
eval $(qpdf --completion-zsh)
```

If QPDF is not in your path, you will need to give its complete path in order to use completion.

Similarly, if a PDF is protected by a password, in order for QPDF to manipulate it, you will need to give the password with the option `--password=PASSWORD`. Without the password, even the information options will not function. If, as often happens, the original PDF has two passwords, one for viewing and one for editing, you will have to enter options for both passwords unless they are the same.

If you want a PDF that displays quickly on the web, select `--linearization`. This display makes such changes as reducing the resolution of images so that they load

Author

Bruce Byfield is a computer journalist and a freelance writer and editor specializing in free and open source software. In addition to his writing projects, he also teaches live and e-learning courses. In his spare time, Bruce writes about Northwest coast art (*http://brucebyfield.wordpress.com*). He is also co-founder of Prentice Pieces, a blog about writing and fantasy at *https://prenticepieces.com/*.

Table 1: Information Options

`--show-encryption`	Quickly show encryption settings
`--check-linearization`	Check file integrity and linearization status
`--show-linearization`	Check and show all linearization data
`--show-xref`	Show the contents of the cross-reference table
`--show-npages`	Print the *n* number of pages in the file
`--show-pages`	Gives info for each page
`--with-images`	Shows the object/generation number for each page plus object IDs for images on each page
`--check`	Check file structure plus encryption and linearization

faster. Documents that are all text will benefit minimally from linearization.

Information Options

QPDF's options for retrieving information about a PDF are useful for troubleshooting or for using the QPDF library in automated test suites (Figure 1). The `--check` option gives a quick summary of the file structure, encryption, and linearization. The function of other information options is evident from their names (Table 1).

Page Selection

QPDF can manipulate the pages shown in the output file. The `--pages` option must be used after the basic command, and the page range after the original file. Individual page numbers can be separated by commas, or a range of pages by a dash. Individual pages and ranges can be listed together, so that `3,5,11-14` would be a valid listing of pages. Pages are printed in the order that they are listed, so `11-14,3,5` prints pages 11-14 first in the output file. Other values include `z-1` to print in reverse order starting from the last page, and `r2-r1` prints the last two pages, while `r1-r2` prints the last two pages in reverse order.

Output files can also be created that use multiple source PDFs. When using multiple files, place the `--pages=PAGES` option after the name of each source file, rather than after the basic command. After the basic command, you can add `--collate` so that the output file begins with the first page or range for the first file in the command, followed by the first page or range for the second file, then the second page or range for the first file, and so on. For example:

```
qpdf file first.pdf pages=1-4 z
  second.pdf pages=r4-r1 merged.pdf
```

Still another way to select pages is to define a particular page as either an overlay or an underlay, in effect creating a watermark. Whether you use an overlay or underlay is a matter of choice, usually determined by what you want to be displayed clearly. `--overlay` or `--underlay` is added after the basic command, and the first page specified for the first file becomes the overlay or underlay for all the pages specified in the second file. Alternatively, where the overlay or underlay file is applied can be specified by adding `--to=PAGES` and `--from=PAGES` after it.

Encryption Options

Contrary to some passing references on the web, QPDF's main purpose is not to crack password protected PDFs. It may enable cracking with the use of `--password-is-hex-key`, which interprets the password as a hexadecimal-encoded key value. However, the lack of a viewer to support this mode means that the option is only possibly useful, allowing the

```
bb@nanday:~$ qpdf --show-encryption --password=test ./chapter15.pdf
R = 3
P = -3096
User password = test
extract for accessibility: allowed
extract for any purpose: not allowed
print low resolution: not allowed
print high resolution: not allowed
modify document assembly: not allowed
modify forms: allowed
modify annotations: allowed
modify other: allowed
modify anything: not allowed
```

Figure 1: Information options like `--show-encryption` display detailed information about a file's structure.

Table 2: PDF Permission Settings

Key Length = 40	
`--print=[yn]`	Allows printing
`--extract=[yn]`	Allows text or image extraction
`--annotate=[yn]`	Allows comments and form fill-in and signing
Key Length = 128	
`--accessibility=[yn]`	Allows accessibility to visually impaired
`--extract= [yn]`	Allows text or image extraction
`--assemble=[yn]`	Allows rotation and reordering of pages
`--annotate=[yn]`	Allows comments, form fill-in, and signing
`--form=[yn]`	Whether filling form fields is allowed
`--modify-other=[yn]`	Allows all document editing except those controlled separately by `--assemble`, `--annotate`, and `--form`
`--print=print-opt[full, low, none]`	Controls printing resolution or whether it is allowed
`--modify=[all, annotate, form, assembly, none]`	Controls modify access
Key Length = 256	
`--use-aes=[yn]`	Uses AES encryption instead of RC4 encryption

output file to be viewed with forensic tools – although the manual is careful not to specify which tools.

However, if you have the password for a PDF, you can edit its encryption options. If you have the password, the encryption key can be viewed with `--show-encryption-key`. You can also remove all encryption with the option `--decrypt`.

In addition, you can edit a PDF's built-in permissions. The necessary snippet of the command structure is:

```
--encrypt USER-PASSWORD OWNER-PASSWORD
  KEY-LENGTH PERMISSIONS
```

`USER-PASSWORD` and `OWNER-PASSWORD` refer to the passwords added when the PDF is created. And, despite its name, `KEY-LENGTH` does not refer to the public key used in an application like GPG, but to groups of settings that are part of the PDF standard. These groups are designated by lengths of 40, 128, and 256. Each group has its own settings, as shown in Table 2.

The lengths of 40 and 128 give the same permissions as are available using CommonPDF file creators. Be aware that the built-in encryption is notoriously weak and can be bypassed by a number of applications that are available for the download. If you are seriously concerned about security that goes beyond providing an obstacle for unsophisticated users, be sure to include a key length of 256, which provides more serious encryption. My recommendation is to use it alongside the 128 key length, which provides comprehensive options. If no key length is specified, the output file is fully editable.

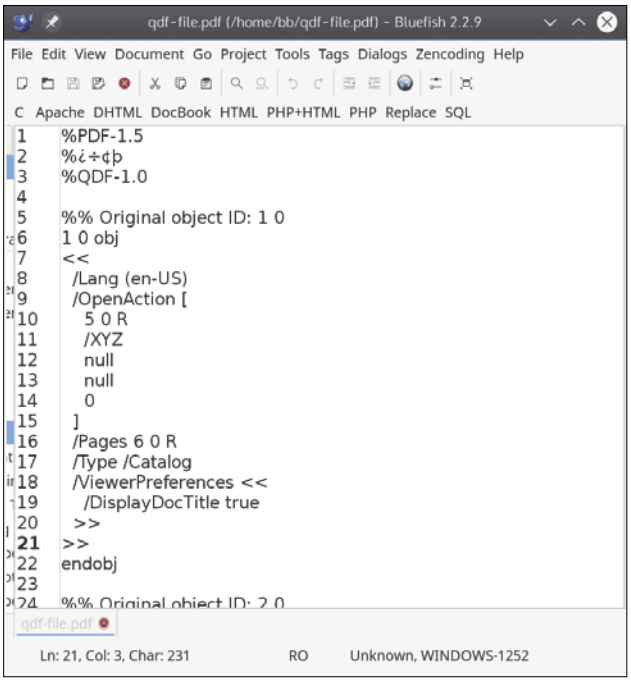


Figure 2: QDF mode allows you to view and edit the structure of a PDF in a text file.

QDF Mode

Generally, the easiest way to edit a PDF file is to open it in LibreOfice Writer. Writer is especially ideal if you are using a hybrid PDF – that is, one created in Writer that also includes a copy of the file in OpenDocument Format, LibreOffice's default format. At the cost of a file twice as large as an ordinary PDF, a hybrid provides a fully editable file that also updates the accompanying PDF file when saved. But if you do not have a hybrid file, then a PDF can only be edited line by line in Writer and other editors, and new lines are only practical in blank space.

QDF mode is a format that displays like any other PDF, but it can be edited in a regular text editor, as long as there is no password protection. If a file does have a password, it can be viewed, but not edited. The catch is that the format displays all objects in numerical order. This format takes some practice to read. Content is easy to find, but objects like images need to be carefully edited – for instance, if you remove an image, you need to update every other image, or else the output file will not build or display properly (Figure 2).

To create a file in QDF mode, simply add the `--qdf` option. If you run into trouble with a QDF mode file, try using `--fix-qdf`. This option tries to repair everything from object streams to cross-reference tables, although the repairs may not be entirely what you hoped. Also, be aware that QDF mode is incompatible with linearization, which essentially gives the same view of the file.

Other Options

This article only covers the uses of QPDF that might be useful to end users. The QPDF manual [2] is current and contains almost as much information again for developers. As well as options for testing and debugging, QPDF has options for how it handles Unicode passwords and file names and for use in C++, C, JavaScript, and Python.

However, you do not need to be a developer to find QPDF useful. Although you will probably want to work with the latest version of the manual open, QPDF is a comprehensive toolkit and can replace several common scripts under one command. If you regularly edit PDFs, QPDF is in many ways an essential application. ■■■

Info

[1] QPDF: *http://qpdf.sourceforge.net/*
[2] PDF Manual: *http://qpdf.sourceforge.net/files/qpdf-manual.html*

Linux Magazine **is your guide to the world of Linux. Look inside for advanced technical information you won't find anywhere else!**

Expand your Linux skills with:

- In-depth articles on trending topics, including Bitcoin, ransomware, cloud computing, and more!
- How-tos and tutorials on useful tools that will save you time and protect your data
- Troubleshooting and optimization tips
- Insightful news on crucial developments in the world of open source
- Cool projects for Raspberry Pi, Arduino, and other maker-board systems

If you want to go farther and do more with Linux, subscribe today and never miss another issue!

Subscribe now!

shop.linuxnewmedia.com/subs

GET IT NOW!
FAST DELIVERY WITH OUR PDF EDITION

Making PDFs More Secure in LibreOffice Writer

LibreOffice Writer PDF Security

Depending on your needs, LibreOffice Writer offers varying degrees of security for PDFs. *By Bruce Byfield*

PDFs date from a less security-conscious era than our own. However, over the years, the PDF format has added security features. Today, if you need security, you have two choices: passwords and permissions for casual security of digital certificates or GNU Privacy Guard (GPG) keys for serious encryption. Both are available from tabs on LibreOffice's PDF Options window when exporting to PDF.

Passwords and Permissions

PDFs have their own system of passwords and permissions, which are available from *File* | *Export As* | *Export As PDF…* | *PDF Options* | *Security* (Figure 1). To set them up, begin by entering a password to open the exported file, and a second one to alter the permissions (in other words, how the files can be used). After the second password is entered, three kinds of permissions are available: Printing, Changes, and Contents. Together, options can be as strict as allowing a user only to view the file, as loose as allowing any user to alter the file at will, or something in-between.

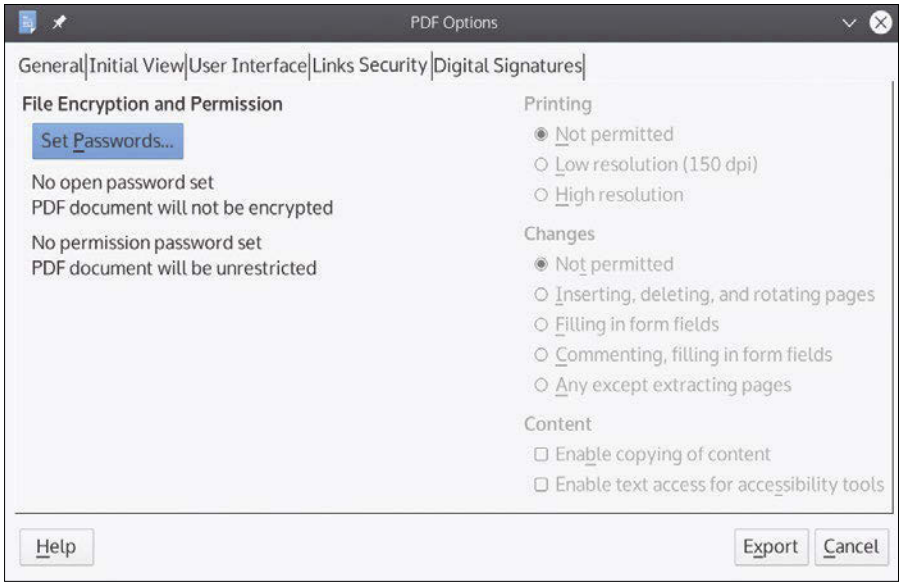


Figure 1: The *Security* tab in the PDF Options window offers a casual grade of security with a variety of options.

Dating from a less security-conscious era, the reasons for these restrictions may seem arbitrary today. For example, why restrict printing to 150dpi, a resolution that is low, but still allows printed pages to be scanned and enhanced? The inability to print in high resolution seems trivial compared to the ability to print at all. Similarly, the combinations of allowable changes seem inconvenient. For instance, while you may not want users to fill in forms, why is there no way to allow comments on forms alone?

In fact, before setting permissions on a PDF file, you might ask if doing so is worth the effort. Over the years, PDFs have been notorious for security weaknesses; unsurprisingly, numerous ways to bypass a password are available. On Windows, proprietary applications like PDFelement or iSumsoft PDF Password Refixer are available for downloading. On Linux, PDFCrack does dictionary-supported brute force attacks to open a

password-protected PDF. Easier still, Ghostscript can bypass the password:

```
gs -q -dNOPAUSE -dBATCH ➥
  -sDEVICE=pdfwrite➥
  -sOutputFile=[unencrypted.pdf] ➥
  -c .setpdfwrite➥
  -f [encrypted.pdf]
```

And these are just the available tools I found in a five minute search. Undoubtedly, other tools are available, no matter what operating system you use. Moreover, once the file is opened, of course, all the carefully set permissions can be altered without any problem.

PDF permissions can be classified as a subset of security through obscurity – the practice of not mentioning security risks and hoping no one notices, which is widely condemned by security experts. Better yet, PDF permissions could be described as security through ignorance, working only so long as users have no idea how wide-open they are to anyone who can do a web search. All they are really useful for is controlling unsophisticated users' behavior. Anyone who really wants to bypass the password and the permissions will find a way to do so.

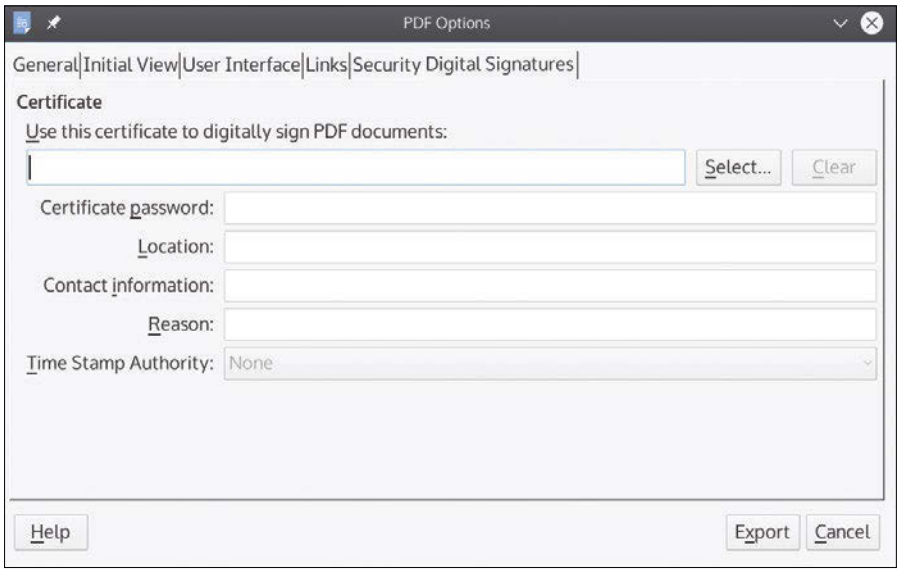


Figure 2: The *Digital Signatures* tab of the PDF Options window is the place to add advanced encryption.

Digital Certificates and GPG

Two secure alternatives to permissions are available from *File* | *Export As* | *Export As PDF…* | *PDF Options* | *Digital Signatures* (Figure 2). These alternatives do not allow you to fine-tune how a PDF file can be used or edited, but they do provide stronger security than permissions. In addition, they guarantee that a sent file is actually from you.

Figure 3: Let's Encrypt provides free certificates.

These alternatives are to obtain a digital certificate from a certificate authority or to generate personal keys yourself. Certificates and keys are simply alternative names for the same tool: a passphrase-protected system of encryption. They both consist of a private certificate or key and a public one that the recipient must be sent in order to read the files you send. As the originator, you can use the certificate or key to read your own encrypted files.

Digital certificates are probably best-known in corporate circles. They require interacting with a certificate authority, whose reputation presumably adds weight to the authenticity of the certificate you receive from it. The exact details of using a certificate vary with the certificate authority, your browser, and your version of LibreOffice, but here is a summary of the general steps:

1. Sign into a free-cost certificate authority site like the Linux Foundation's Let's Encrypt [1] (Figure 3) and follow the steps to generate a certificate.
2. Locate the certificate in your web browser's preferences or set up and make it available for files.
3. Depending on the version of LibreOffice, you may need to make Writer aware of the certificate using *File* | *Digital Signature* | *Digital Signatures…*, and then restart Writer.
4. Add the certificate to the PDF file using *File* | *Export As* | *Export As PDF …* | *PDF Options* | *Digital Signatures*, and fill out the required information. Alternatively, use *File* | *Digital Signature* to add a certificate to an already generated PDF.

However, using a digital signature can be an involved process. Despite the name, in recent versions of Writer, the Digital Signatures tab also recognizes keys created using a variant of Pretty Good Privacy (PGP), such as GPG. By using GPG, in effect, you sacrifice whatever reassurances using a certificate authority may have for the convenience of doing everything yourself (Figure 4).

If you already used GPG, the process of adding a key to a PDF file is similar to any other use. To generate keys with GPG, run the command:

```
gpg --full-generate-key
```

GPG takes you through the five steps in creating keys: adding your name and email, creating a passphrase, choosing the algorithm, setting the key size, and assigning an expiration date. If you are unsure about some of the technical choices, you can always accept the defaults. As a last step, you should create a revocation certificate, which allows you to make the new key invalid if it is ever compromised, with the command:

```
gpg --armor --output revoke.asc ➥
  --gen-revoke PUBLIC KEY ID
```

```
bb@nanday:~$ gpg --full-generate-key
gpg (GnuPG) 2.1.18; Copyright (C) 2017 Free Software Foundation, Inc.
This is free software: you are free to change and redistribute it.
There is NO WARRANTY, to the extent permitted by law.

Please select what kind of key you want:
   (1) RSA and RSA (default)
   (2) DSA and Elgamal
   (3) DSA (sign only)
   (4) RSA (sign only)
Your selection?
```

Figure 4: The start of generating a key in GPG.

The key can be selected and details added on the *Security* tab of the PDF Options window. Once the key is created, you can send out the public key with

```
gpg --output YOURNAME.gpg ➥
  --export KEY-EMAIL
```

or as a protected plain text file with the format:

```
gpg --armor --output YOURNAME.gpg ➥
  --export KEY-EMAIL
```

Again, the key can be selected and details added on the *Security* tab of the PDF Options window. Recipients of the file can verify it is from you with:

```
gpg --fingerprint KEY-EMAIL
```

Then create a decrypted copy of the file with:

```
gpg --decrypt ENCRYPTED-FILE
```

The file's text appears in the command line, and an unencrypted version of the file in the same directory as the encrypted file.

Whether you choose a certificate or a GPG key depends on your preferences and convenience. From a security viewpoint, one is generally as secure as another, except that different certificate authorities may default to different levels of encryption.

Choosing the Security Method

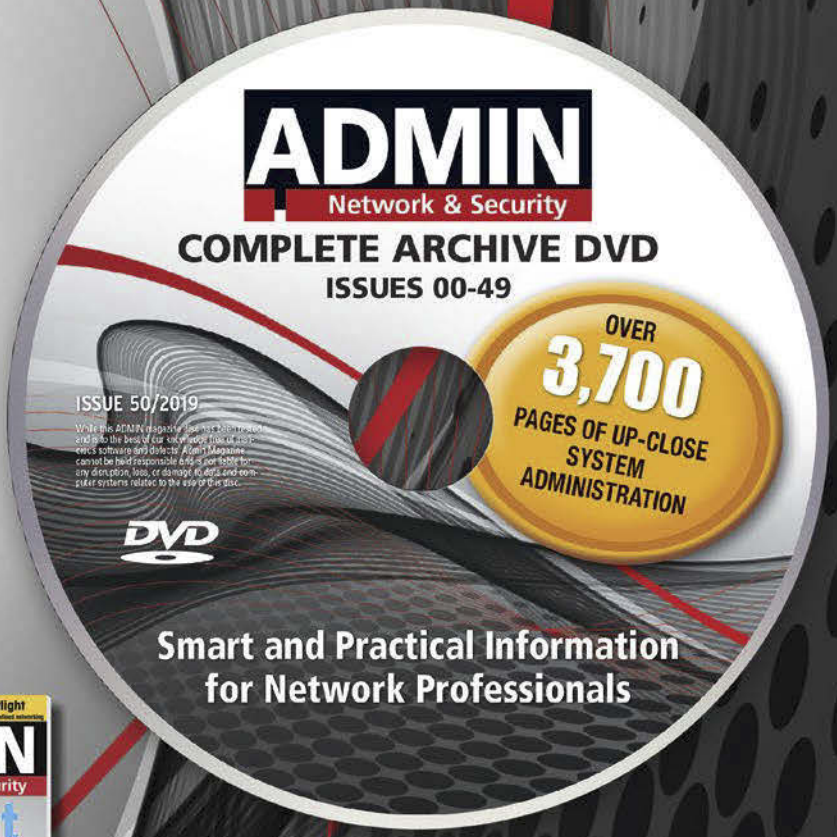
Neither passwords and permissions nor certificates and keys are entirely satisfactory on their own. Passwords and permissions have the advantage of controlling access in particular ways, but as security features, they are so weak that in many cases they are pointless.

By contrast, certificates and keys have strong security, but their access is all or nothing – you either have access to the PDF, or you don't. However, their lack of choice is probably preferable in most cases to the lack of acceptable security with passwords and permissions. ■■■

Info

[1] Let's Encrypt: *https://letsencrypt.org/*

9 Years of ADMIN on One DVD



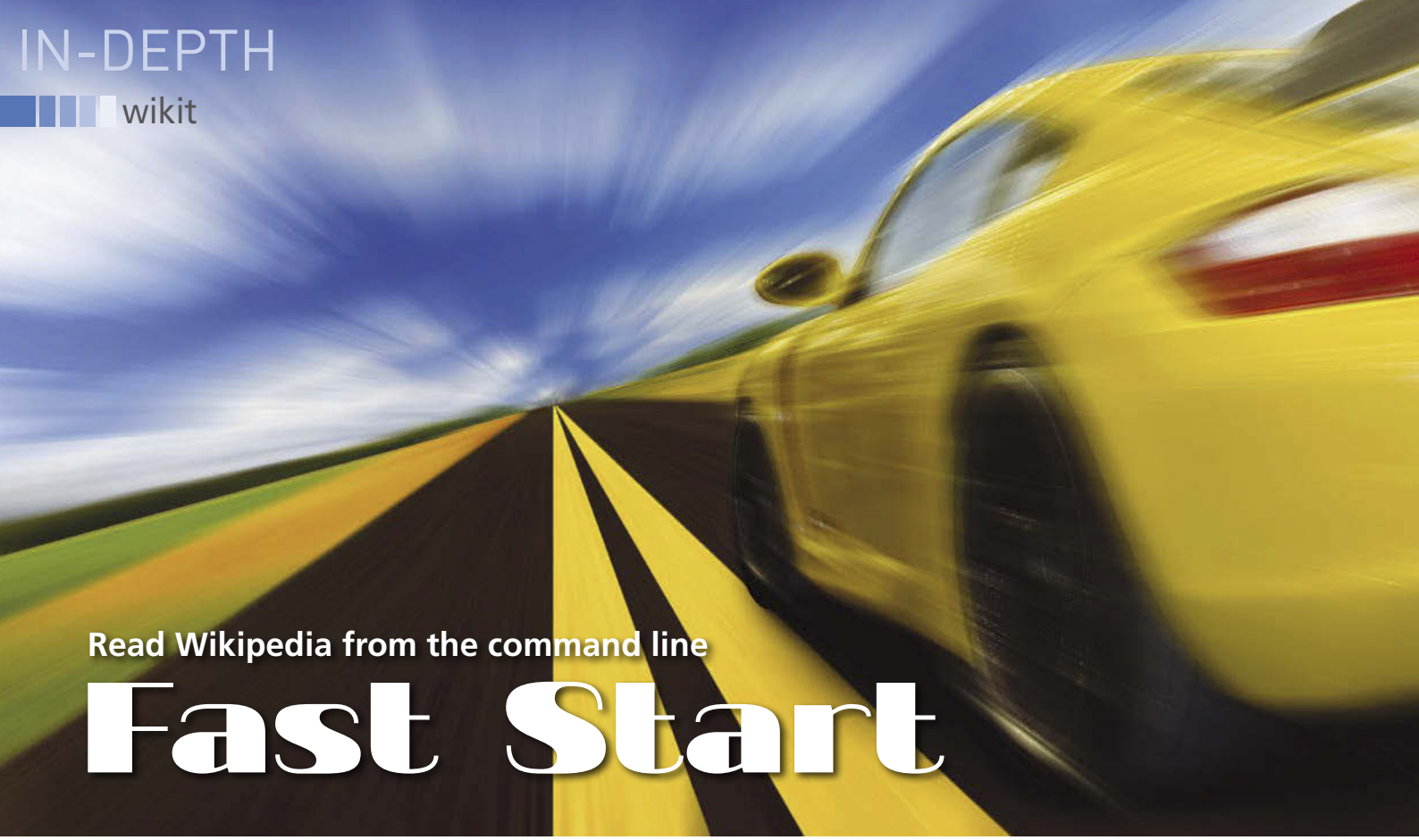
This searchable DVD gives you 50 issues of ADMIN, the #1 source for:

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

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

ORDER NOW!

shop.linuxnewmedia.com



Read Wikipedia from the command line

Fast Start

Save time and mouse clicks by using wikit to search and read Wikipedia entries at the command line. *By Ferdinand Thommes*

If, like me, you are a big fan of the command line, you'll be happy to hear about any script that keeps your hands on the keyboard while keeping you out of a web browser. This is how I felt when I discovered `wikit` [1].

Accessing and searching Wikipedia is part of everyday life just like coffee with breakfast. However, I rarely read entire articles. Instead, it's more about getting a quick introduction to a topic or retrieving a few facts. `wikit` does this by displaying a summary of various Wikipedia articles at the command line. The summary includes the portion of a Wikipedia page that precedes the table of contents.

`wikit` includes a few options, which are quickly memorized, making deployment easy. One option lets you call articles in different languages; another option lets you call the complete article in either a GUI or text-based web browser. In addition to specifying the desired language when calling up an article, you can also set a fixed language or influence the line length.

Installation with npm

Installing `wikit` requires the Node.js [2] JavaScript platform and its package manager `npm` (as is often the case with applications based on web technology). You will quickly have them installed using your distribution's package manager.

For Debian and its derivatives, installing Node.js just means typing:

```
$ sudo apt install nodejs npm
```

For Fedora, use:

```
$ sudo dnf install nodejs npm
```

With RHEL or CentOS, you first integrate the EPEL repository using

```
$ sudo yum install epel-release
```

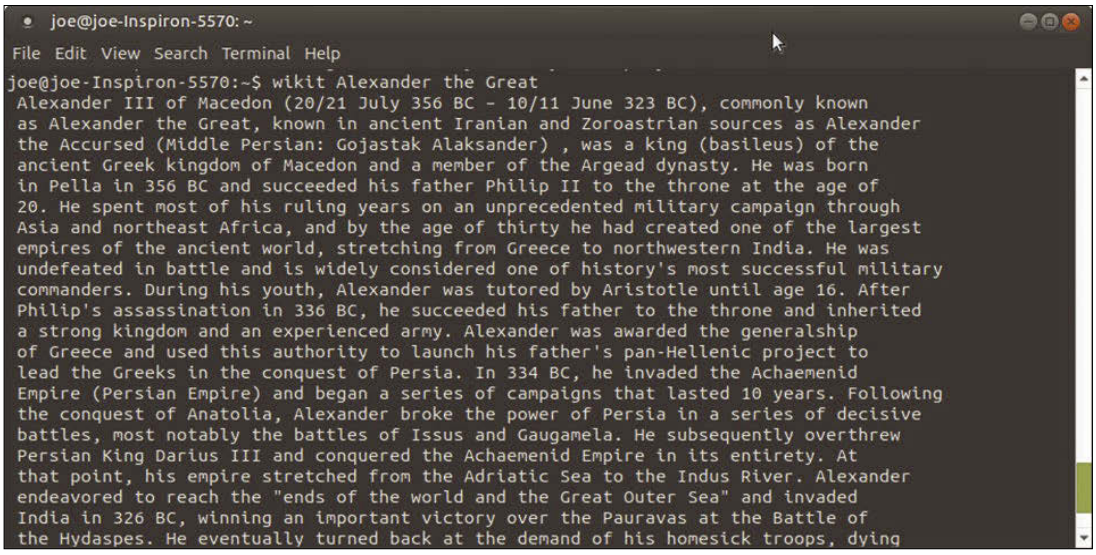


Figure 1: The `wikit` utility makes it easy to access Wikipedia from the command line.

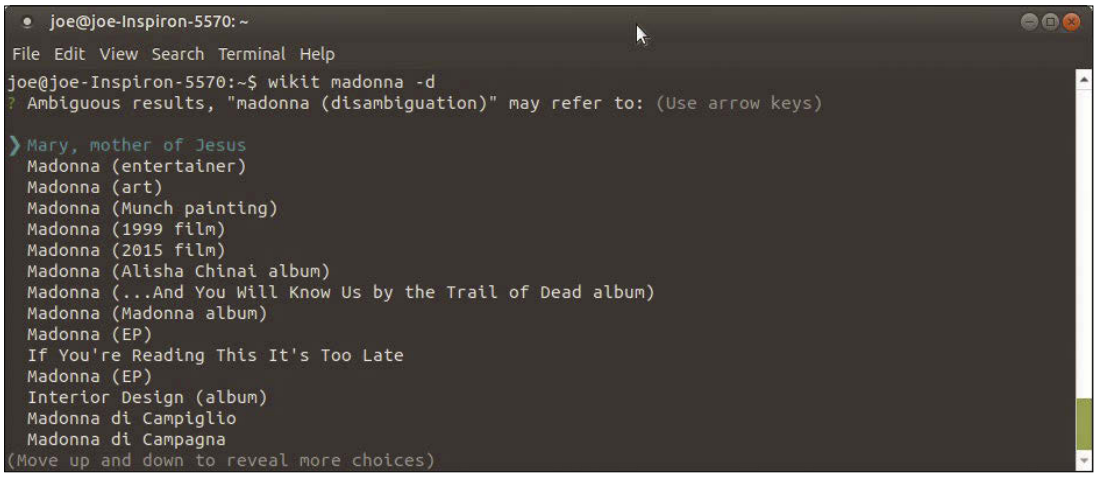


Figure 2: If a search term branches to several articles, `wikit` lists them all so you can browse the list.

and then install using:

```
$ sudo yum install nodejs npm
```

And, with Arch Linux, the command is:

```
$ sudo pacman -S nodejs npm
```

If your distribution is too old for Node.js, install the current Node.js from the project's GitHub site [3] for security purposes. You may also need to update since `npm` could refuse to work with older versions. Currently, the long-term support version 8 or the newer versions 10 and 11 are available.

Short and Sweet

Next, you install `wikit` via `npm` with the `-g` parameter globally for all users:

```
$ sudo npm install wikit -g
```

The software is now ready for use. Type `wikit` followed by the desired Wikipedia entry:

```
wikit Alexander the Great
```

If no parameters are specified, the English version of the article appears (Figure 1). If you'd like to specify a different language (if translations are available), use the `-l` option as follows:

```
wikit Alexander the Great -l de
```

If the search term is a proper name, as in the example, then you must type it exactly.

If several Wikipedia articles match your search term, `wikit` will list all the possible matches so that you can choose the desired article (Figure 2).

In the Browser

If you need more than the article summary, you can call the entire article in a standard browser (Figure 3) with:

```
wikit Mount Olympus -b
```

You can also specify the browser with:

```
wikit Mount Olympus --browser firefox
```

The browser option lets you call Wikipedia using either a GUI or a text-based web browser, such as w3m, Links, or Lynx.

Figure 3: If the summary is not sufficient, open the entire article in a browser (including text-based browsers).

If necessary, you can adjust the output to match the terminal's line length, which is 80 characters. You may reduce this to a minimum of 15 characters or increase it by more than 80 characters. For example,

```
wikit Alexander the
  Great --line 40
```

halves the output's line length.

Conclusions

If you do most of your work at the command line, `wikit` is a useful tool that saves mouse clicks and time. It works intuitively and gives you a quick overview of a topic, without the distraction of images and formatting. The GitHub page only explains a few options in detail, but if you do experience any problems with `wikit`, you can contact the developer [1]. ■■■

Info

[1] wikit: *https://github.com/KorySchneider/wikit*

[2] Node.js: *https://en.wikipedia.org/wiki/Node.js*

[3] Node.js on GitHub: *https://github.com/nodesource/distributions/blob/master/README.md#deb*

The sys admin's daily grind: Log2Ram and frontail

Just for the Record

If you run 25 Raspberry Pis at home, and an equal number of other IP devices, you might also think like Charly does when it comes to log management. The result is atomic technology and a logfile disk that is not permanently overloaded. *By Charly Kühnast*

From time to time, I use `nmap -sP 10.0.0.1-254` to check how many IP devices are online in my home network. There are now more than 50, half of them Raspberry Pis. The need for a central syslog server is slowly growing. An old miniature PC with an Intel Atom, which I retrofitted with an SSD, is the designated candidate for this permanent task. The syslog server comes courtesy of the standard rsyslogd. In its configuration file (`/etc/rsyslog.conf`), the following lines ensure that the server can receive syslog data from other hosts via UDP and TCP:

```
$ModLoad imudp
$UDPServerRun 514
$ModLoad imtcp
$InputTCPServerRun 514
```

On the other machines, I added an entry of `*.* @10.0.0.254` to `rsyslog.conf` so that they all send their log data to the server on 10.0.0.254.

However, the incoming syslog messages generate huge numbers of writes, and I'm worried about the SSD service life. Enter Log2Ram [1] stage left. It creates a RAM disk on `/var/log` to which the central rsyslogd writes all the incoming data. Once an hour, the collected data are written to disk in one fell swoop.

Need to Talk

I installed Log2Ram by running the following command line on the log server:

```
git clone https://github.com/
  azlux/log2ram
```

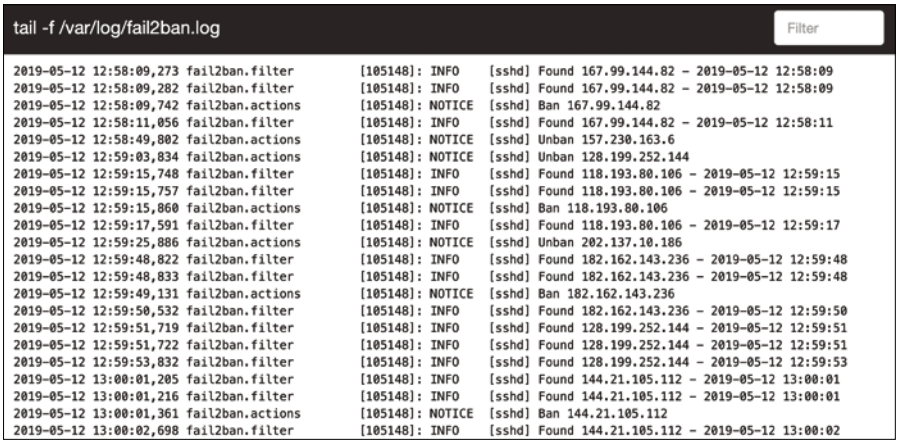


Figure 1: `frontail` **opens a viewing window into the log bucket.**

I then changed to the directory created in the last step and executed the `install.sh` script. At first the installation failed because the Mailutils package was missing, and Log2Ram insists on the ability to mail to the admin in case of problems.

Also the size of the RAM disk, 40MB by default, was too small for my setup, but this can be adapted with a manual edit of the configuration file.

Now I just have one more wish: I don't want to be restricted to viewing the logs with `tail -f` on the log server console, instead I also want to inject them into a web page, just in case I feel the urge to inspect the files while I'm on the road. A small tool by the name of `frontail` [2] helps me do exactly this. It is based on Node.js, so you need to install the `npm` installer. You then install `frontail` and launch it like this:

```
npm i frontail -g
frontail /var/log/syslog
```

This starts a small web server on port 9001. Now, when I open the page in a web browser, I'm welcomed by the syslog (Figure 1). With just a little manual intervention, I can enjoy the view and an SSD that should survive for a couple of years. ■■■

Info

[1] Log2Ram: *https://github.com/azlux/log2ram*
[2] frontail: *https://github.com/mthenw/frontail/blob/master/README.md*

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.

Lead Image © Andrea De Martin, 123RF.com

Using ARP for Network Recon

Network Sleuth

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

The most obvious thing system administrators and hackers have in common is the need for network reconnaissance (recon). In both cases, such recon needs to be carried out as quickly and with as little impact to users as possible. One such recon technique involves finding every network-connected device on a subnet. You might think that this is an easy task, but it isn't. The first tool everyone thinks of is ping. However, ping can be, and usually is, blocked from use against important network-connected devices such as routers, firewalls, switches, intrusion detection appliances, intrusion prevention appliances, servers, and even workstations. Ping is not an effective tool for finding every network-connected device. Instead, an effective solution is to use the Address Resolution Protocol (ARP). ARP maps IP addresses to MAC (hardware) addresses.

ARP is effective in finding all network-connected devices, because you cannot block ARP. ARP must be allowed on a network for proper host-to-host communications. It is this feature (or flaw) that makes ARP a valuable reconnaissance tool. Fortunately, some clever programmers developed an easy-to-use, command-line tool, called ARP Scan (`arp-scan`), that makes quick work of this type of reconnaissance. The only limitation of using ARP in this manner is that its use is confined to a local subnet. In other words, you can scan all devices on the 192.168.1.0/24 subnet, but you cannot scan the 192.168.2.0/24 network unless you scan from one of those 192.168.2.xxx addresses. To put it simply: ARP is non-routable.

ARP Provides a Wealth of Information

Although `arp-scan` is a very versatile tool, my use of it is usually limited to the following five general usage scenarios:

- Discovery of all IPv4 network-connected devices.
- Quickly identify and map IP addresses to MAC addresses.
- Find duplicate IP addresses.
- Isolate and locate rogue devices.
- Identify devices by NIC vendor.

`arp-scan` can scan every address in a /22 (1,024 hosts) network and generate a re-

Listing 1: Partial Output of ARP Subnet Scan

```
192.168.1.1   3c:0e:23:xx:xx:xx  Cisco Systems, Inc
192.168.1.2   b8:27:eb:xx:xx:xx  Raspberry Pi Foundation
192.168.1.3   0c:c4:7a:xx:xx:xx  Super Micro Computer, Inc.
192.168.1.4   00:15:5d:xx:xx:xx  Microsoft Corporation
192.168.1.5   00:04:ac:xx:xx:xx  IBM Corp
192.168.1.7   00:15:5d:xx:xx:xx  Microsoft Corporation
192.168.1.11  5c:f9:dd:xx:xx:xx  Dell Inc.
192.168.1.12  00:21:9b:xx:xx:xx  Dell Inc.
192.168.1.18  00:21:9b:xx:xx:xx  Dell Inc.
192.168.1.20  00:1e:2a:xx:xx:xx  NETGEAR
192.168.1.26  00:15:17:xx:xx:xx  Intel Corporate
```

port in under five seconds. Listing 1 shows the partial output of a typical ARP subnet scan, and the results from running `arp-scan` are displayed in columns: IP address, MAC address, and vendor. (I have obfuscated my actual MAC addresses with `xx:xx:xx`).

As you can see from the sample `arp-scan` output in Listing 1, it provides a huge amount of information very quickly. Using this information, you can then perform a DNS lookup scan of all "live" IP addresses giving you enough information to identify every host on a subnet by name, IP address, MAC address, and NIC vendor.

As a system administrator, you can find rogue devices that users or outsiders have connected to your network. By having the MAC address, you can locate the switch port they are connected to and physically locate the device. `arp-scan` works equally well on wireless networks. It will be more difficult to locate a rogue device on a wireless network, but at least you have a good starting point from which to work. As a hacker, you now have all the information you need to spoof IP and MAC addresses and to exploit any vendor-related vulnerabilities. You also have a list of "live" IP addresses so that you could assign a free one to your rogue device, bypassing any required authentication protocol to obtain one via DHCP. And if your target uses MAC filtering, you can easily spoof one from your list.

What other information can you infer from this list? Using the vendor information, you can take a pretty good guess as to the device's operating system. This information is handy because a port scan can set off alarms on a well-monitored network. For system administrators who've obtained permission to do so, port scanning is not an issue.

At the end of each `arp-scan` report, the program provides valuable statistics to the user as shown below.

```
94 packets received by filter,
0 packets dropped by kernel
Ending arp-scan 1.9.2:
1024 hosts scanned in 4.759 seconds
(215.17 hosts/sec). 94 responded
```

This information is valuable because it informs the user of how many devices are alive and connected to this particular subnet. Ninety-four hosts provide a lot of fodder for a hacker looking for vulnerabilities. And remember that hackers don't just look for vulnerabilities in Windows, Linux, or macOS-based systems. An ARP sweep provides information about printers, network-attached storage devices, phones, postage meters, and any other network-connected device that might provide a vulnerability or an easy way to establish a presence on your network.

Installing arp-scan

Because `arp-scan` is a command-line only tool, you must either compile it from source or install it as a package using your distribution's package manager. `arp-scan` has two dependencies whether you're installing from source or using a package manager. You will need both `automake` and `autoconf`.

If you're a purist or if your distribution doesn't provide `arp-scan` as a package, you'll need to install from source.

1. Run `git clone https://github.com/royhills/arp-scan.git` to obtain the project source code.
2. Run `cd arp-scan` to enter source directory.
3. Run `autoreconf --install` to generate a viable `./configure` file.
4. Run `./configure` to generate a `makefile` for your system.
5. Run `make` to build the project. (Optionally run `make check` to verify that everything works as expected.)
6. Run `make install` to install (you'll need root or sudo for this part).

For example, to install a distribution package on CentOS/Red Hat, use:

Listing 2: Network Interface and Subnet Scan

```
sudo arp-scan -I enp0s3 192.168.1.0/24

Interface: enp0s3, datalink type: EN10MB (Ethernet)
Starting arp-scan 1.9.2 with 256 hosts (http://www.nta-monitor.com/
  tools-resources/security-tools/arp-scan/)

192.168.1.81  10:dd:b1:xx:xx:xx  Apple, Inc.
192.168.1.64  60:30:d4:xx:xx:xx  (Unknown)
192.168.1.69  28:cf:e9:xx:xx:xx  Apple, Inc.
192.168.1.71  dc:68:eb:xx:xx:xx  (Unknown)
192.168.1.72  88:71:e5:xx:xx:xx  (Unknown)
192.168.1.76  8c:70:5a:xx:xx:xx  Intel Corporate
192.168.1.75  84:a9:3e:xx:xx:xx  (Unknown)
192.168.1.79  cc:6d:a0:xx:xx:xx  Roku, Inc.
192.168.1.86  cc:6d:a0:xx:xx:xx  Roku, Inc.

10 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.2: 256 hosts scanned in 1.939 seconds
  (132.03 hosts/sec). 10 responded
```

Listing 3: Identifying Network Devices

```
sudo arp-scan -I enp0s3 192.168.1.0/24

Interface: enp0s3, datalink type: EN10MB (Ethernet)
Starting arp-scan 1.9.2 with 256 hosts (http://www.nta-monitor.com/
  tools-resources/security-tools/arp-scan/)

192.168.1.81  10:dd:b1:xx:xx:xx  Apple, Inc.
192.168.1.75  84:a9:3e:xx:xx:xx  Hewlett Packard
192.168.1.69  28:cf:e9:xx:xx:xx  Apple, Inc.
192.168.1.76  8c:70:5a:xx:xx:xx  Intel Corporate
192.168.1.79  cc:6d:a0:xx:xx:xx  Roku, Inc.
192.168.1.71  dc:68:eb:xx:xx:xx  Nintendo Co.,Ltd
192.168.1.86  cc:6d:a0:xx:xx:xx  Roku, Inc.
192.168.1.72  88:71:e5:xx:xx:xx  Amazon Technologies Inc.

9 packets received by filter, 0 packets dropped by kernel
Ending arp-scan 1.9.2: 256 hosts scanned in 1.953 seconds
(131.08 hosts/sec). 9 responded
```

Listing 4: Finding Duplicate IP Addresses

```
sudo arp-scan -I enp0s3 192.168.1.0/24 |grep --i dup

192.168.1.81  10:dd:b1:xx:xx:xx  Apple, Inc. (DUP.)
192.168.1.81  84:a9:3e:xx:xx:xx  Hewlett Packard (DUP. 2)
```

```
sudo yum -y install arp-scan
```

To check usage options, use the following:

```
arp-scan --help
```

or

```
man arp-scan
```

Using arp-scan on a Subnet

A good place to begin is to run a complete scan of your local network. You must run the `arp-scan` tool as root.

```
sudo arp-scan --localnet
```

If you receive an error at this point, be sure that you are running the arp-scan tool as root. If you are running as root and you receive an interface error, issue the `ifconfig` command to identify your network devices. For example: `eth0`, `em1`, or `enp0s3`. And use the `--I` option to specify which network device with which you're scanning. Remember that you can use any interface for scanning including wireless, such as `wlan0`.

The command in Listing 2 is equivalent to the one given above but specifies the network interface and the subnet.

Unknown devices are not necessarily rogue; they are simply not in the `arp-scan` vendor databases. To identify a device, you can use one of the online MAC finder sites. You only need to provide the first three octets. For example, one of Apple, Inc.'s Organizationally Unique Identifiers (OUI) is `10:dd:b1`; The unknown OUI `84:a9:3e` in Listing 2 is Hewlett Packard. My guess as to why `arp-scan` didn't identify this vendor is that the HP printer that I have connected to my network is too new and its OUI is not yet in the `arp-scan` database.

Alternatively, you can update the `arp-scan` databases. There are two methods for updating the databases. The first method, is:

```
cd /usr/share/arp-scan
sudo get-iab -v -u ➥
  http://standards.ieee.org/➥
  develop/regauth/iab/iab.txt
sudo get-oui -v -u ➥
  http://standards.ieee.org/➥
  develop/regauth/oui/oui.txt
```

This process might fail. If it does, use the second method:

```
cd /usr/share/arp-scan
sudo wget http://standards.ieee.org/➥
  develop/regauth/oui/oui.txt
sudo wget http://standards.ieee.org/➥
  develop/regauth/iab/iab.txt
```

The second method is slow, so be patient.

Also, the second method delivers the updates in a format that cannot be used by arp-scan directly. You have to convert it:

```
sudo get-iab --u
sudo get-oui --u file:///usr/share➥
  /arp-scan/oui.txt
```

These commands convert the raw `iab.txt` and `oui.txt` to `ieee-iab.txt` and `ieee-oui.txt`. You do not have to convert the files if the first method works for you. The `get` scripts perform that function.

Now, run the `arp-scan` command again as shown in Listing 3, which shows that the update has accurately identified all devices on my network.

The final scenario that I use `arp-scan` for is in finding duplicate IP addresses. This is a great feature for locating a device with a duplicate IP address if you have a mixed static and DHCP network like many of us do. You can easily find duplicates by filtering a scan as in Listing 4.

By using the MAC address, you can check your switches to find the device with the duplicate IP address and fix the problem.

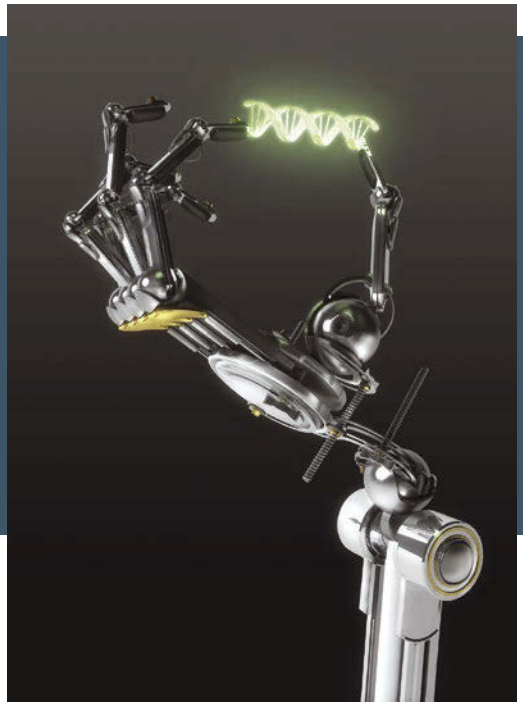
This introduction to `arp-scan` gives you an overview of this tool's power for network device reconnaissance. For me, `arp-scan` is an essential system administrator tool. If I were a hacker or pen tester, it would also be one of my favorite recon tools to gain as much information as quickly and as stealthily as possible. Please remember to use this and other security tools responsibly and always get permission to run them on your network. ■■■

Author

Ken Hess is a freelance technical writer and journalist. He covers a variety of open source topics, including Linux databases, and virtualization. You can reach him via his website at [*www.kenhess.com*].

Special Thanks

This article was made possible by support from Linux Professional Institute.



MakerSpace

Controlling a four-axis robot arm

Gentle Touch

Learn the basics of robotics with a Raspberry Pi and a PiXtend controller. *By Martin Mohr*

Robotics might seem like a daunting subject to tackle, but a Raspberry Pi, a PiXtend module, Codesys control software, and a bit of Structured Text (ST) programming are all you need to get started in this compelling discipline. In this article, I describe how to set up and program a robot arm.

For this project, I opted for the four-axis robot arm by SainSmart [1]; it has a good price-to-performance ratio and a

How Servos Work

The servos used in this project are small motors you can get from hobby shops – not industrial servomotors like those found in the big robots of the automotive industry. Strictly speaking, hobby servos are not motors at all, but actuators. Although they have a motor, additional electronic components make them servos.

The servos used in this project are controlled by a pulse-width modulation (PWM) signal. Imagine a square wave signal with a constant frequency. Now change the ratio between pulse (on) and pause (off) – PWM really is that simple (Figure 1).

PWM can be used to encode information in the pulse length. For the servomotors in this project, the length of the signal indicates which angular position the servo arm should assume. To make this work, a position sensor in the servo indicates where the servo arm is at the moment. An electronic controller continuously compares the actual position with the set point position and controls the motor so that the difference between the two is as small as possible. In practice, it is not always possible to achieve a difference of zero. When that happens, the servo starts humming. Inside the servo, a gear translates the rotary motion of the motor.

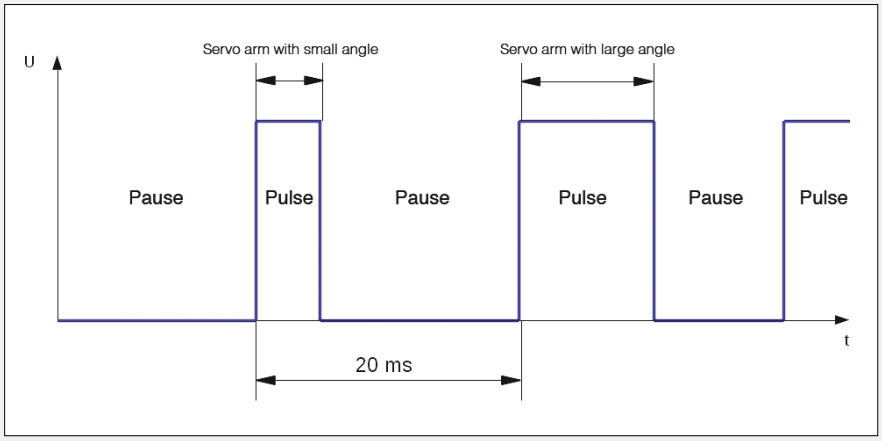


Figure 1: PWM control signal: A short pulse equals a small servomotor step; a long pulse equals a giant step.

The most important characteristics of servos include the maximum angle difference of the servo arm, the force it applies, and its positioning speed. A servo with real ball bearings and metal gears lasts longer than a plastic servo with plain bearings. The differences in quality are also reflected in the price.

Now that you know how a servo works, you need to take a closer look at the PWM signal, which has a constant frequency of 50Hz, corresponding to a signal length of 20ms. Many servos can also be controlled with a shorter signal length (e.g., 10ms). The pulse length, on which the angular position of the servo depends, is 1-2ms.

Table 1: Motor Labels

Label	PiXtend Connection	Function
M1	PWM2A	Open and close gripper
M2	PWM1B	Rotate gripper
M3	PWM0B	Extend and retract arm
M4	PWM0A	Raise and lower arm
M5	PWM1A	Rotate arm in base

large working range. Simple hobby servos let the robot arm equipped with a simple gripper [2] move objects. The Thingiverse website provides the 3D printer files for printing the gripper [3]; alternatively, you can purchase the gripper on Amazon [4]. The gripper also is driven by a servo and requires a PWM signal for control. For more information on servos, see the box "How Servos Work."

The best way to keep track of the motors is to label each one (Table 1). Figure 2 shows the robot with the gripper, and the wiring diagram in Figure 3 shows how to connect the individual motors. Please note that hobby servos can handle a maximum of 6V.

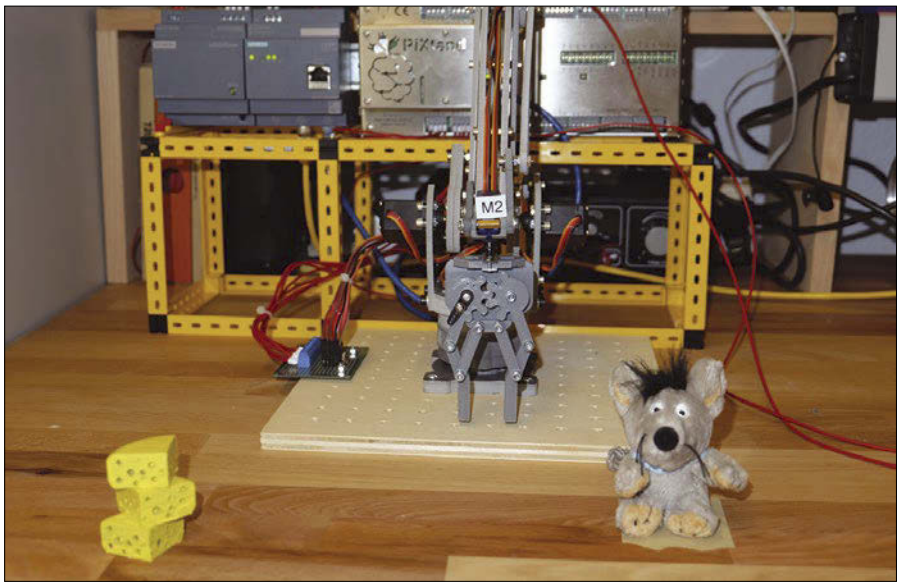


Figure 2: The complete, ready-to-use robot arm. The gripper was created on a 3D printer.

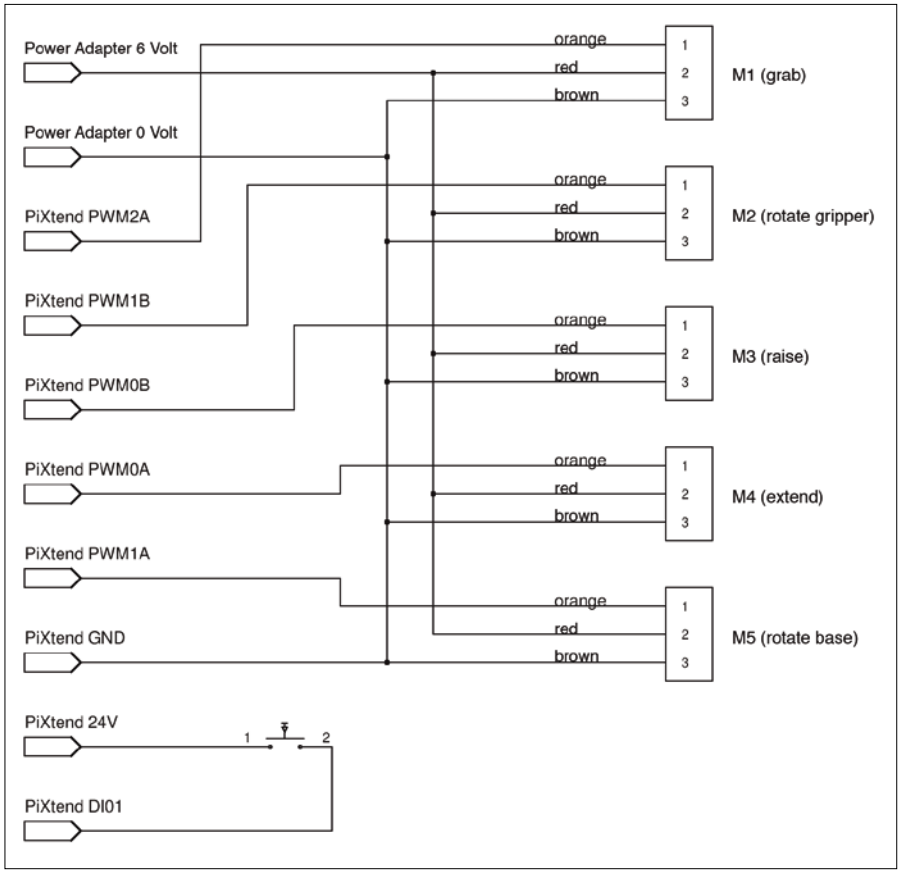


Figure 3: The wiring diagram shows how the servomotors on the robot arm and gripper are wired to the PiXtend.

Control

As mentioned, the PiXtend controls the motors. If you are not familiar with this controller yet, have a look at the "PiXtend at a Glance" box; the "PiXtend

PiXtend at a Glance

The PiXtend V2 -L- is a professional expansion board for the Raspberry Pi. It features industry-compatible I/O ports and additional interfaces that the Raspberry Pi lacks. In addition to analog inputs and outputs, the board also includes relay outputs with a very high switching capacity.

A whole range of alternatives are available for programming. The Linux tools include some command-line programs that call the PiXtend functions, which means you can easily create scripts that handle the control tasks. FHEM is a widespread, Linux-based program for home automation that runs as a Perl server. The PiXtend Python Library provides an API that allows easy access to the PiXtend interfaces. OpenPLC is open source PLC software often used at universities. Thanks to the Node-RED IoT development tool, complex processes can be displayed graphically in a web front end. The Codesys IDE helps in creating programs for industrial controllers that many controller manufacturers use. If you learn how to use this programming environment, you can create programs for almost any controller on the market.

The PiXtend board used here is only one of many PiXtend variants. The manufacturer's website has detailed information about the different boards [5]. Although the PiXtend V1 is well suited for training and further education, the V2 variants are aimed more at professional use.

PiXtend Technical Data

- 16 digital inputs (3.3V/5V/12V/24V)
- 12 digital outputs (max. 30V; 0.5A each)
- 6 PWM/servo outputs (6x16 bits)
- 4 relays (max. 230V, 6A)
- 4 voltage inputs (0-5V/0-10V)
- 2 current inputs (0-20mA)
- 2 analog voltage outputs (0-10V)
- 4 GPIOs (5V)
- Serial interface: RS232, RS485 (CAN)
- Real-time clock (RTC) with battery buffering
- Up to 4 DHT11/DHT22/AM2302 sensors (temperature and humidity)
- Onboard voltage regulator
- Input 12-24VDC (max. 30V)
- Output 5VDC/2.4A (powers PiXtend V2 -L-, Raspberry Pi, and connected USB devices)
- Retain/persistent memory: 64 bytes of flash EEPROM
- Compatible with Raspberry Pi models B+/2B/3B/3B+
- Certification: CE, RoHS

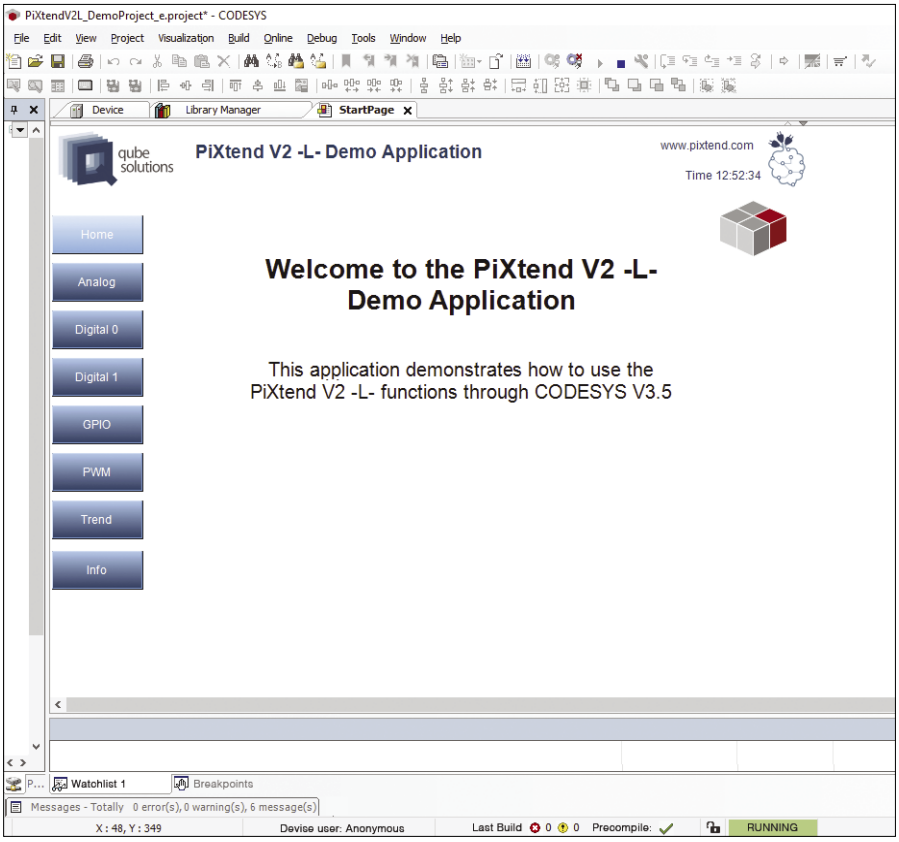


Figure 4: The Codesys demo project opens up many options for accessing the PiXtend controller.

Technical Data" box summarizes the technical specs.

I am using the PiXtend V2 -L- [6], which has a large number of interfaces that let you implement almost any geek project. Before you start programming the robot, you need to install the Codesys software on the PiXtend and on a Windows PC. Thus far, no Linux version of the program is available.

The complete software – programs and packages for Codesys, including the sample application – can be found on the PiXtend website [7]. Version 3 of the Codesys Development System is freely available online [8]; the installation is driven by a wizard. For the development environment to work with the PiXtend, you need to install two additional packages [9] [10]. Just click on the downloaded files and the Codesys Package Manager will install them.

For the PiXtend itself, you also need the appropriate software, which is available as a preconfigured SD card image [11]. You will find a detailed description for setting up the complete software in the PiXtend [12] manual.

First Test

The basis for the test is a demo project [13] that lets you control the PWM outputs of the PiXtend individually. A graphical user interface allows access from the Raspberry Pi IP address. Alternatively, you can operate the front end from the Codesys IDE (Figure 4), which provides test options for all functions of the PiXtend.

To put the robot into operation, you first need to choose the *PWM* tab for controlling the outputs (Figure 5), which should be set to *Servo Mode*. For commissioning, enable the individual PWM outputs one after the other and use the slider to set a position for the servo. The controller values allow a setting between 0 and 16,000. In the worst case, the servo will perform fast, uncontrolled movements; thus, you should center the PWM signal controls before activation.

Now you have to find out which values correspond to the maximum and center positions of the servo – these values vary from model to model. Make a note of the final and median values. You can also mark the working range in the robot's workspace: This makes programming easier later. With the help of the demo project, values can be determined

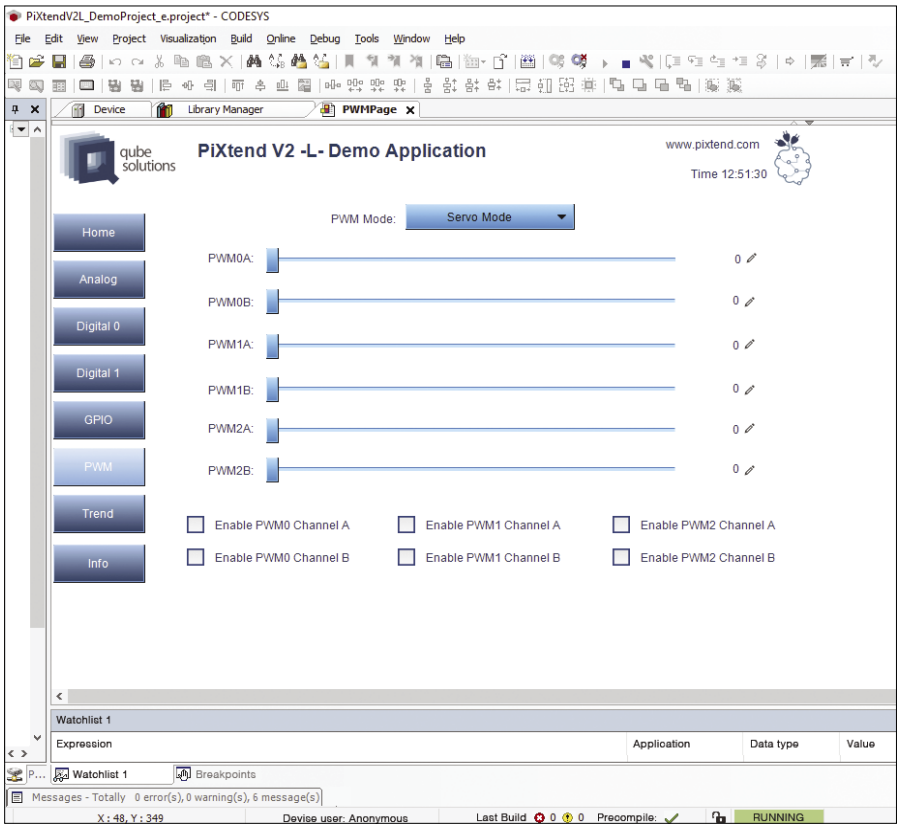


Figure 5: The Codesys demo project lets you try out the various servomotor angular positions to find the optimal settings.

for all positions that the robot needs to occupy.

During the first tests, another problem occurs with very unpleasant side effects: If for any reason (blockage or overload) one of the servos draws so much current from the power supply that it breaks down, the control electronics of the remaining motors also will no longer work correctly. All the servos suddenly move uncontrollably and chaotically back and forth. To stop them, disable all PWM outputs; then, reactivate the servos individually to isolate the problem.

A laboratory power supply with built-in current monitoring is a massive help with troubleshooting. An indication that something is wrong is a deep hum generated by the servos when they can't reach a position. Even a continuous current of more than 600mA at a fixed servo position indicates a fault. However, sometimes higher currents also flow when moving. As soon as the motor has stopped, it should no longer consume electricity. After working through the tests described here, move on to the first genuine program.

Listing 1: Variable Definitions

```
VAR
  xInit :BOOL;    // init flag
  lock  :BOOL :=FALSE;
  step  :UINT :=1;
  timer :TON;
END_VAR

VAR constant
  Steps : ARRAY [1..19, 1..6] OF UINT := [
  // M4,    M3,     M5,     M2,     M1
  12000, 8000,  6500,  8000,  8000, 2,
  12000, 3000,  6500,  8000,  1000, 2,
  12000, 3000, 10000,  8000,  1000, 2,
   6000, 3000, 10000, 13000,  1000, 2,
   6000, 7000, 10000, 13000,  1000, 2,
   6000, 7000, 10000, 13000, 12000, 2,
   6000, 3000, 10000, 13000, 12000, 1,
   6000, 3000,  9000, 13000, 12000, 1,
   6000, 3000,  8000, 13000, 12000, 1,
   6000, 3000,  7000, 13000, 12000, 1,
   6000, 3000,  6000, 13000, 12000, 1,
   6000, 3000,  5000, 13000, 12000, 1,
   6000, 3000,  4000, 13000, 12000, 1,
   6000, 3000,  3000, 13000, 12000, 1,
   6000, 7000,  3000, 13000, 12000, 1,
   6000, 7000,  3000, 13000,  1000, 1,
  12000, 3000,  3000, 13000,  1000, 1,
  12000, 3000,  6500, 13000,  1000, 1,
  12000, 8000,  6500,  8000,  8000, 2];
END_VAR
```

Robot Arm in Action

To breathe life into the robot arm, you initiate a chain of steps to tell it when to move to a specific position. The sequence of steps realized in this project works on a purely time-controlled basis and, thus, is processed without external influences one position after another. The previous test already provided which values are required for specific positions of the robot arm.

The program uses the Pascal-style ST programming language, which is frequently used in industrial controllers. The control program uses a two-dimensional array to store the complete sequence: One dimension stores the number of the current step, and the second stores the five values of the individual servos, along with a value for the time in which the robot needs to reach the position.

Listing 1 shows the variables used by the program: The `xInit` variable initializes the controller in the PiXtend and must not be deleted, the `lock` variable prevents the sequence from starting several times, `timer` implements the time delay, and `step` saves the current step in which the sequence is located.

The control array is defined as a constant, because the values do not change at run time. If you want to make changes to one of the constants, you have to make sure to load the program on the PiXtend with the *Login with download* option (Figure 6); otherwise, the software will not update these constants.

The program (Listing 2) starts after the comment in line 7:

```
// put your program code here...
```

The code that precedes that line needs the program to establish a communication channel between the PiXtend and the Raspberry Pi. The first step is to query the *DI0* start button input (line 9) by using the alias for the complete input byte. The number after the period indicates which bit you want to query. If the button is pressed and the `lock` variable is set to `FALSE` (inactive sequence), the variable switches to `TRUE`, and the sequence starts to run.

In the next `IF` block starting in line 13, the code queries the `lock` variable. If it is set to `TRUE`, the block is processed – activating the PWM outputs of the PiXtend first. The sequence then loads the values for the current step into the control registers of the PWM outputs. The robot now moves to that position.

At the same time the registers are loaded, `timer` is loaded with the wait time from the array and started. After `timer` has expired, the `step` variable is incremented by 1 and `timer` is deactivated; then, everything starts all over again for the next step in the sequence. After all the steps have been executed, `lock` changes back to `FALSE` and `step` assumes a value of 1. The sequence is then ready for the next round.

The last `IF` block starting in line 38 switches the PWM outputs to `FALSE` at the end of the sequence, deactivating the servos completely, which means they should use hardly any power.

In a YouTube video [14] of the robot arm in action, you can see

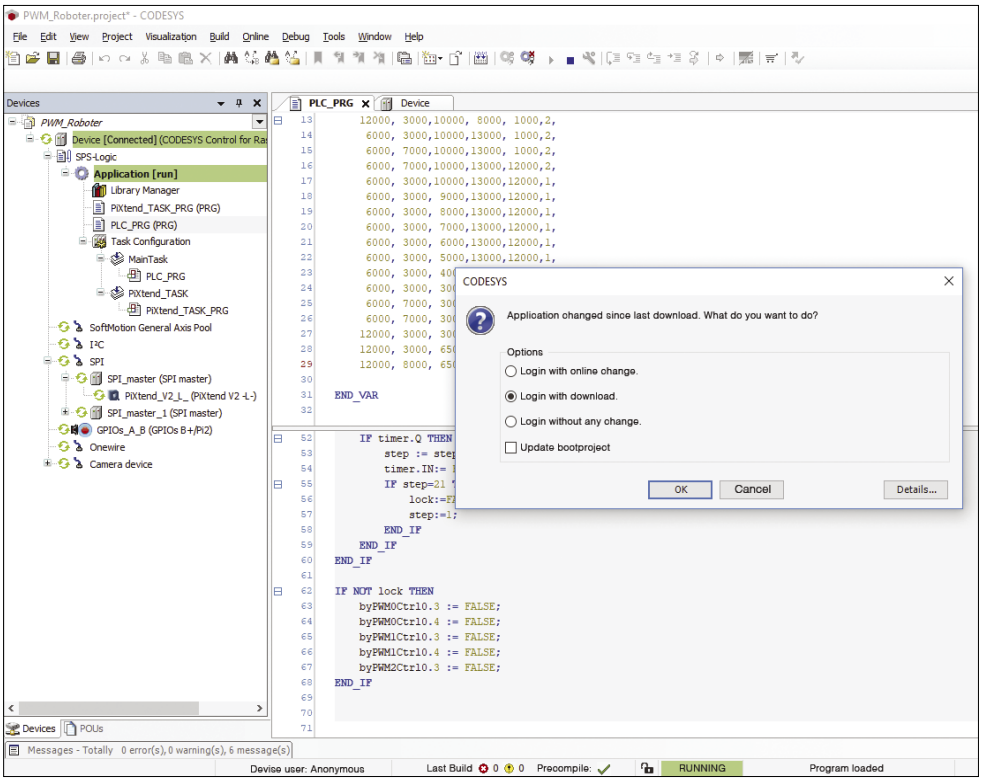


Figure 6: If you change constants, you need to load the program by choosing the *Login with download* option; otherwise, the software will not adopt the changes.

Listing 2: Positioning Sequence

```
01 // Initialization of PiXtend V2 -L-
02 IF NOT xInit THEN
03   RPI_GPIO_OUT24 := TRUE;
04   xInit          := TRUE;
05 END_IF
06
07 // put your program code here...
08
09 IF byDigitalInputs0.0 AND lock = FALSE THEN
10   lock := TRUE;
11 END_IF
12
13 IF lock THEN
14   // activate PWM Channels
15   byPWM0Ctrl0.3 := TRUE;
16   byPWM0Ctrl0.4 := TRUE;
17   byPWM1Ctrl0.3 := TRUE;
18   byPWM1Ctrl0.4 := TRUE;
19   byPWM2Ctrl0.3 := TRUE;
20   // Write Values to PWM Output
21   wPWM0A :=steps[step,1];
22   wPWM0B :=steps[step,2];
23   wPWM1A :=steps[step,3];
24   wPWM1B :=steps[step,4];
25   wPWM2A :=steps[step,5];
26   timer(PT:=DINT_TO_TIME(steps[step,6])*1000);
27   timer.in := TRUE ;
28   IF timer.Q THEN
29     step := step +1;
30     timer.IN:= FALSE;
31     IF step=21 THEN
32       lock:=FALSE;
33       step:=1;
34     END_IF
35   END_IF
36 END_IF
37
38 IF NOT lock THEN
39   byPWM0Ctrl0.3 := FALSE;
40   byPWM0Ctrl0.4 := FALSE;
41   byPWM1Ctrl0.3 := FALSE;
42   byPWM1Ctrl0.4 := FALSE;
43   byPWM2Ctrl0.3 := FALSE;
44 END_IF
```

noticeably jerky motion sequences, because the servos always try to reach the new position at maximum speed. One way to reduce this jerk is to keep the changes in values in the PWM registers to a minimum. You can see this in the video to some extent when the arm transports the plush toy across the table in a few small steps so that the arm does not fling the toy out of the gripper.

To control the servos more sensitively, you would have to design the program to approach target positions in small steps. This programming is a bit more complex and is beyond the scope of this article, but it would be a good exercise for a rainy Sunday afternoon.

Conclusions

The comparatively simple example in this article illustrates how to control a small model robot with the PiXtend controller and is suitable, for example, to familiarize students with the basics of robotics. The advantage of using the ST programming language is that almost all industrial controllers understand the resulting code.

The PiXtend V2 -L- is clearly aimed at the professional user in the industry, but its pricing makes it quite suitable for ambitious private projects. Alternatively, the smaller PiXtend boards can also accomplish the job. A selection of other models can be found on the manufacturer's website [5]. ■■■

Author

Born in the era of magnetic core ring accumulators and rotary dial phones, Martin Mohr developed a love for everything that flashes at an early age. After studying computer science, the qualified electrician developed Java applications. His old passion for electronics was revived by the Raspberry Pi.

Info

[1] Robot arm: *https://www.amazon.com/SainSmart-Desktop-Robotic-Assembled-MEGA2560/dp/B00NB1DFF2/ref=sr_1_2*

[2] Gripper: *https://www.thingiverse.com/thing:2415*

[3] 3D files for printable gripper: *https://www.thingiverse.com/thing:2415/files*

[4] Grippers for purchase: *https://www.amazon.com/Seamuing-Robotic-Gripper-Mechanical-Compatible/dp/B07DXZTQ4F*

[5] PiXtend boards: *https://www.pixtend.de/downloads/downloads-v1-3-english/*

[6] PiXtend V2 -L-: *https://designsparkmarketplace.com/en/listings/736-pixtend-v2-l-large-extension-board-plc*

[7] PiXtend downloads: *https://www.pixtend.de/downloads/pixtend-v2-l-downloads-english/*

[8] Codesys Development System: *https://www.pixtend.de/files/downloads/codesys/CODESYS%203.5.14.0.exe*

[9] Raspberry Pi package: *https://www.pixtend.de/files/downloads/CODESYS%20Control%20for%20Raspberry%20PI%203.5.14.0.package*

[10] PiXtend package: *https://www.pixtend.de/files/downloads/PiXtend_V2_Professional_for_CODESYS_V2_1_6.package*

[11] PiXtend SD card image: *https://www.pixtend.de/files/downloads/PiXtend_Image_CODESYS_V2_1_0_3_L.zip*

[12] PiXtend V2 software manual: *https://www.pixtend.de/files/manuals/pixtend_v2_software_manual_EN.pdf*

[13] Demo project: *https://www.pixtend.de/files/downloads/PiXtendV2L_DemoProject_e.project*

[14] Robot arm on YouTube: *https://youtu.be/YacAiM_SKUQ*

MakerSpace

Improving USB reliability

Programmable USB Hub

The Programmable USB Hub promises to resolve common issues with USB reliability in the short term, making way for future open hardware testing and debugging. *By Bruce Byfield*

Capable Robot Components (CRC) [1] is a company that is only 18 months old. According to founder Chris Osterwood (Figure 1), the goal of the company is "enabling robotic systems to get to market faster by allowing engineers to focus more on application-specific development, instead of building underlying infrastructure. On sites like Hackster.io and Adafruit Industries, CRC has already gained attention with SenseTemp [2], an open hardware precision temperature sensor. More recently, it has been fundraising for the Programmable USB Hub [3], a multipurpose device designed specifically for improving the reliability of embedded devices, but it also includes a variety of other functions.

Osterwood has a decade of experience in robotics companies. "I've designed everything from five-degree of freedom robot manipulators and 3D vision systems to autonomous sewer inspection robots to monocular cameras and autonomous industrial floor scrubbers. I've loved the breadth of electrical and software experience I've had." Much of his experience has involved bug fixing, code contributions, and testing in such projects as CircuitPython, CadQuery, cqparts, and Field Programmable Gate Array (FPGA) development frameworks such as Migen and LiteX. "I've benefited from open source software throughout my professional career," Osterwood says, "and I'm thrilled to be contributing back to the community here at Capable Robot Components."

Figure 1: Chris Osterwood, CRC founder and CEO.

Lead Image © lightwise, 123RF.com

Author

Bruce Byfield is a computer journalist and a freelance writer and editor specializing in free and open source software. In addition to his writing projects, he also teaches live and e-learning courses. In his spare time, Bruce writes about Northwest coast art (*http://brucebyfield.wordpress.com*). He is also co-founder of Prentice Pieces, a blog about writing and fantasy at *https://prenticepieces.com/*.

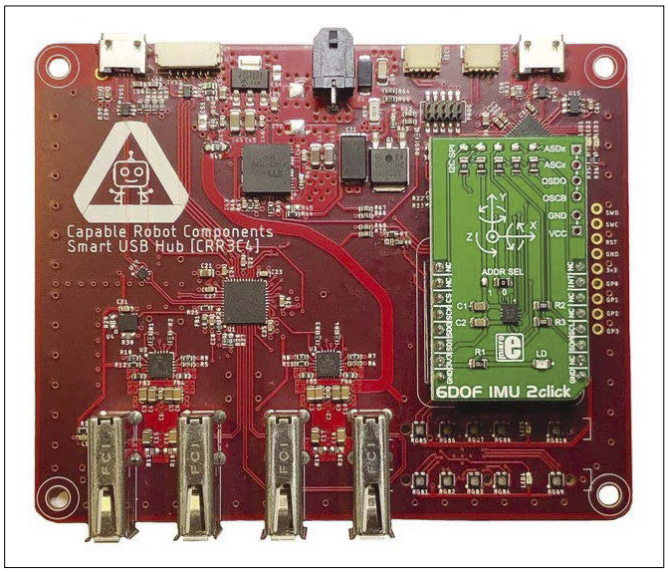


Figure 2: The Programmable USB Hub is already becoming known as a time-saving testing tool.

The idea for the Programmable USB Hub came from Osterwood's own experience with USB in mobile robotic systems, as well as from conversations with roboticists developing warehouse and autonomous picking robots. Osterwood explains that "USB is used in many robotic systems due to the high-bandwidth and the ubiquity of USB on embedded computers and sensors. It's generally faster to initially integrate a USB sensor into a system than an Ethernet-based one, and USB sensors are generally lower cost because they have a lower implementation cost. But USB has poor reliability in industrial and commercial environments, especially over long-term operation. It's relatively common for USB devices to disappear from the bus from time to time, and sometimes they don't come back. I've also seen instances where a USB device fails in a way that prevents other devices on the same USB bus from performing normally – as all bus bandwidth is shared."

A Multipurpose Tool

Primarily, the Programmable USB Hub (Figure 2) is designed to resolve such common issues. Via the hub, unreliable devices can be reset by disconnecting and reconnecting power and/or data flow. A failed device can be removed by disconnecting the data lines. "All this can be done with software controls," Osterwood emphasizes. "No access to the USB ports is needed."

LEDs are mounted on the front of the case to show the state of each of the Hub's four ports. White, green, and blue LEDs tell whether the connected device is operating at high, full, or low speed respectively, while an orange LED means that the power and data lines are disabled for a port, and a red one means that a port has been intentionally turned off. However, these LEDs are simply indicators (Figure 3).

Rather, control of the Hub is through a connected computer. In keeping with

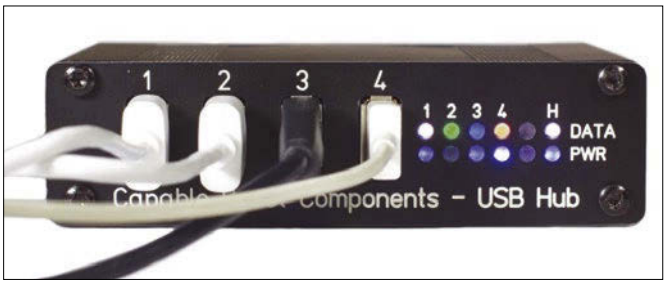


Figure 3: LEDs indicate the status of each connected device.

its modular design philosophy, the computer's connection to the Hub is a CircuitPython circuit board. Such boards are popular, easy to use, and widely available on sites like Adafruit for under $25, depending on the model. In addition, the use of a CircuitPython board means that the Hub can serve as a development board that can be easily connected to a variety of devices. To aid in this connectivity, the Hub supports both 12V and 24V nominal power input, voltages that are common in embedded devices, as well as rubber bumpers and mounting flanges to aid in customizing different physical hardware configurations.

No matter how the connected hardware is arranged, the board's USB microcontroller (MCU) gives access to the internal Serial Peripheral Interface (SPI) flash memory. The SPI flash contains the operating files that are run on the microcontroller. Because these files are written in Python – and therefore compiled each time they run – they can be edited in any file editor. From a connected computer, devices attached to the Hub can be monitored and controlled individually, with both power and data lines disconnected and reconnected as needed. When the modified files are saved, the microcontroller automatically reboots and starts to use the modified code. "No special flashing software is needed, no special editor, nor special hardware," Osterwood emphasizes. "The zero friction of changing MCU code is one of my favorite features of CircuitPython." Should connected devices include open source Python drivers, which is increasingly common these days in embedded devices, the opportunities for further control become even greater.

As Osterwood notes, the ability to program the Hub "allows for some interesting use cases." For example, enumeration can be delayed so that devices do not immediately come online. Similarly, it allows switching between several identical USB devices. Power and data can also be disconnected automatically if a connected device stops communicating or an error occurs, which helps to ensure that other devices are not affected.

However, these are only the Hub's main features. Among the Hub's other features is the ability to serve as a USB to TTL serial adapter and as a power supply that provides 5V/6A and 13mA resolution monitoring to connected devices. Yet another feature is the power to switch USB data pairs individually, which can be used to emulate the removal and addition of devices entirely through software.

In fact, the Hub's features are so numerous that Osterwood says that "the biggest challenge with [developing the Hub] was integrating all of this functional flexibility into a small device. The numerous high-speed differential pairs must be kept away from noise sources, and that's difficult to do given the amount of power being regulated and distributed in the Hub. Even seemingly easy things like putting connectors and 10 RGB status LEDs on the front and rear panels become an integration challenge when there are so many, especially when trying to minimize trace length and signal coupling."

Osterwood continues, "additionally, there are a few exclusive signal options that I wanted to present to the end user in a clear way. For example, the UART RX/TX pins and the SPI chip select pin on the internal mikroBUS header can be jumpered to either the USB Hub chip or the internal microcontroller – there are valid reasons for both configurations." Similarly, resetting power quickly was a function that was at odds with the Hub's communication with SPI devices and had to be disabled as soon as possible.

But this is only a high-level overview of the Programmable USB Hub. A more technical description of the Hub's engineering is available in a series of Jupyter Notebooks on GitHub [4].

Advance Interest

As I write, the Programmable USB Hub is a few days into its crowdfunding campaign on Crowd Supply. It is already at 70 percent of its modest goal of $10,000, so the campaign seems certain to succeed. At Crowd Supply's Teardown conference in late June 2019, Osterwood found that what excited most attendees about the Hub was the idea of using the Hub to develop and test embedded electronic devices. As Osterwood notes, "flashing, testing, and debugging microcontrollers and embedded processors generally requires a number of different hardware devices to be connected or disconnected at different times. The power and data disconnection of the Hub allow you to automate those changes – instead of physically plugging and unplugging cables."

Another popular use case according to Osterwood is the continuous integration testing of embedded electronics. "The Hub allows power and data to the device under test and device stimulus hardware to be controlled and sequenced as you need. This makes it easy to flash software, power cycle, test, and repeat with different software builds in a completely automated way."

The Programmable USB Hub is unlikely to find its way on to the average user's desktop. However, due to its potential to aid other open hardware development, it seems certain to ensure that a number of other open hardware devices will. ■■■

Info

[1] Capable Robot Components: *https://capablerobot.com/*

[2] SenseTemp: *https://www.crowdsupply.com/capable-robot-components/sensetemp*

[3] Programmable USB Hub: *https://www.crowdsupply.com/capable-robot-components/programmable-usb-hub*

[4] Technical description on GitHub: *https://github.com/CapableRobot/notebooks*

Available Now

* 2018 EDITION *

Linux Magazine Archive DVD

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

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

ORDER NOW!

shop.linuxnewmedia.com

MakerSpace

Wireless thermo-hygrometer

Comfy Digs

A WiFi sensor monitors indoor humidity and temperature and a Node-RED dashboard reports the results, helping you to maintain a pleasant environment. *By Martin Mohr*

If you live in a dry climate – or a hot region that requires air conditioning – you might find that your skin starts to feel taut: a good indication that the room air is too dry. Typical thermo-hygrometers can help you monitor humidity and room temperature, but they are usually stationary devices that can only be read when standing right in front of them.

In this article, I build a thermo-hygrometer to keep track of the temperature and humidity in your home. The device connects to a WiFi network and relies on an MQTT server to display its measured values on a Node-RED dashboard. For more information on the MQTT messaging protocol, refer to the "MQTT" box.

Initially, building your own wireless thermo-hygrometer doesn't sound that complicated, but after taking a closer look, you will realize that getting to the finished project involves a large number of individual steps.

MQTT

Like Node-RED, the MQTT server runs on the Raspberry Pi. Information about setting up Node-RED can be found in a previous article [2]. I use Mosquitto as the MQTT server; you can install it, query its status, start it, and stop it with the following:

```
$ sudo apt update
$ sudo apt upgrade
$ sudo apt install mosquitto ➥
  mosquitto clients
$ sudo service mosquitto status
$ sudo service mosquitto start
$ sudo service mosquitto stop
```

To test the server for functionality, open three terminal connections to the Raspberry Pi. In two of them, launch a Mosquitto subscriber for the `raspberry` topic:

```
$ mosquitto_sub -h localhost ➥
  -v -t raspberry
```

Lead Image © shamain, 123RF.com

MQTT

The acronym MQTT stands for Message Queuing Telemetry Transport. The protocol was developed in 1999 to monitor oil pipelines, but it is still in use today for distributed measurement in the professional sector. Today, the MQTT protocol is increasingly used in the IoT environment, because its low network load and low CPU consumption make it particularly suitable for such applications.

MQTT has three components: a message generator (publisher), a message recipient (subscriber), and a broker that decouples the communication between publisher and subscriber. Therefore, not all components have to be online at the same time to transmit messages.

To ensure that messages are not transmitted in a disorderly manner, MQTT has a tree structure of topics (i.e., message channels that bundle certain types of transmissions). For example, a topic could comprise *Building/Floor/Room/Sensor/Temperature*. If you want to retain the values of all temperature sensors in a building, you can use wildcards within topics (e.g., *Building/+/+/Sensor/Temperature*).

MQTT libraries exist for many programming languages. By default, the service runs on TCP port 1883. Further information about the MQTT protocol can be found on the official project website [1].

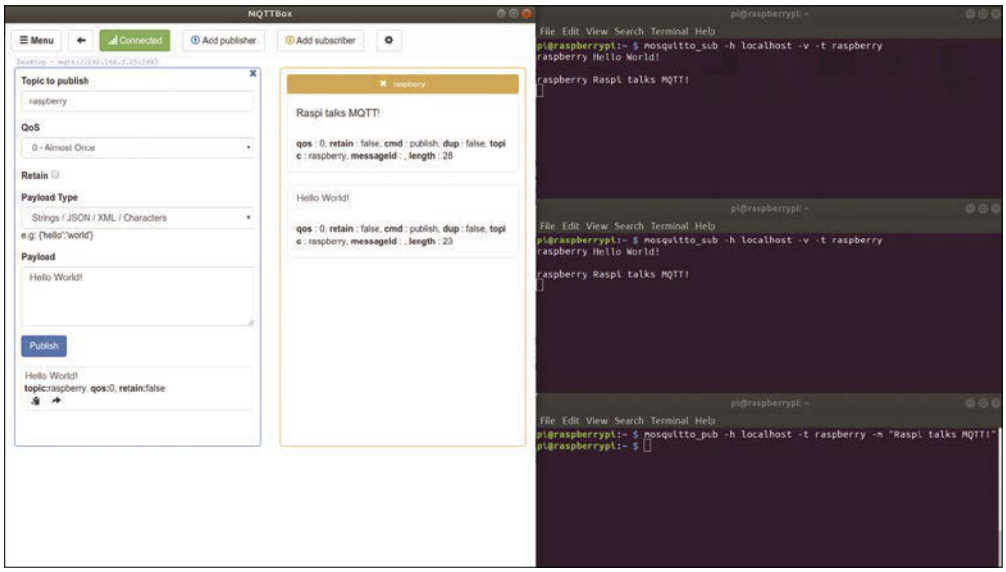


Figure 1: A typical example of an MQTT message transfer with multiple clients.

In the third terminal, start a publisher that sends a message for the `raspberry` topic:

```
$ mosquitto_pub -h localhost ↵
  -t raspberry -m ↵
  "Raspi talks MQTT!"
```

After sending the message, it appears on the two subscriber terminals. Alternatively, messages can also be sent or received with the MQTTBox Chrome app [3] (Figure 1).

The Sensor

Figure 2 shows the complete circuit diagram of the sensor. You can solder the components onto a prototype board or plug them into a PCB Prototyping Board [4]. An ESP8266 microcontroller with built-in WiFi is the central component. The calibrated digital AM2321 sensor measures temperature and humidity and is read from the I2C interface. Connect the sensor to the ESP8266 on GPIO4 and GPIO5.

A USB-to-serial module, which you plug in to the test board from a pin header, is used to program the ESP8266. This setup makes it easy to remove the module to program other sensors. Two 1.5V batteries serve as the power supply; the C1 capacitor in the circuit keeps the operating voltage of the ESP8266 stable.

The LED on GPIO14 is for test purposes. Special attention should be paid to the GPIO16 connection: The internal timer that wakes the ESP8266 from its deep sleep phase so it can transmit the currently measured value depends on this. The R7 and R8 resistors serve as pull-up resistors for the I2C bus. They ensure that the signal levels on the SDA and SCL lines are clean. All other components – four 10K resistors and two jumpers – are needed to program and operate the ESP8266.

As Figure 3 shows, the ESP8266 is soldered to an adapter board that is connected to the base board by a pin header. Some of these adapter boards already come with some of the resistors neces-

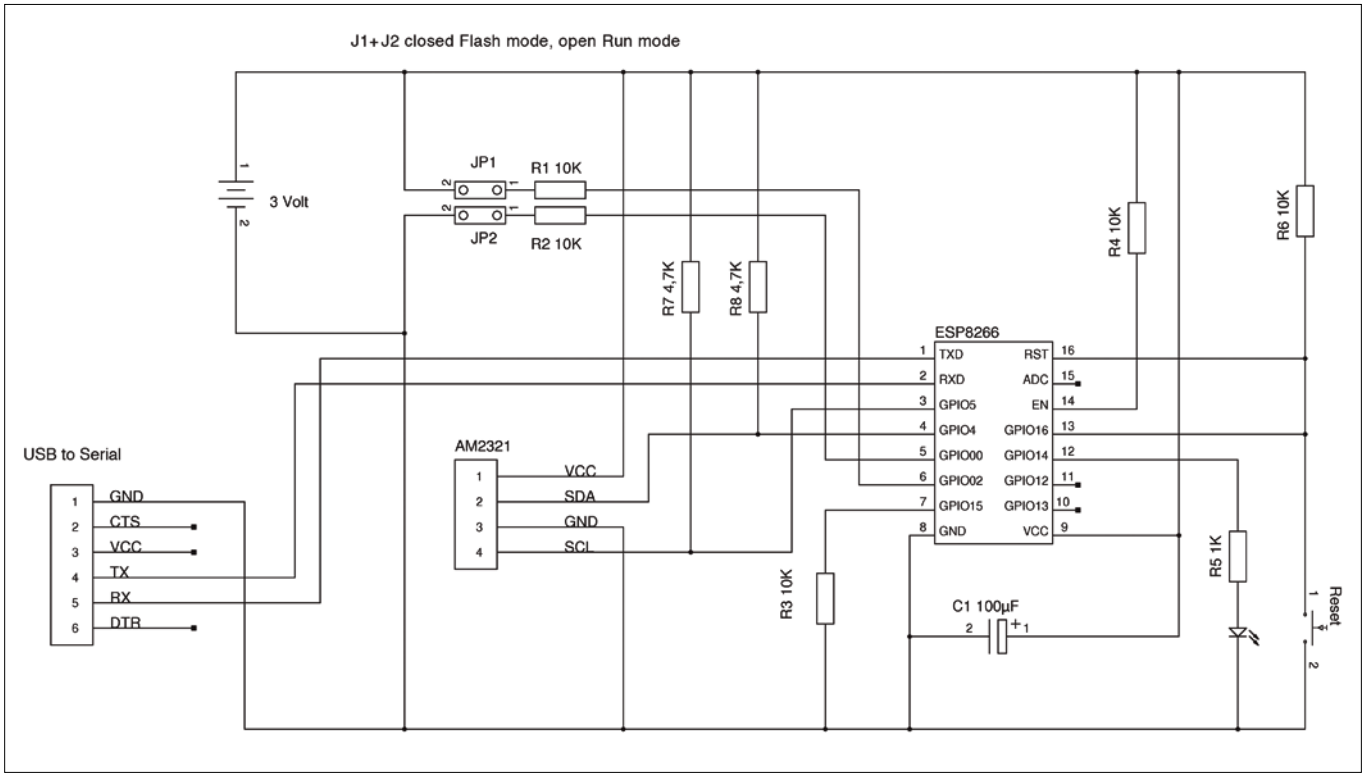


Figure 2: The circuit diagram for the thermal-hygrometer sensor. The heart of the circuit is the AM2321 temperature and humidity sensor and ESP8266 microcontroller.

sary for operation. Examples of the components needed for the project include:

- ESP8266 and ESP adapter [5]
- USB-to-serial adapter [6] (if not included in your ESP8266 kit)
- AM2321 [7]

Arduino IDE

To write programs for the ESP8266 you need an appropriate development environment. The Arduino IDE, which you can download from the project's homepage [8], is ideal: Just select the appropriate version for your operating system and install it. To start the development environment, change to the installation directory and enter `./arduino` on the command line.

Now, you have to adjust the IDE for the ESP8266 by entering the URL *http://arduino.esp8266.com/stable/package_esp8266com_index.json* in the *Additional Boards Manager URLs* text box under *File | Preferences*. The development environment then adds the parameters required for the ESP8266 to the list of board managers.

To install the board manager for the microcontroller, go to *Tools | Board: <Board> | Boards Manager*, find and select the ESP8266 entry, and click *Install*. Next, click *Tools | Board: <Board> | Generic ESP8266 Module*. To upload programs to the ESP8266, you need to connect the USB-to-serial adapter to your computer. Figure 4 shows all the settings needed to upload a program without errors.

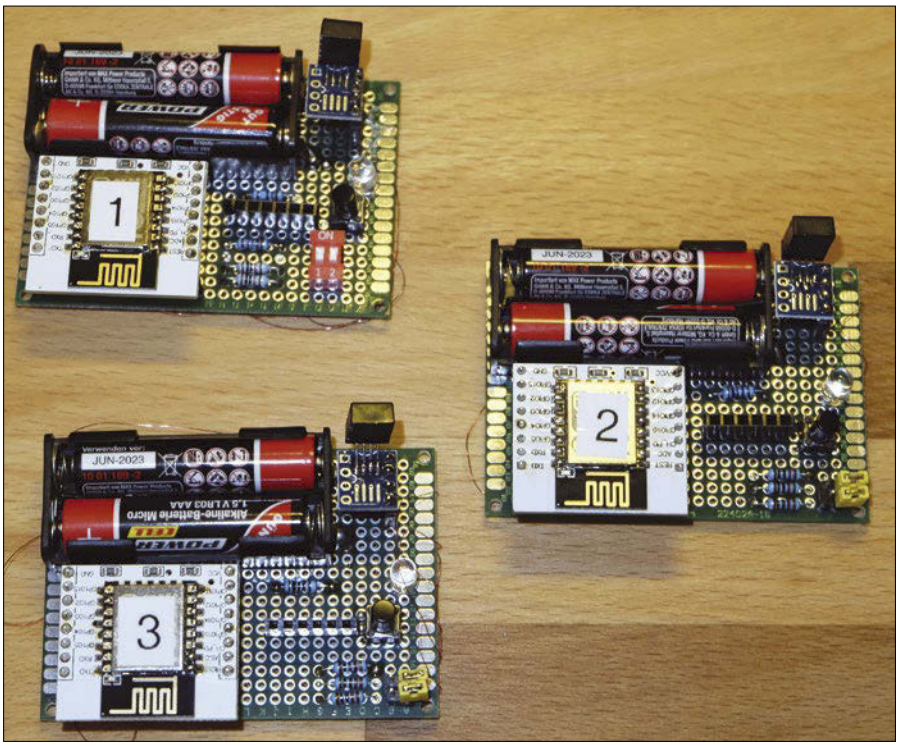


Figure 3: The three completely assembled MQTT sensors transmit the humidity and temperature to the MQTT server wirelessly at one-minute intervals.

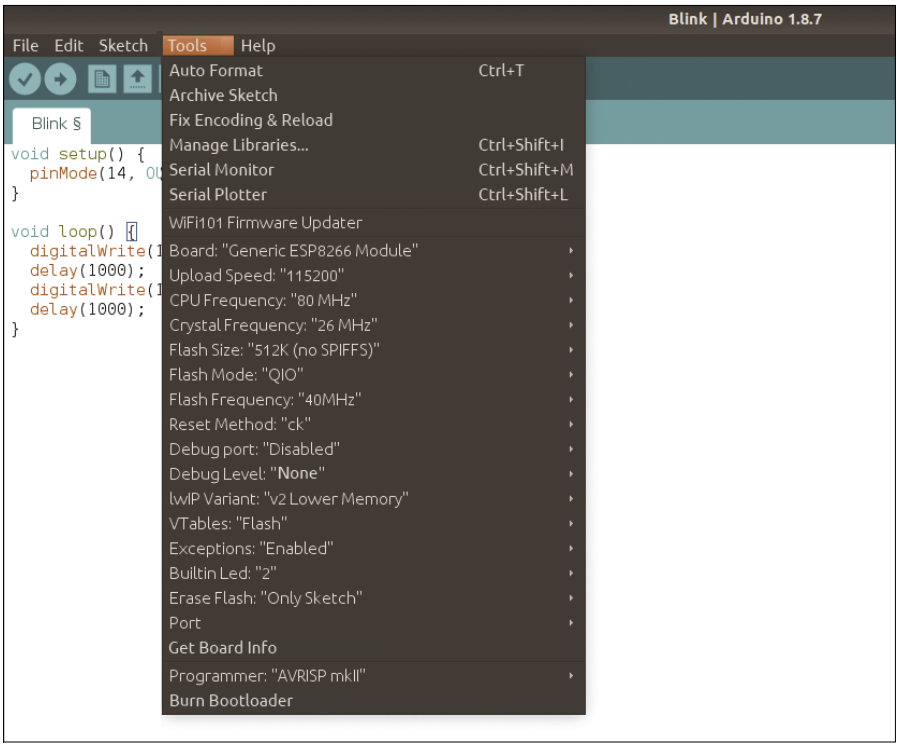


Figure 4: If you use the communication settings shown here in the Arduino IDE, uploading programs to the microcontroller should work smoothly.

Test Program

The test program in Listing 1 [9], which causes the LED on GPIO14 to flash, determines whether programs can be loaded into the ESP8266. To upload and start the program, simply switch the ESP8266 to Flash mode, close the JP1 and JP2 jumpers, and press the reset button. Now click the *Upload* button (i.e., the icon with the arrow pointing to the right) in the upper left corner of the IDE. The upload will start automatically. If the LED starts flashing, the upload was successful.

If the upload does not work right away, you should check the hardware: Did you solder everything correctly and select the correct boards manager (Generic ESP8266 Module)? Is the

Listing 1: Test Program

```
void setup() {
  pinMode(14, OUTPUT);
}

void loop() {
  digitalWrite(14, HIGH);
  delay(1000);
  digitalWrite(14, LOW);
  delay(1000);
}
```

transmission speed *115200* (baud)? In many cases, the problem is choosing the wrong port. You always need a *TTYUSB* port, not a *COM* port. Remember to remove the two jumpers when the test is finished.

The sensor needs to send its measured values to the MQTT server. To do this, you install a library by clicking *Sketch* | *Include Library* | *Manage Libraries*; then, search for the *PubSubClient* and install it. Now you can send MQTT messages from ESP8266.

Sensor Program

Listing 2 shows the sensor program. Put simply, the sensor establishes a wireless connection, reads the AM2321, and sends the data to the MQTT server. A closer look reveals many details of the program code.

After importing all necessary libraries (lines 1 to 3) the program stores all the required parameters in constants. First is the data for the wireless connection (lines 5 and 6), where you store the appropriate values for your network. Enter the con-

Listing 2: Sensor Program

```
01 #include <Wire.h>
02 #include <ESP8266WiFi.h>
03 #include <PubSubClient.h>
04 // WLAN
05 const char* ssid = "<Your SSID>";
06 const char* pass = "<Your WLAN Password>";
07 //MQTT Server
08 const char* serverIp = "<Your MQTT Server>";
09 const char* topic ="/home/sensor2";
10 const int serverPort = 1883;
11 // Energy Save
12 const int interval = 600;//sec
13
14 WiFi client client;
15 PubSubClient mqtt(client);
16
17 void setup() {
18   Wire.begin(4, 5);
19   Serial.begin(115200);
20   Serial.println("\r\n");
21   WiFi.begin(ssid, pass);
22   while (WiFi.status() != WL_CONNECTED) {
23     delay(500);
24     Serial.print(".");
25   }
26   Serial.println("");
27   Serial.print("Connected: ");
28   Serial.println(ssid);
29   Serial.print("IP address: ");
30   Serial.println(WiFi.localIP());
31   // Measurement
32   Wire.beginTransmission(0x5c);
33   Wire.endTransmission();
34   Wire.beginTransmission(0x5c);
35   Wire.write(0x03);
36   Wire.write(0);
37   Wire.write(4);
38   Wire.endTransmission();
39   delay(3);
40   Wire.requestFrom(0x5c, 8);
41   uint8_t buf[6];
42   for (int i = 0; i < 6; ++i)
43   buf[i] = Wire.read();
44   unsigned short crc = 0;
45   crc = Wire.read(); //CRC LSB
46   crc |= Wire.read() << 8;//CRC MSB
47   unsigned int humidity;
48   signed int temperature;
49   humidity = buf[2] << 8;
50   humidity += buf[3];
51   temperature = buf[4] << 8;
52   temperature += buf[5];
53   if(temperature & 0x8000) temperature = -(temperature &
     0x7fff);
54   double temp = temperature;
55   double hum = humidity;
56   temp =temp/10;
57   hum=hum/10;
58   Serial.print("Temperature:");
59   Serial.println(temp);
60   Serial.print("Humidity: ");
61   Serial.println(hum);
62   Serial.print("CRC: ");
63   Serial.println(crc);
64   //JSON
65   char msg[64] ="""";
66   sprintf (msg,"{ \"temperature\": \"%-2.2f\",
     \"humidity\": \"%-2.2f \"\"}",temp,hum);
67   Serial.println(msg);
68   // Connect to MQTT Server
69   mqtt.setServer(serverIp,1883 );
70   while (!mqtt.connected()) {
71     Serial.print("Connecting...");
72     if (!mqtt.connect("Sensor 1")) {
73       Serial.print("failed, rc=");
74       Serial.print(mqtt.state());
75       Serial.println(" retrying in 5 seconds");
76       delay(5000);
77     }
78     Serial.println();
79   }
80   mqtt.loop();
81   mqtt.publish(topic,msg);
82   delay(500); // Wait for MQTT Server
83   Serial.println("goSleep" );
84   ESP.deepSleep(interval*1000000);
85 }
86 void loop()
87 {
88 }
```

nection data for the MQTT server in the constants that follow (lines 8 to 10).

If you use more than one sensor, they have to send their measured values to different topics; otherwise, they cannot be isolated later. The last parameter (line 12) indicates how long the ESP8266 should sleep after a measurement before starting a new measurement. To conserve the batteries, the microcontroller switches off completely between measurements; the power consumption is then only in the microamp range.

The next two lines (14 and 15) define a connection to the WiFi network and to the MQTT server. Both are client connections. The advantage of MQTT is that the publishers/submitters define which topic they use. The broker provides the topics dynamically, but you have to make sure that the topics match. Because I am working with three sensors in this example, I can simply number them.

After starting a microcontroller, the `setup()` function always runs first, usually executing just the parts of the program that are required only once. In this case, the function is the complete program because all program parts have to be executed once for a measurement.

The `Wire.begin(4, 5)` command (line 18) starts an I2C interface on GPIO4 (SDA) and GPIO5 (SCL). The next line initializes the serial output of the program. All messages written here appear on the serial monitor of the Arduino IDE – make sure the monitor uses the same transmission speed as the program (here, 115,200 baud) – which makes it easier to trace bugs in the software and hardware.

The `WiFi.begin()` command (line 21) establishes and configures a connection to the WiFi network. Because this process takes some time, the following `while` loop continuously checks the status of the WiFi connection. The loop is only left after the connection has been established. The lines that follow display the connection status on the serial monitor.

Now the actual measurement begins. The first two commands (lines 32 and 33) wake up the AM2321 from energy-saving mode. The next five lines tell it to measure the temperature and humidity, and the `delay(3)` waits a while before reading the values.

A series of commands follows beginning in line 40 that read the measured values from the I2C bus and store them, formatted in variables. Line 53 is especially interesting; it ensures that even negative numbers are displayed correctly. To transfer the measured values, they are converted to JSON format (lines 65-67), and the string is stored in the `msg` variable.

Starting in line 69, the software establishes the connection to the MQTT server and transmits the message. Much like connecting to the WiFi network, it tries to establish contact in a loop until communication works. The data transfer is quite unspectacular with the `mqtt.publish(topic,msg)` command (line 81). The following `delay(500)` ensures that the ESP8266 does not go into power save mode without transmitting the message completely. If your MQTT server is running very slowly, you might have to increase the wait. The last line, `ESP.deepSleep(interval*1000000)`, in the `setup()` function puts the ESP8266 back into power save mode.

Now the measurement has been transferred and completed. When the ESP8266 wakes up after the time defined in the `interval` variable, the program restarts. The `loop()` function is never called because the program switches the ESP8266 to energy-saving mode at the end of the `setup()` function. To upload the program to the ESP8266, proceed as described for the test program.

Messages in MQTT

Before you process the messages from the sensors in Node-RED, you first need to check whether they are reaching the MQTT server as expected. To do this, use the MQTT app for Chrome described at the beginning. So that the messages from all three sensors appear in one slide, connect to the `/home/+` topic. Now, press the reset buttons on the individual sensors one after the other to send a measured value to the MQTT server. The output of the MQTT app should look like Figure 5. If the sensors provide implausible humidity values, the battery voltage is probably a little too low: The AM2321 sensors require an operating voltage of at least 2.6V.

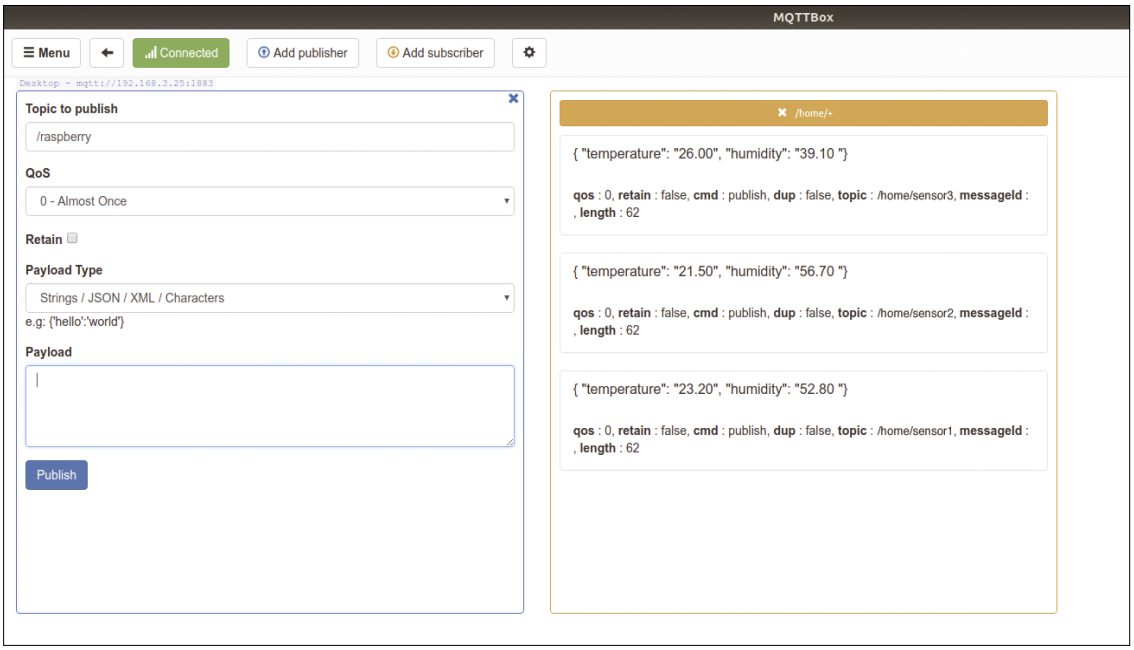


Figure 5: An example of testing the sensors with the corresponding output from the MQTT app.

Final Steps

Now you need to integrate the messages of the MQTT server into the Node-RED flow. You can do this directly via

the Node-RED interface. To do so, start the input node by typing *mqtt* and configure it to receive messages from the topic of one sensor (*/home/sensor1*). To create the input node, you need to define an MQTT server (Figure 6).

So the Node-RED can process the sensor data, the JSON strings need to be converted to JavaScript objects. Because this type of conversion occurs very often, a JSON node does this by default, allowing you to convert back

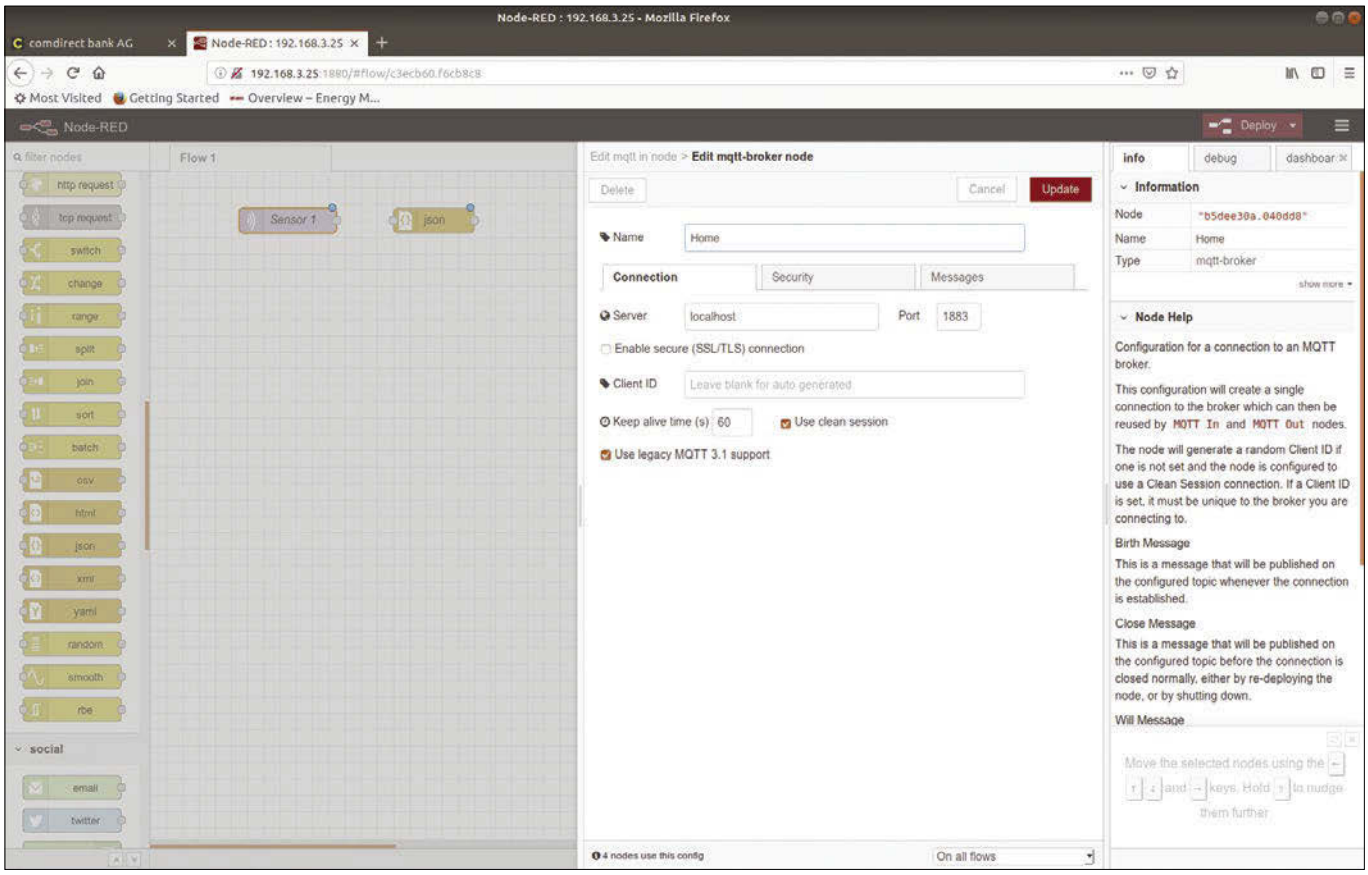


Figure 6: You can define the MQTT input node quite conveniently in the Node-RED interface.

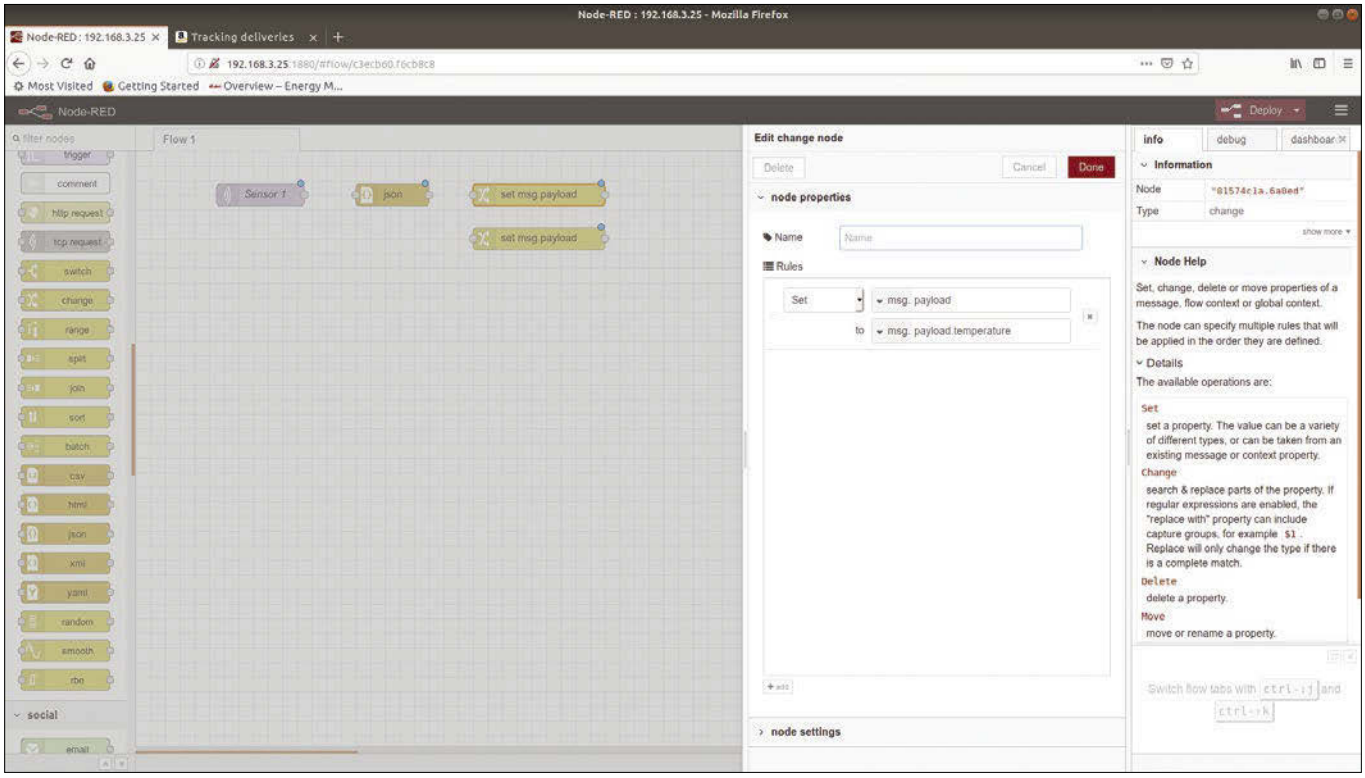


Figure 7: A *change* node extracts the temperature specification from the JavaScript object.

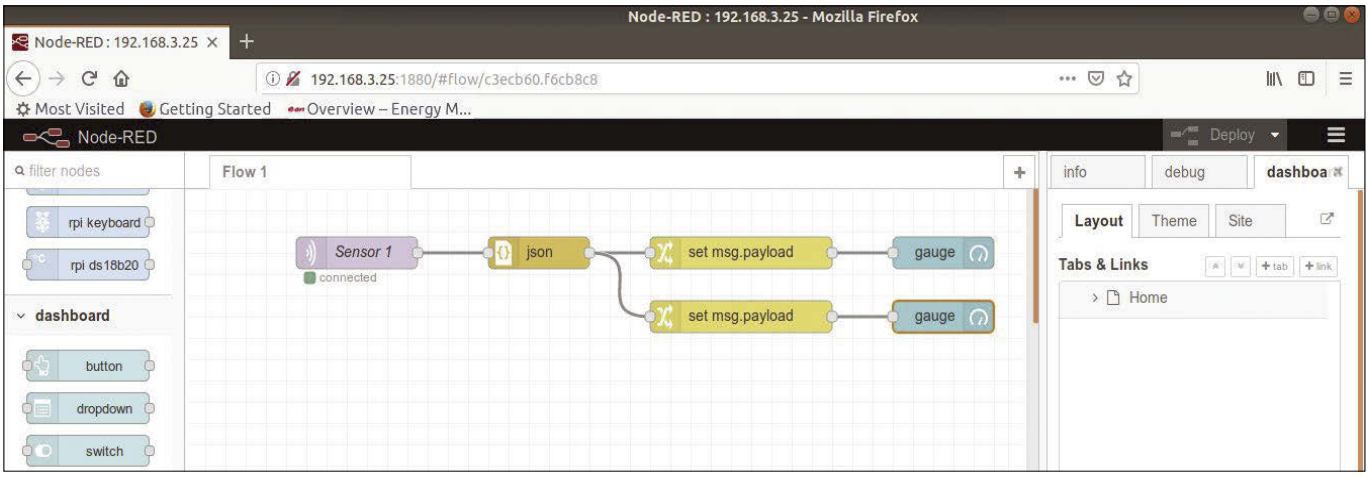


Figure 8: The Node-RED flow for a sensor with one display each for temperature and humidity.

and forth between JSON strings and JavaScript objects.

Before passing on the JavaScript object to the *gauge* display instrument, you have to separate the values for humidity and temperature. You can do this by using two *change* nodes (Figure 7), which you will set up to extract the values for temperature and humidity from the message and pack them into new messages:

```
msg.payload.humidity ↵
  => msg.payload, ↵
   msg.payload.temperature ↵
  => msg.payload
```

The *gauge* nodes have to be configured (double-click on the nodes) so that the measured values lie within their display intervals; then, connect the individual nodes (Figure 8). Repeat the described steps for the other two sensors of the encoder trio. Finally, adjust the layout for your dashboard (Figure 9).

Conclusions

The easy-to-use Node-RED tool is ideal for visualizing measured values. Other tasks from the IoT sector can also be implemented easily. When setting up the sensors, you can learn a great deal about the ESP8266 microcontroller, which is also suitable as a basis for your own applications of all kinds.

A small problem remains: The AM2321 works at a level very close to its minimum operating voltage, which is why the measured values are sometimes slightly inaccurate. To solve this problem, you would have to make the power supply of the complete circuit a little more complex by using three batteries and generating the required 3.3V operating voltage with a voltage regulator. ■■■

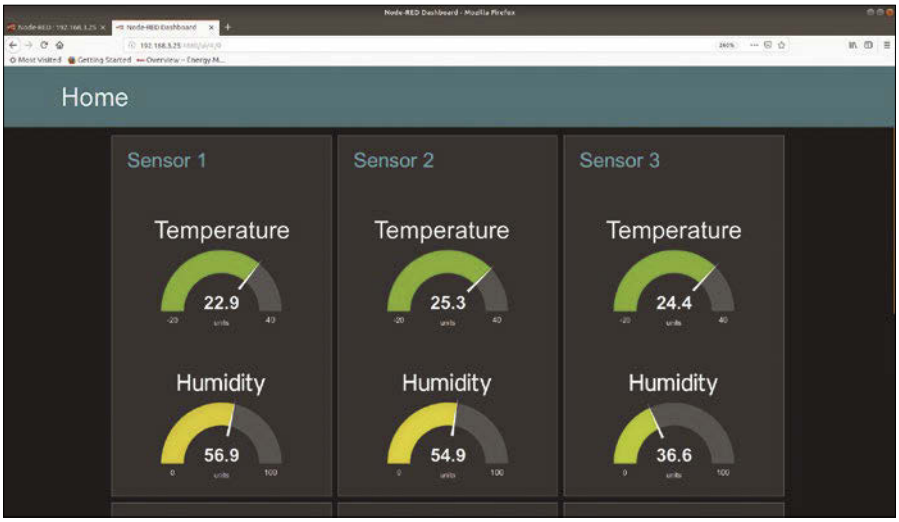


Figure 9: The Node-RED dashboard displays the measured values of the different sensors in an attractive graphic.

Info

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

[2] "RaspPi-controlled toy sailboat" by Pete Metcalfe, *Linux Magazine*, issue 216, November 2018, p. 52, *http://www.linuxpromagazine.com/Issues/2018/215/RaspPi-Controlled-Toy-Sailboat*

[3] MQTT app for Chrome: *https://chrome.google.com/webstore/detail/mqttbox/kaajoficamnjijhkeomgfljpicifbkaf*

[4] PCB prototyping board: *https://www.aliexpress.com/item/32824842884.html*

[5] ESP8266 and breakout board: *https://www.sparkfun.com/products/15258*

[6] USB-to-serial: *https://www.amazon.com/XCSOURCE-FT232RL-Adapter-Arduino-TE203/dp/B00HSX3CXE*

[7] AM2321: *https://www.amazon.com/Quickbuying-Digital-Temperature-Humidity-Replaced/dp/B07H3SSVYL/ref=sr_1_2*

[8] Arduino IDE: *https://www.arduino.cc/en/Main/Software*

[9] Code and diagrams for this article: *ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/226/*

Author

Martin Mohr developed a love of everything that flashes at an early age, which was reinforced by training to become an electronics technician. After moving on to study computer science, his field of activity was mainly developing Java applications. The Raspberry Pi rekindled his old love of electronics.

Complete Your Open Source Library with Archive DVDs!

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

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

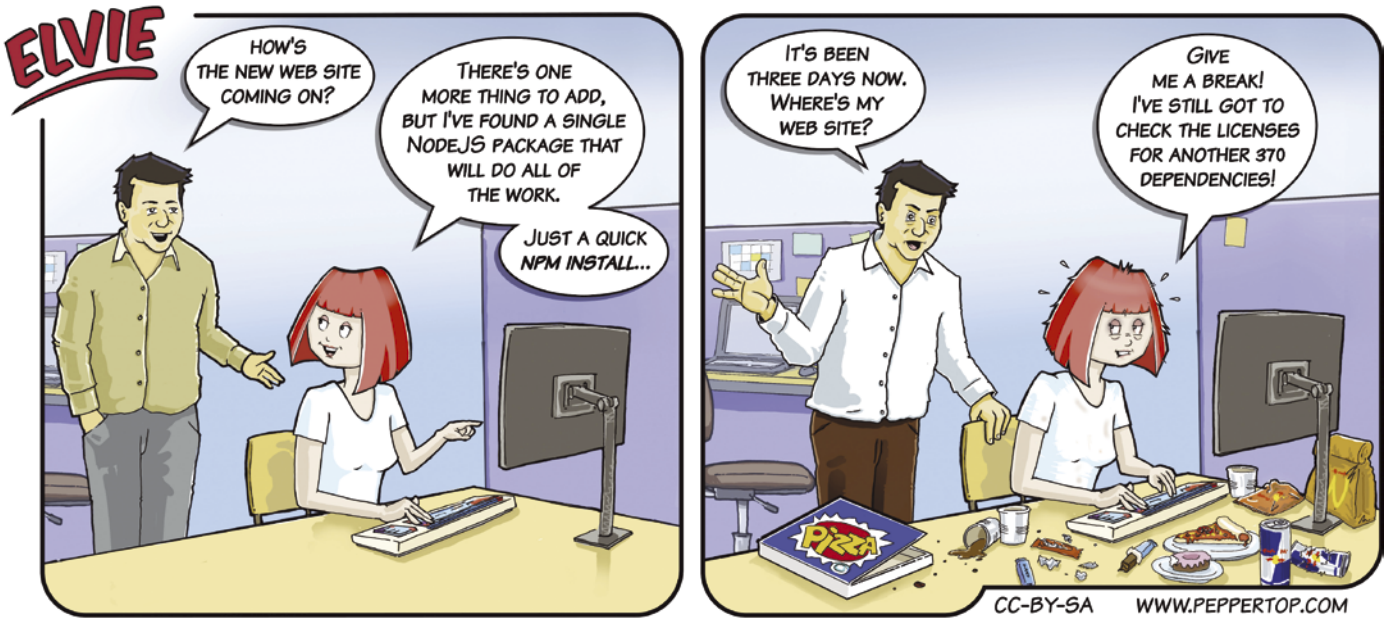
Order Your DVD Now!

shop.linuxnewmedia.com

Small is beautiful! Small Linux distros fit neatly on a USB stick and run smoothly on old systems that don't have the resources for the modern behemoths. But can a small distro meet the daily needs of a busy Linux user? In this month's LinuxVoice, we take a closer look at one of the mighty midgets highlighted in this month's cover story: Puppy Linux. We also show you how to create your own flyers and brochures using the famous FOSS desktop publishing tool known as Scribus. In our tutorials this month, we help you build a website with the user-friendly WordPress CMS, and Paul Brown rounds out his study of 3D printing with a look at some tools for monitoring the printing process.

Image © Olexandr Moroz, 123RF.com

LINUXVOICE



MADDOG'S DOGHOUSE

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

FreedomBox offers users both the convenience of the cloud and the ability to maintain control over their own data. BY JON "MADDOG" HALL

FreedomBox and the private cloud

About 10 years ago (time flies when you are having fun), Dr. Eben Moglen decided that people needed the ability to have control of the resources and services they use. While free culture can give you control of your hardware, software, and data, the use of commercial cloud services means that you may lose control over the data you use.

Eben not only drafted the GPLv3 license, but is the founder, director counsel, and chairman of Software Freedom Law Center among many other things. Used to starting and doing lots of things, and really wanting control and freedom of his data, Eban started the FreedomBox Foundation.

Eben's vision is to have a "wall wart" server so small you can put it into your pocket and take it with you. Open hardware, so you can inspect it, and free software, so you can inspect that too, configured to be secure, hardened, and private.

He enlisted a lot of people to help with this task, not the least of which was my good friend Bdale Garbee, a long time Debian leader and technologist. While they produced version 0.1 in 2012, they have been steadily improving it, and it is now a stable, usable version. In addition, there is an optional small open hardware wall wart available from Olimex, which comes with a tailored distribution that includes only the FreedomBox software based on Debian. Each modern Debian distribution now includes the code for FreedomBox, and there is a procedure that tells you how to set it up using any hardware that can run Debian.

With all the offerings of server capabilities (that we call "the cloud"), which are either relatively low cost or "free," why would anyone want to buy one of these small servers and spend the time setting up and running this specific code?

According to Eben, who teaches law at Columbia University, the issue is privacy, security, and freedom – the same tenets that drove the Free Software Foundation in the first place. While the GNU/Linux operating system (and similar movements in hardware and data) continue to drive for control of your own hardware, software, and data, large "cloud" companies use "cloud services" to take back those fundamental rights in exchange for the hardware and software they provide for you.

In addition, Eben points out that the Internet was designed to be distributed, and this distribution made it resilient to attacks of various types, but, when services become concentrated, the point of concentration becomes a focused point of potential attack, whether a denial of service attack or an attack by some agency on your data.

Eben is not against a lawful, court-ordered search of your data. What he is against is someone (whether a commercial entity or a government agency) searching your data indiscriminately.

For years, I have toyed with the concept of putting together my own server at home – one that would be on constantly, accessible from every place in the world, and would allow me to concentrate all of my data in one place.

At home, this would allow me to access my data at speeds that would be comparable to a local disk and, when I was traveling, would be (of course) slower but still give me access to any file I needed for my work on the road.

Over the years, I have satisfied this need simply by making my "server" my laptop computer and by requiring my laptop to have a considerable amount of CPU, RAM, and ever increasing amounts of disk space. The issues with this are legion, both in battery life of the laptop, weight, and issues of theft of data (normally satisfied through frequent backups and encryption).

The proliferation of devices (desktops, tablets, phones, and other "devices"), as well as other "accounts" (various email identities developed over the years), and the need to share appropriate data between them have made the lack of a private cloud device increasingly painful. In addition I write in several different audience-targeted blogs, but nowhere do I have a personal blog where I write things not related to these targeted blogs.

The advent of FreedomBox and other open source "private cloud" technologies have inspired me to restructure my electronic life.

Not only does the software allow you to (fairly) easily set up your own cloud services, but the documentation shows you how to get certificates and reverse DNS for free. The main cost after that is getting a good domain name for your site … and I have several available to me.

For those of you who are (fairly) technical, I suggest you go to *freedombox.org*, look at its capabilities, and decide if you want to set it up.

Next month, I will be describing more of the actual setup of my personal cloud. ■■■

Getting Started with Scribus 1.5

Color Outside the Lines

Scribus is a desktop publishing tool for Linux that expands your options when designing simple flyers and brochures, giving you plenty of room to be creative.

BY MARTIN LOSCHWITZ AND CHRISTOPH LANGNER

Occasionally, you may find yourself having to produce a brochure, flyer, or other printed products. If you use LibreOffice and similar programs, you will probably achieve acceptable results – but LibreOffice Writer is not the ideal tool for such tasks.

In fact, there is a separate program category for designing printed products: desktop publishing (DTP). Scribus [1] is a classic DTP tool for Linux. Unlike LibreOffice, it offers a variety of approaches to putting text around photos, arranging text in many different ways, linking text modules across page boundaries, creating contiguous double page spreads, and much more.

Although the Scribus developers strive to make their tool as simple to use as possible, it can be hard to know where to get started with Scribus. This article introduces the program and shows you some of the basic features by creating a simple flyer.

Where to Get It

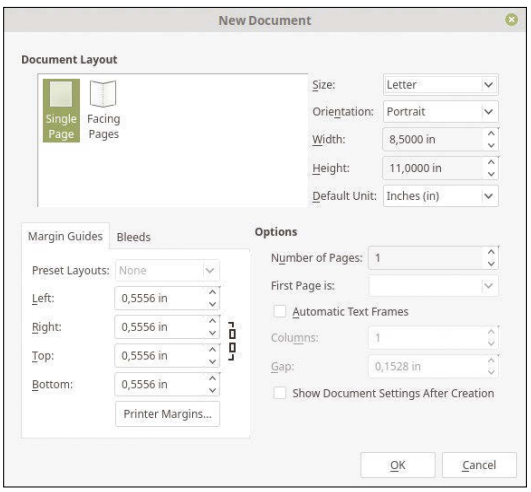
Scribus is included with most distributions. However, this does not mean that a `zypper install scribus` on openSUSE or an `apt-get install scribus` on Debian, Ubuntu, or Mint will generate the desired effect. Before installing Scribus, you need to decide whether you want to use Scribus 1.4 or Scribus 1.5.

Scribus 1.4 is more than a little out of date; its interface in many ways no longer reflects today's standards. The Scribus 1.5 branch has long been an alternative, but Scribus developers still say that it is under development, which translates to the software having a beta status. In everyday use, however, Scribus 1.5 turns out to be just as stable as its predecessor Scribus 1.4, which has not yet been replaced, and it offers many more functions.

To get started with Scribus, it makes sense to use version 1.5. However, this is not included with every distribution out of the box. Fortunately, on their download page [2], the Scribus developers point you to a large number of different Scribus packages for the individual versions and also for other operating systems.

This is one of the program's greatest strengths: If you use Scribus on Linux, you can edit the file later on – for example in the office – with Scribus on Windows or macOS. For the popular Fedora and openSUSE distributions, Scribus 1.5 RPM packages exist in a separate repository. For Ubuntu, the developers run their own PPA from which the `scribus-ng` package can be installed.

Figure 1: You will find many options when creating a new document in Scribus, but start with *Size* and *Orientation*.



First Steps

You'll start by creating a new document, just choose *New* from the File menu. In the New Document dialog box, choose the size of the document and the page orientation (Figure 1). Then press *OK* to create the document as described. Scribus now displays the main window, with the empty page of the newly created document occupying most of the window.

There is no harm in taking the time to familiarize yourself with the Scribus user interface. Although the view is very reminiscent of the familiar look of word processors, Scribus is a DTP tool that works differently than a classic writing program in many ways.

As usual, there is a menubar at the top that gives you access to all the Scribus functions. What is more important though is the line underneath it. This is where you will find a variety of icons that provide quick access to the most important features of the program. Scribus comes with a large number of tools, but you can quickly locate the most important ones here. These include the text and image icons, which you use to create text boxes and embed graphics in the document. Below the document you will find a further bar where you can change the zoom level with a mouse click.

Let's briefly look at the contents of the Windows menu in the taskbar. This is where you open various additional windows for the individual Scribus functions. As soon as you start using text boxes in the document, you will find options for the font type and size, as well as the desired layout in the additional Text window. If you do not immediately see this window, click on *Properties* and choose the *Text* option.

Adding Color

A white background on a flyer is about as creative as socks as a Christmas present. You can immediately make your flyer more interesting by adding a background color. You do this by creating a frame in Scribus, extending it to cover the entire page, and then selecting a fill color.

First select the shape icon with the stylized gray rectangle. Then click in an area of the page, hold down the left mouse button, and drag a rectangle of any size onto the sheet. Now release the left mouse button. Scribus then draws the new box on the screen and automatically selects the element, as you can see from the red border. With the frame selected, you can resize it to the desired size and drag and drop the frame so that it is exactly on top of the document. The snap to grid guides that are enabled in Scribus by default will help you do this.

Now right-click on the frame and select *Properties*. Scroll down to *Colors* and select one of your choice to use as a background. You can add new colors by going to *Edit* | *Colors and fills* and creating a new color. Explore the options in the colors dialog box to create an endless variety of new colors (Figure 2).

Working with Text and Images

The advantage of working with DTP tools such as Adobe InDesign or Scribus is that they offer considerable flexibility in placing text on the document. A text frame can be positioned anywhere in the document and easily resized or rotated.

You can also connect different text frames so that they work like a single text frame. This is useful when you want to work across a page break, for example, as part of a double-page spread. In addition, text can be made to wrap around images, and it doesn't have to appear in a straight line (Figure 3).

Text can be processed in Scribus in many ways. In the simplest variant, you click on the icon that allows you to insert a text frame, drag a rectangle on the page as you did previously for the background shape, and enter the text. You may also have an icon that allows you to edit text. If you want to adjust the appearance of the text, you can do this via *Windows* | *Text Properties*.

Don't be fooled by the fact that Scribus always draws a thin line around the text frame – this frame disappears when exporting or printing. The box is only intended to help you estimate the outline of the text box when creating the flyer.

Similarly, you can place images in Scribus by using the photo frame. You will find the icon

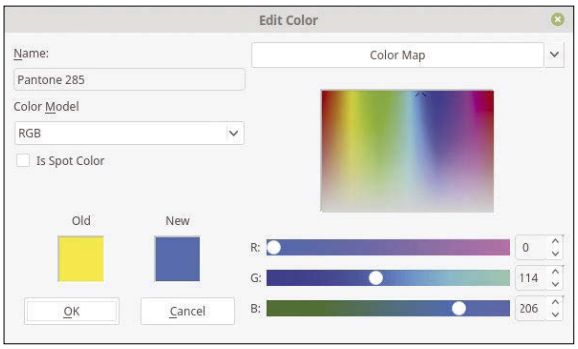


Figure 2: Scribus supports both RGB and CMYK color models and has many advanced features for choosing colors.

next to the icon for inserting text. Drag the frame to the point in the document where you want the image to appear. Right-click it and select *Get Image* from the menu. Now select the desired image, which then appears in the frame. Then right-click the photo again and select *Picture | Adjust Image to Frame*. Now the frame with the image can be moved within the document like any other object type.

Once you have finished designing your flyer, you need to prepare the document for the printer, whether for a simple office printer or a professional print job. To export as a PDF, go to the icon bar at the top and click on the PDF symbol, the seventh from the left. While there are many options for saving files that would be useful for a professional print job, you can also use simple presets.

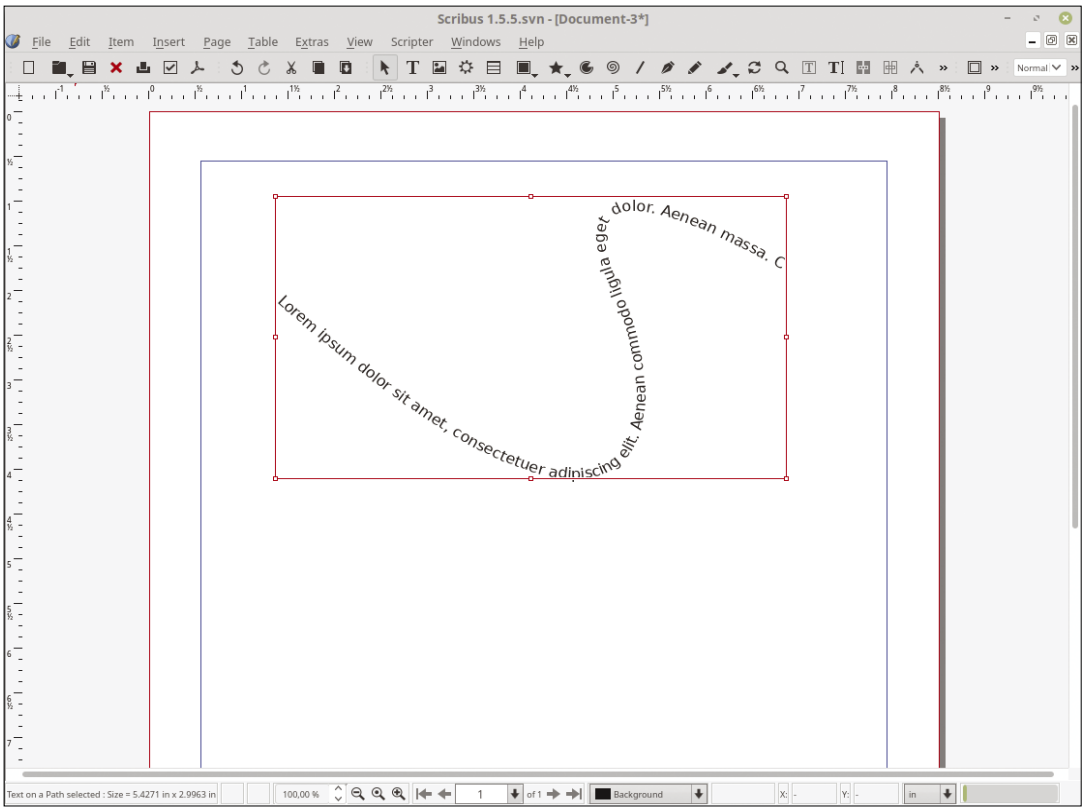
More Advanced Features

Scribus supports working with layers. This allows elements in the document to be edited individually, although they overlap. Layers also let you apply various special effects. And if something goes wrong, a layer can be quickly deleted and rebuilt – at least faster than the entire document. Selecting *Windows | Layers* takes you to the Layers dialog, where you can create new layers; it also gives you the option to delete layers you no longer need.

For many print projects, you may choose to work from a template. Scribus v1.5.0 and later let you open documents created with Adobe InDesign. However, support is limited to the Adobe DTP program's XML exchange format (IDML) and Adobe InDesign Snippets (IDMS). The common INDD format, in which numerous free and commercial templates are also available, is only on the wish list [3] thus far. The Scribus developers would not only have to crack the proprietary Adobe format in order to implement this function, but would most likely also have Adobe's lawyers breathing down their necks [4].

So if you stumble across a template on the web that was built for Adobe's professional publishing software, you should ask a friend or colleague to export the document from InDesign as IDML. In this way, you can build a new document from this template when you launch Scribus and edit it as you wish. This is always more convenient than building a document yourself – especially when it's not just a simple flyer, but a more elaborate creation. ■■■

Figure 3: Working with the more advanced tools in Scribus, you can make text flow in any shape or direction you choose.



Info

[1] Scribus: *https://www.scribus.net*

[2] Scribus 1.5: *https://www.scribus.net/downloads/unstable-branch*

[3] File format wish list: *https://wiki.scribus.net/canvas/File_formats_that_should_be_supported_by_Scribus_(wish_list)*

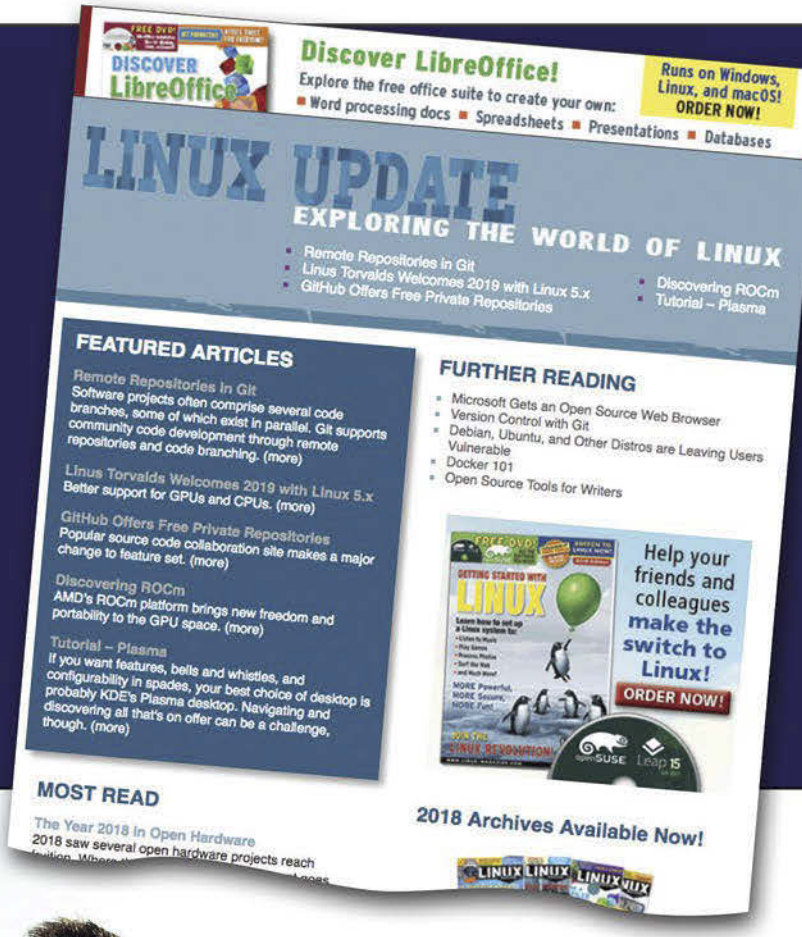
[4] "Why Scribus doesn't support QuarkXpress and other major publishing applications": *https://wiki.scribus.net/canvas/Why_Scribus_doesn't_support_QuarkXpress_and_other_major_publishing_applications*

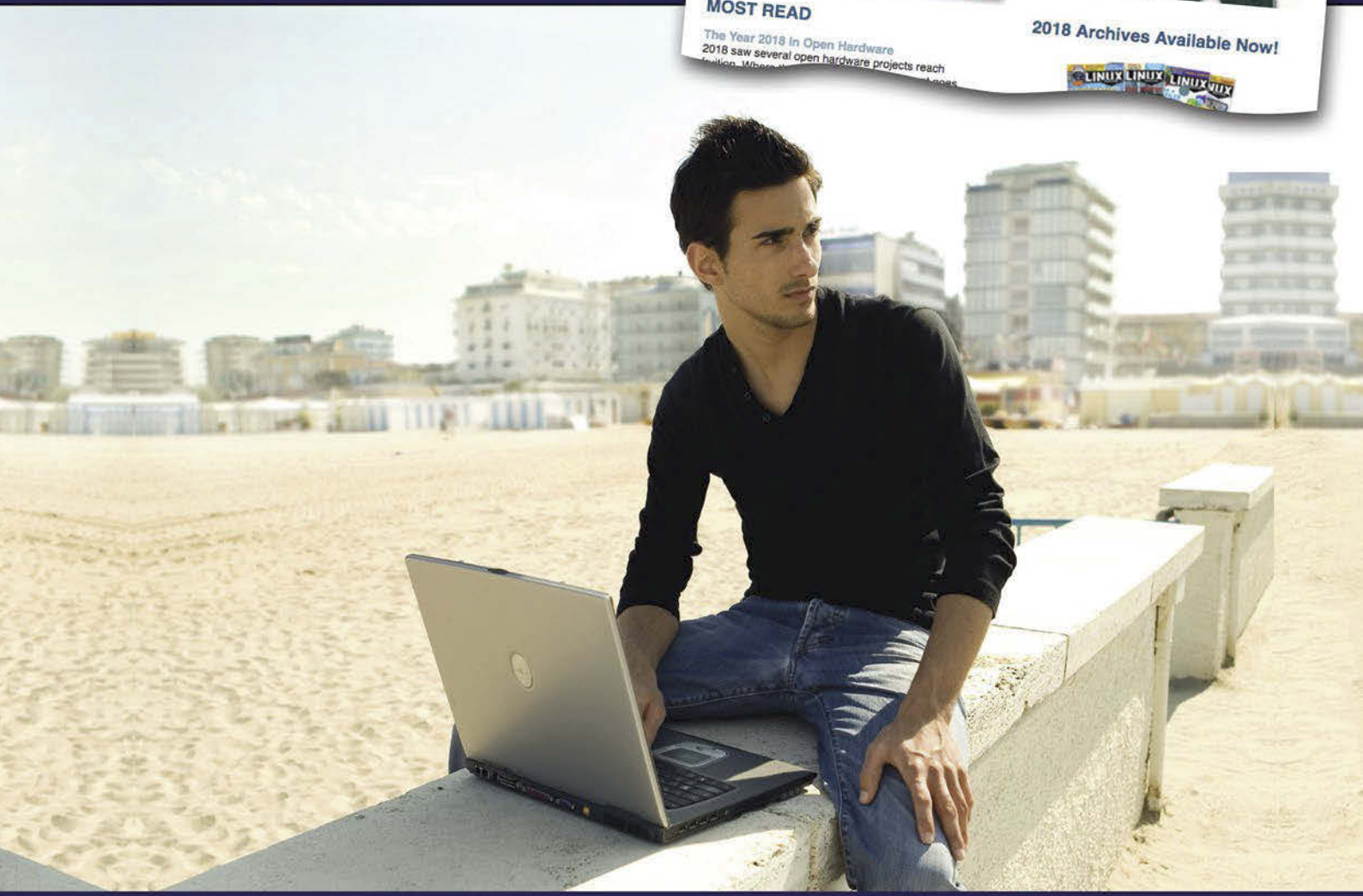
LINUX UPDATE

Need more Linux?

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

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





https://bit.ly/Linux-Update

Get to know BionicPup, the latest version of Puppy Linux

Puppy Love

Like its predecessors, this ultralight OS is fast and versatile with an easy-to-use interface. BY NATE DRAKE

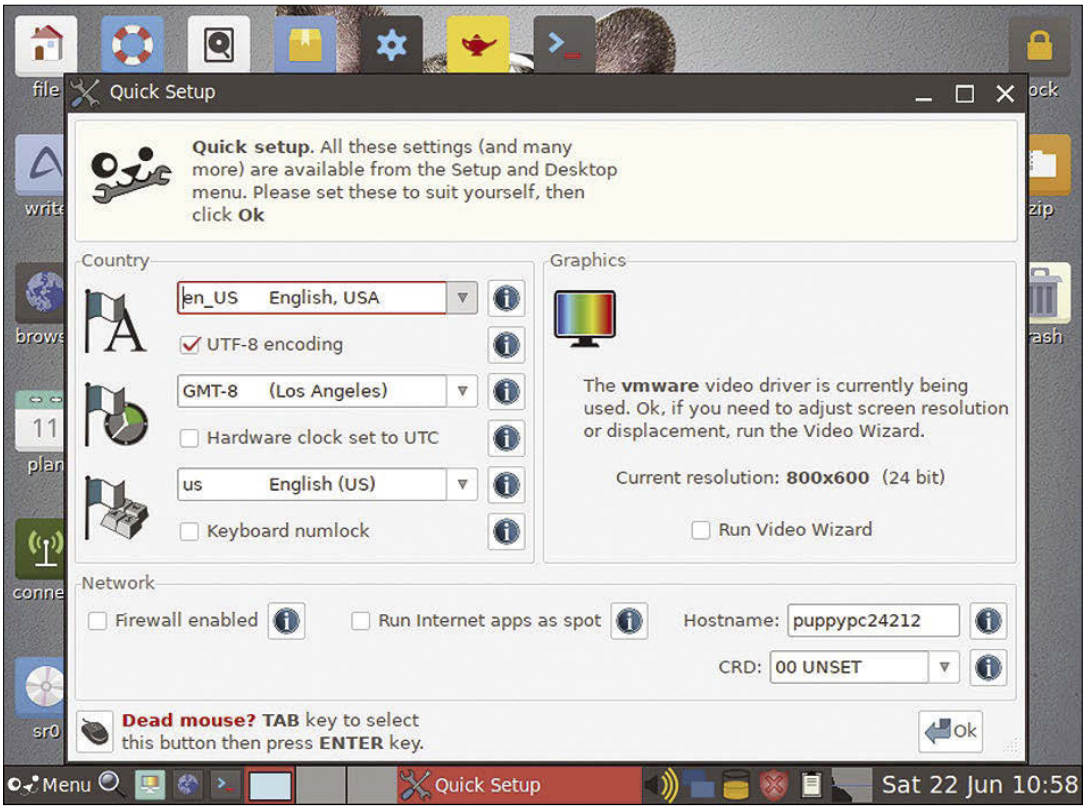
It's small, zippy and eminently lovable: For 16 years the bounder that is Puppy Linux has stolen Linux users' hearts with its cutesy graphics, inhumanly fast load times and simple interface. The operating system (OS) was originally whelped by Barry Kauler in 2003 and has since been nurtured by a number of developers. Each new version of Puppy is based on the most recent Ubuntu Long Term Support (LTS) release. At the time of writing, this is Ubuntu 18.04 Bionic Beaver, hence the code name of version 8.0: BionicPup. This review focuses on BionicPup, which like all versions of Puppy Linux is designed to load directly into your machine's RAM from a Live CD or USB. This makes the OS perfect for quick and dirty data retrieval from a corrupted system, as well as offering an easy way to use friends' computers without interfering with their lives.

Puppy can also be installed onto a wide variety of systems. Its requirements are so light that it supports hardware considered obsolete by modern standards. For this review, we installed BionicPup in a virtual machine with 1GB RAM and an 8GB virtual hard disk.

Getting Started

If you've been bitten by the pup bug, visit the download page [1]. From here, you can download an ISO file of the most recent 32- or 64-bit version of Puppy Linux. True to its ultra-lightweight reputation, the current ISO weighs in at just 354MB. If you're using a more modern device

Figure 1: Use Quick Setup to set your preferences for language, keyboard setup, graphics, and more.



with no DVD drive, we recommend using a third-party tool such as Etcher to create a bootable USB from the ISO file [2].

Power users can do this directly from the Linux command line using `dd` for example:

```
sudo dd if= bionicpup64-8.0-uefi.iso
  of=/dev/sdc
```

Make sure to substitute `bionicpup64-8.0-uefi.iso` and `of=/dev/sdc` for the name of your Puppy Linux ISO and the address of your USB device respectively.

The most popular variety of Puppy is based on Ubuntu, but there are also versions based on Slack and Debian. At the time of writing, there's no official version for ARM-based architectures, such as the Raspberry Pi. However, you can use the Woof-CE utility [3] to build a custom version CD of Puppy Linux if you wish. Once you've downloaded or created your install medium, insert it into your machine and power it on to boot into Puppy's Live desktop environment.

Unlike most distros, Puppy loads directly into your RAM. According to the Puppy Linux wiki [4], this is achieved through Puppy's unique boot process: First, the Linux kernel, `vmlinuz`, loads into your machine's virtual memory followed by `image.gz`, which loads into a fast ramdisk. This file is then decompressed into this ramdisk, turning the ramdisk into the very basic Puppy filesystem. The end result is a live desktop environment, which responds nearly instantly to your clicks and commands.

First Boot

Most variants of Puppy Linux use Openbox and JWM to create the desktop interface. Click the stylized Puppy button at the bottom left of the screen to launch the main menu. From here, you can change system settings, such as your network configuration. You can also launch apps: These are neatly categorized into sections such as Multimedia and Business.

If you don't feel like navigating the various program categories, you can also find a number of default apps on the desktop. Most of these are self-explanatory (e.g., Trash). If your machine connects to the Internet via WiFi, double-click *Connect* to launch the network setup wizard.

You can also manage your network settings by clicking the *Interface* icon at the bottom right of the screen. This is nestled amongst a number of other useful default programs such as a clipboard and storage managers. Windows automatically minimize to the taskbar at the bottom of the desktop.

If you're planning to install Puppy Linux, take some time to work through the Quick Setup guide (Figure 1). From here, you can adjust your country and language settings, as well as your default language and keyboard layout. If you need to change your graphics settings, such as altering your screen resolution, check the box marked *Run Video Wizard*. Visit the *Network* section to enable Puppy's firewall and change your hostname. Once you've worked through the Quick Setup options, Puppy Linux will also open a Welcome screen. If you haven't done so already, click *Internet Connection* to configure your network. The *Setup* section offers more advanced features than the Quick Setup guide you used previously. For instance, you can configure printer and sound settings.

If you need further guidance, click *I need help!*. This will launch the comprehensive help page (Figure 2) for your specific version of Puppy. This is written in HTML and located in `/usr/share/doc/`.

Default Puppy Apps

On first launch, Puppy offers a choice selection of preinstalled applications.

One of the programs bundled with BionicPup is Claws Mail. This GTK app originally started out as Sylpheed-Claws in 2001 and was used as a test bed for new features rolled out for the Sylpheed mail client. However, it became an entity in its own right by 2006 when both codesets were no longer synced together. Claws has many more features compared to its counterpart Sylpheed and is still just as lightweight and stable. In 2015, a major vulnerability was discovered in Claws Mail whereby plaintext versions of passwords were sent to IMAP servers, making them vulnerable to interception. If you are still worried about this, you can download another mail client, such as Thunderbird.

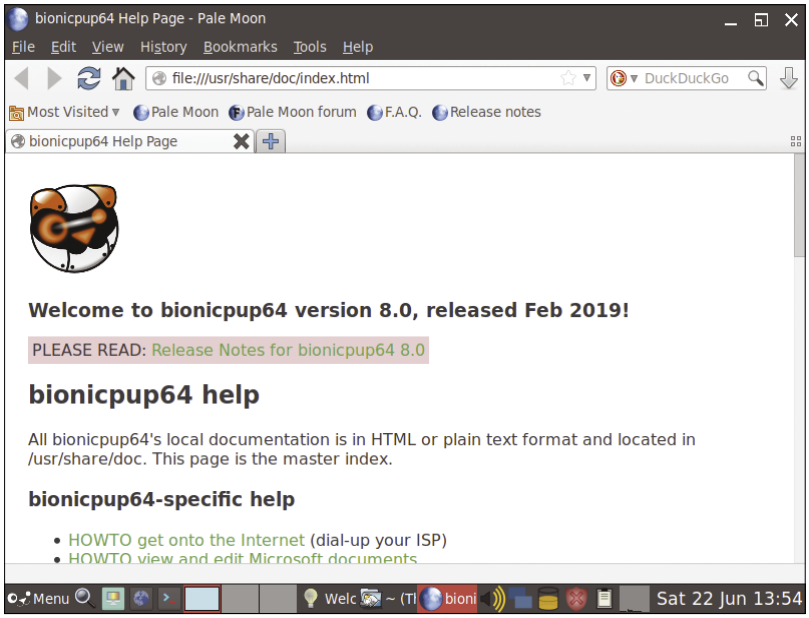


Figure 2: When you click on the *I need Help!* button, Puppy's browser, Pale Moon, launches with the help page.

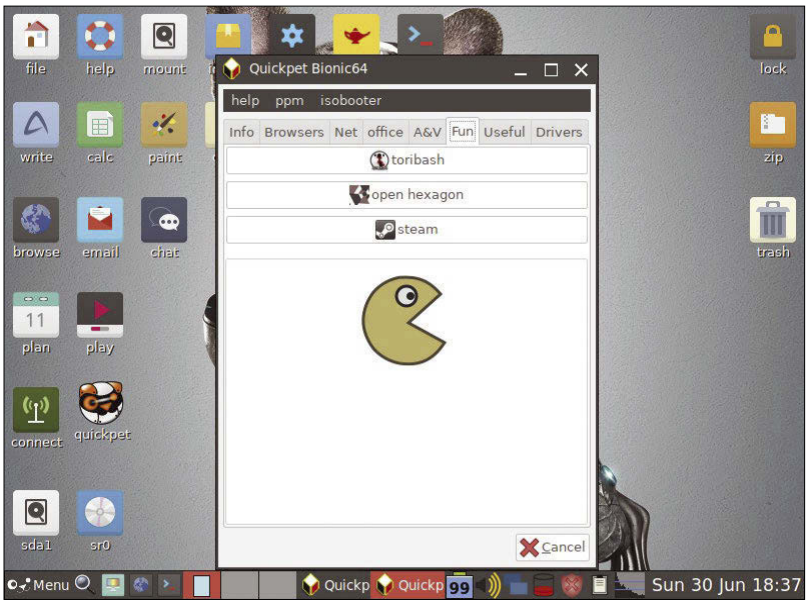


Figure 3: While Quickpet may not have the same variety you can find for larger distros, it still offers some of the most popular programs.

HexChat is another useful app. It's a free and open source (FOSS) program based on XChat. It has been translated into several languages and is actively developed. HexChat includes spell check, SASL Proxies, and DCC support among others. When you first open HexChat, you will be greeted with a Network List pop-up box. Simply fill in your details to start chatting.

BionicPup comes with its own browser: Pale Moon, which is based on Mozilla/Firefox. Users can install plugins such as Adblock Latitude (specially developed for Pale Moon), YouTube, Baidu, and eBay among others.

Puppy Linux uses AbiWord as its word processor. To automatically open a new document, click the *Write* shortcut on the desktop. AbiWord offers only limited support for saving documents in Microsoft Word format (`.docx`). If you know that you'll be exchanging files with Windows users, you might want to install the LibreOffice suite instead.

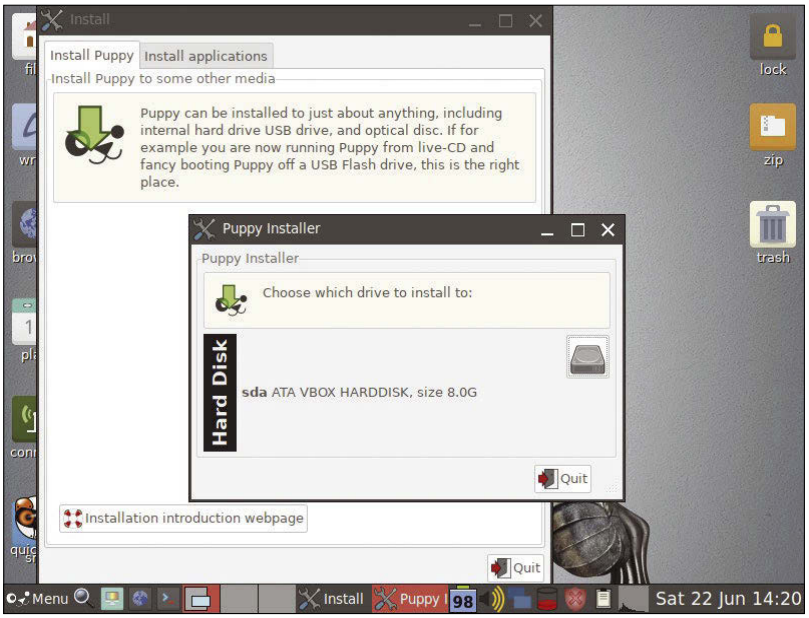


Figure 4: Installing Puppy Linux onto a hard disk is a breeze and completes in a matter of minutes.

Installing New Applications

New software can be installed from the command line, Package Manager, or Quickpet Bionic 64 (Figure 3). Quickpet is particularly handy if you are interested in downloading from a selection of the more popular apps available to Linux. While the selection is fairly modest, users can find some excellent packages for most tasks. Quickpet does not have the variety of the Ubuntu Software Center, but the basic principles are the same. From here you can download top Linux apps like Thunderbird and LibreOffice. If you're dissatisfied with Pale Moon, you can also install the standard Mozilla Firefox browser or even Chromium.

If your program of choice isn't available via Quickpet, you can launch Puppy Package Manager via the *Setup* shortcut on the desktop. While the selection here is smaller than that offered by bigger distros like Ubuntu, it does make more programs available than Quickpet. For instance, while Quickpet only contains a few generic games platforms in its *Fun* category, you can use Puppy Package Manager to install the ever popular Sunfish Chess.

Although Puppy Linux is based on Ubuntu, it's an operating system in its own right. This means that traditional `apt-get` commands won't work in Bionic's Terminal. Technically you can use the Terminal to manually download and compile programs that are unavailable via Quickpet or Puppy Package Manager, but you'll still need to use the Package Manager to install any dependencies.

Some gifted developers have compiled their own Puppy-compatible versions of apps into installable PET files. Many of these are available via the Puppy Linux forum [5]. Install at your own risk. The lack of support for `apt-get` also means that users can't update or upgrade Puppy in the same way as Ubuntu. The easiest and fastest way to update the OS is by installing new ISOs as they become available.

Installing Puppy to a Hard Disk Drive

Puppy's minimal paw print combined with its special write-caching method on flash drives makes it suitable for installation on a wide variety of media (Figure 4). If you're booting from a Live CD or USB, click the Install icon to begin setup. You'll be offered a choice of the more traditional Universal Installer, which is designed for hard drives, or BootFlash USB Installer, which is perfect for USB devices. To install Puppy Linux to your machine's HDD/SSD, click *Universal Installer*. Select your drive type from the list available, and then choose the

specific installation disk. The OS supports a number of drive formats including FAT32 and NTFS, but for best performance use a drive formatted specifically for Linux (e.g., ext3). If your drive has no partition table, the installer will prompt you to launch GParted to format and make it bootable. Press *OK* to confirm your install location.

Setup offers two installation methods. The first, Frugal, will install a bare minimum of apps to get you started. The Full installation is self-explanatory. Whichever you choose, you can always add more programs later via Quickpet or Puppy Package Manager. If you choose a full installation, you can opt to use an initial ramdisk. This is an initial root filesystem, which is mounted prior to the real one. An in-depth technical explanation of this feature is available via the IBM website [6], but as the installer explains, this offers greater speed and stability over installing everything to disk.

Paws for Thought

Puppy's main strength lies in its ease of setup and use. If you're looking for a lightning fast and simple way to recover files from a booted hard drive, there's no better alternative: The OS is optimized to run in RAM, so it will be ultra-efficient at recovering files.

Annoyances may begin to creep in for Linux users who are used to a more flexible and customizable OS. Puppy geeks must content themselves with the default suite of apps or those available via the Package Manager, unless they fancy some hardcore compiling.

Installation to hard disk is an absolute breeze, but the OS does need some tinkering, such as adding a more powerful office suite, before it can become a fully-fledged desktop OS. Given its lightweight nature, it's surprising that there's no officially maintained port for ARM devices like the Raspberry Pi, as Puppy Linux would otherwise run perfectly well on a small-board computer.

While updating and upgrading Puppy Linux isn't as simple as other Ubuntu-based distros, this is balanced out by an extremely quick setup process. There's also an excellent help guide and community forum available for users who run into difficulties. If you need a quick response, use the excellent preinstalled app HexChat to connect to Puppy Linux's IRC channel [7]. ■■■

The Author

Nate Drake is a freelance journalist specializing in cybersecurity and retro tech.

Info

[1] Puppy Linux: *http://puppylinux.com/index.html*

[2] Etcher: *https://www.balena.io/etcher/*

[3] Woof-CE: *http://puppylinux.com/woof-ce.html*

[4] Puppy Linux wiki: *http://wikka.puppylinux.com/HowPuppyWorks?redirect=no*

[5] Puppy Linux forum: *http://murga-linux.com/puppy/index.php*

[6] Initial ramdisk: *https://developer.ibm.com/articles/l-initrd/*

[7] Puppy Linux ChatRoom: *http://wikka.puppylinux.com/ChatRoom*

FOSSPicks

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

In the year of Blade Runner, Graham is hoping that the imminent release of Valve's Index virtual reality headset and controllers, with Linux support, will let him port KDE to virtual reality. BY GRAHAM MORRISON

Music-making application

LMMS

The great thing about Linux Music Making Studio (LMMS) is that it doesn't have a steep learning curve, and it can make music production fun, even when you have no idea what you're doing. LMMS is able to do this by also being capable of professional results, albeit in an electronic or loop-based pop genre. LMMS has been around for a long time, but without a major release, it's been difficult to justify its addition here. Fortunately, this changed with the release of LMMS 1.2.0, its first major update in four years.

In many ways, the LMMS workflow is very similar to old versions of an incredibly popular Windows application called FruityLoops Pro. Your music's basic building block is a pattern. This can be constructed out of a rack of drum or percussion sounds being triggered from a matrix, from a single sound in the rack being programmed from a MIDI keyboard, or from the onscreen Piano Roll editor. These are all very standard ways of working with notes, and LMMS makes them immediately playable and clickable as you load a template and start adding and removing notes. When you have a pattern you like, you can copy, paste, and edit to make some variations before using the Song Editor to sequence in which order, and when, each pattern is triggered.

1 **Side panel:** Quickly access your files, presets, and the sound library that comes with LMMS. 2 **Song Editor:** After you've created a few patches, you can sequence them and add audio tracks. 3 **Pattern editor:** Trigger samples or synth sounds, perfect for creating drum loops and basslines. 4 **Real-time effects:** LMMS can access LADSPA and VST effects and includes a comprehensive set of its own effects. 5 **Synths:** Alongside effects, LMMS bundles a wide variety of synths, including a SID emulation and its own ZynAddSubFX front end. 6 **Piano Roll:** Add or record MIDI notes for synth and sound playback. 7 **Mixer:** Route audio to effects groups, automate controls, and export high quality audio.

Each instrument has its own uniquely configured sound engine that can be controlled just like a sampler or synthesizer. With the sampler and a few samples, you can use LMMS to create all kinds of sounds, from modern percussive loops to retro strings with modulated filters. You can play and edit at the same time and program the notes and chords as you go along, automating changes in the sound, volume, and panorama in the Song Editor. You can even record or import your own audio, just as you would with a traditional digital audio workstation (DAW) like Ardour.

All of this "composing" can be done while playing in real time from your Linux keyboard, MIDI keyboard, or remote control, while adding effects, new instruments and controls, building patterns, and constructing your song until you have a track you like that you can then export. New output options include 24-bit WAV for professional mastering. LMMS does all this from a wonderfully designed, completely integrated user interface (UI). It even includes its own effects with beautifully responsive graphics, such as the all new parametric equalizer and spectrum analyzer. These can be used alongside your native Linux effects and software synthesizers. There's even a good range of templates and sounds to get you started, a new demo for this release, and some good documentation that covers the basics.

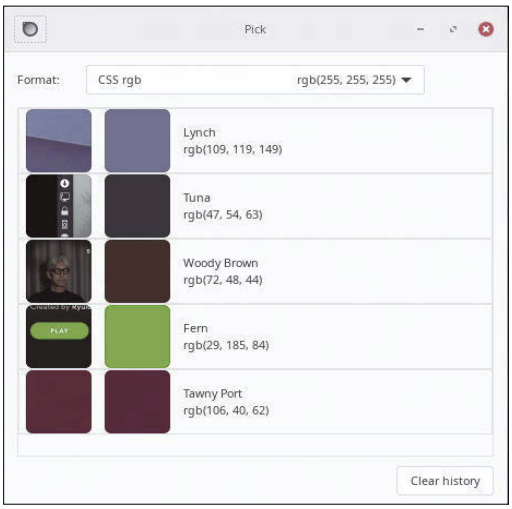
Project Website
https://lmms.io/

Color picker

Pick

Even when you don't work with color as part of your profession, color is obviously vitally important. And yet, on our computers, it's seldom given much thought. Unless you're in a creative profession, playing or editing images and video, you often leave the onscreen colors to content producers and theme editors. Part of the reason for this is that editing color, and color information in general, is relatively inaccessible. Each desktop will typically have its own palette, its own color selectors, and, if you're lucky, even its own names for colors. If you're unlucky, a color is little more than a hexadecimal value, and where and how you change colors or copy them is usually as opaque as a midwinter snowstorm. Which is why Pick is such a useful tool.

Pick is one of those applications that solves a small issue so effectively you wish more applications would do the same rather than trying to cram in every feature under the sun. It's a GTK+ 3 application that integrates perfectly with any desktop. It particular, its top-left "picker" action icon is positioned perfectly. Click this and a perfectly rendered circle is overlaid onto your desktop, with the interior portion of the circle showing a pixelated magnification of an area beneath. A pixel-sized select helps you choose the pixel of the color you want to select. After doing that, the main window registers the color with a swatch of the color itself alongside a thumbnail showing from where the color came. To the right of these, you get the RGB value for your color – and cleverly – a text description, such as Tuna, Lynch, Fern, and Tawny Port. The RGB can then be copied to other apps, used in CSS, or used anywhere else you need access to color.



Pick is an example of brilliant design.

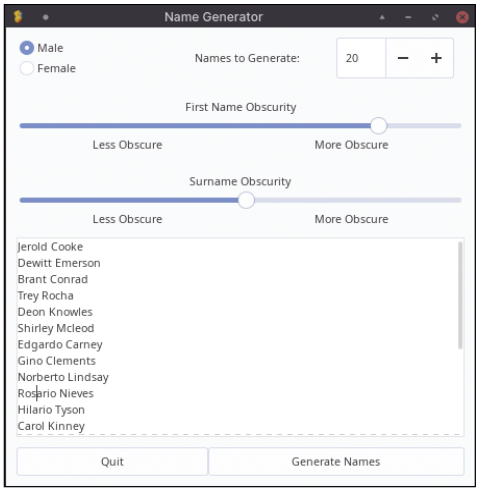
Project Website
https://kryogenix.org/code/pick/

Writer's aid

Name Generator

If you've ever been responsible for choosing a child's name, you'll know how difficult it can be (unless you're calling them $YOURNAME III). It's a huge and impossible task, and you'll never know whether you made the right choice, especially as you're likely to keep quiet if you feel you have made the wrong one! If you're a writer, you encounter a very similar problem when creating fictional characters for a story. Those characters need to live with their names, at least within the confines of your story, and the writer often brings a huge bias that isn't easy to overcome when you're trying to name a personality that isn't supposed to mirror that of the author. When a book might typically have 20, 30, or more characters, it can become a real stumbling block to progress.

Which is where the super simple, super easy to use, and super useful Name Generator can help. With just a couple of sliders, a selector for *Male* and *Female*, and a number for the amount of names you want, you can generate a huge variety of names for your fictional characters. The two sliders let you make either the surname or the first name less obscure or more obscure. Less obscure first names include Susan, Henry, Sandy, and Kevin, while moving this to the right returns names like Luna, Cedrick, Hildred, and Colton. It's obviously a very simple project to create and makes an ideal first foray into GTK+ development, but it's also very useful for those that need a name generator. The names themselves are from public US Census data, and they're simply stored in three text files. This means that even non-programmers can fine tune the names or the name generator to their liking, making this an ideal first project both to study and augment.

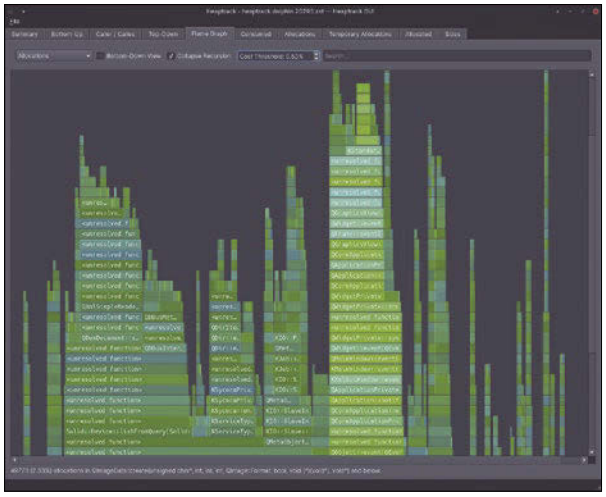


For an application that generates names, we're surprised Name Generator doesn't itself have a better name.

Project Website
https://github.com/jsseidel/namegengtk

Memory profiler

Heaptrack



This is a profile in Heaptrack of the Dolphin file manager browsing and then opening an image file.

Some time ago, we looked at an app called hotspot that operated as a GUI for a complex debugging application called `perf`. `perf` and hotspot let developers look at the internal workings of the applications they're working on to see where they are spending the most time and resources, helping developers improve the performance of that application and track down bugs. Heaptrack performs a similar role, only with a focus on memory allocation rather than CPU resources. If you're a developer, it can help you see when and where your application or utility is allocating and using memory, when and whether that memory is being freed correctly, and which elements within your code are asking for the most memory. Like `perf`, it's a complex process that requires a thorough understanding of the operating system and the application you're trying to profile, but due to Heaptrack's graphical nature, it's also a fascinating window into your applications, whether you're a developer or not.

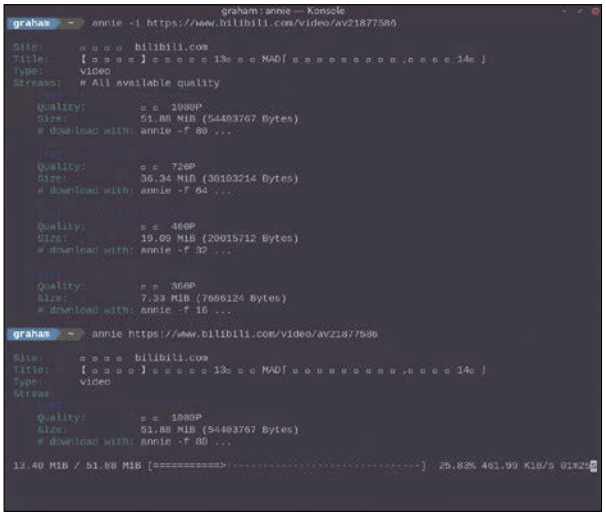
Heaptrack operates in two modes. The first mode is used to generate a profile for the running utility or application and runs through `gdb`. The result will be a file that you pass as an argument to launch Heaptrack in its second mode, which is where you can explore your application's memory allocation through a lovely GUI. There are 10 tabs in this view, all offering a slightly different take on how memory was used during the profiling process. The first tab shows a summary of which calls used the most memory, leaked the most, or required the most temporary storage, but the more interesting tabs use this information to visualize usage over time. With *Allocations*, for example, you can see how memory use grows during execution. The *Flame Graph* is our favorite, though. It groups calls into a hierarchy, with calls calling calls stacked upon each other into mountains of memory use – fascinating!

Project Website

https://github.com/KDE/heaptrack

Video downloader

Annie



Annie is considerably better, and more useful, than either of its namesake films.

There are many reasons for pulling content from an online video, and many ways to do it. One of our favorites is `streamlink`, for example. `streamlink` allows you to stream content from a remote source, such as YouTube or Twitch, into an application like VLC, without first downloading or opening a compatible web browser. It also lets you incorporate those remote sources into a digital video recorder, or a different front end, giving you access to web content without the usual side-panel distractions and comments that typically flood sites that stream content. However, Annie is a new option that promises to be a "fast, simple, and clean video downloader built with Go." And thanks to its dependency on Go, it can usually be installed via a simple `go get` command.

Like `streamlink`, you simply need to provide the command-line utility with an accessible URL. But there are many additional arguments you can add to fine-tune your download. Adding `-i`, for example, will output detailed information about the video to which you're linking, including the quality and resolution options. You can then specify one of these with the `-f` argument to download a video at the exact quality you require. After you start a download, you see an informative progress bar that provides all these details on the stream you've selected, helping you to be sure you've got the correct URL. There are plenty of additional arguments, too, including options for downloading captions, downloading an entire playlist of videos (great for binge watching on a flight), and setting proxy access to help you get around geographically locked content. It works seamlessly and is probably quicker to use with the average download than waiting for the ads to load on the site you're wanting to access.

Project Website

https://github.com/iawia002/annie

Terminal email

aerc

The command line is still commanding lots of attention, with many technical users looking to use the terminal more to absolve themselves of Slack and web-based distractions. Of all the applications that could be difficult to replace on the command line, though, email isn't a difficult challenge. This is because email effectively predates the desktop, and the same solutions that used to work on your old SPARC Solaris still work on your hex core Ubuntu 19.04. Mutt is still particularly popular, for example, and a great way to access your email from the command line or via SSH. In many ways this new terminal email client, aerc, is after Mutt's crown by attempting to implement more efficient and reliable networking, a keyboard shortcut system that's more like Vim, and an embedded terminal that lets you both read and reply to different emails at the same time. Combine this with some unique features, such as a filter that lets you review patches, and you have a compelling argument for giving it a go (that's a pun – aerc is written in Go).

All this newness is evident from the first launch that includes a very atypical command-line configuration wizard for the email account you want to use, plus a walk-through manual page to help get you started. After that, it takes only a few moments for your remote IMAP server to push all the subjects and details to your inbox, which appear almost instantaneously.

No amount of command-line coolness can save you from the spambots.

You navigate them using Vim's navigation keys, pressing *Enter* to open an email. Cleverly, the open email isn't in a new pane, but rather a terminal emulator running `less` with filters for adding color highlighting and rendering HTML with the W3F utility. This utilitarian approach is extended to writing emails, which uses Vim by default, and it's brilliant at being able to fire up your favorite editor to send emails in its native environment.

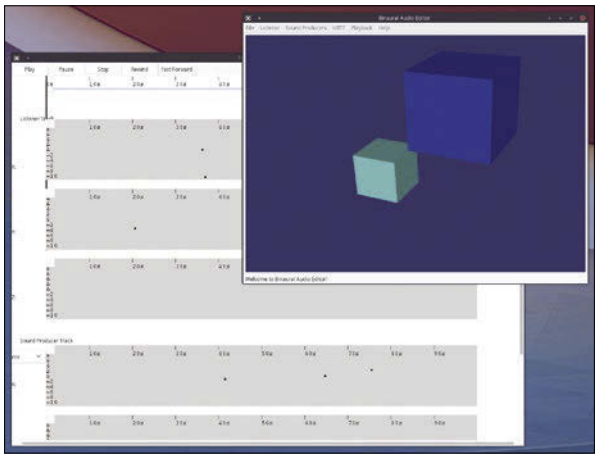
Project Website
https://aerc-mail.org/

Sound processor

Binaural Audio Editor

If you've not listened to binaural audio, you really need to give it a try. It's audio that's usually been recorded to mimic the position and dispersion of your ears with a chunk of dense matter between two microphones placed a few inches apart. Recordings need to be listened to through headphones, and a good recording can sound frighteningly realistic, often fully recreating the 3D stage of the recording environment. We say binaural recordings are "usually" recorded with two microphones, because the effect can be simulated in software, as can the 3D playback, and there are even open source plugins that let you create binaural audio from a mono audio file. Which is exactly what Binaural Audio Editor does.

The application consists of two main windows. The first is a 3D view of a cube. This represents you, the listener. You then need to add something called a sound producer. This is where your audio file will be played back from, and when its location isn't the same as the listener, it's represented in the 3D view by a turquoise cube in the same view. The position of both the listener and the audio producer can be changed from a couple of pop-up panels, but the clever part becomes evident in the second window. This lists *x*, *y*, and *z* values across a timeline for both the listener and the sound producer. You can click within any of these timelines to virtually change the position of your chosen object at that point in time.



Use the *x*, *y*, and *z* timelines to generate real 3D audio and dynamically change the position of both the sound and the listener.

When you then play back the audio from the other window, the 3D location of both the listener and the sound source will change dynamically. You can't yet save this output, although it's easy to capture it, and the application itself is prone to crash. But the application is still new and in rapid development, so it's going to be a great project to follow.

Project Website
https://github.com/adct-the-experimenter/binaural-audio-editor

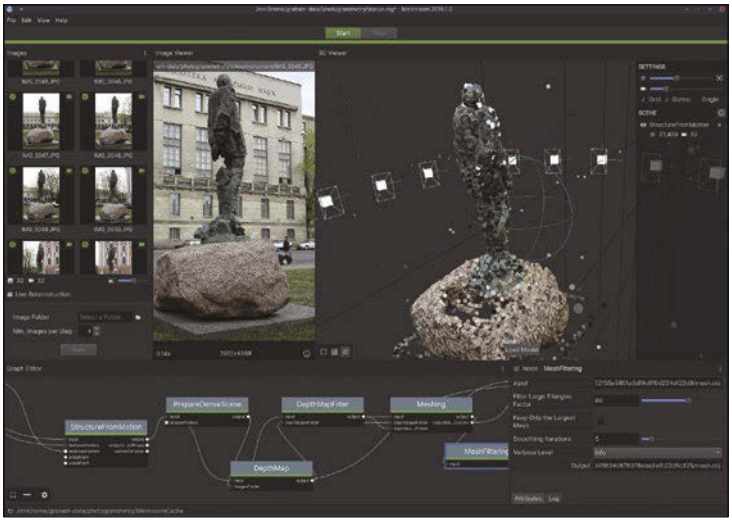
3D reconstruction

Meshroom

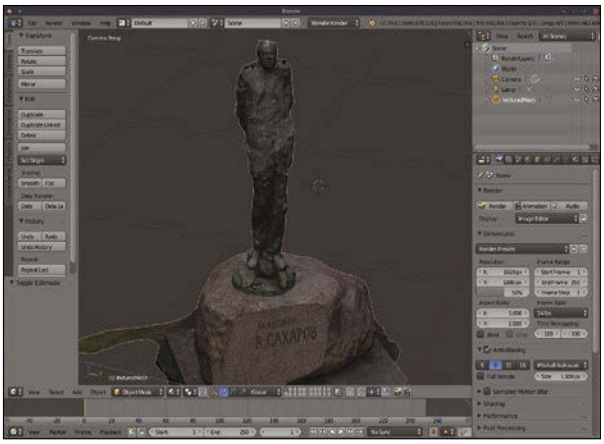
Photogrammetry has been around for as long as there have been photos. It's the process of taking measurements from images and using those measurements to reconstruct models of the items and spaces within the images. But modern photographic and processing technology has taken this process to a whole new level, allowing you to build 3D textured models from a carefully taken collection of 2D photos. To reconstruct a scene, for instance, you need to take hundreds of shots with similar exposure, focus, and range, while making sure all elements within the scene are covered from multiple angles. The same is also true for scanning objects, although physical props like a turntable can make it considerably easier to attain good results. And the results can be incredible. Entering a reconstructed scene in virtual reality, for example, lets you wander around an exact to-scale recreation of a cathedral, a museum, or mountain top, captured perfectly for all time. Similarly, there doesn't seem to be a modern 3D game that hasn't incorporated photogrammetry for its models and locations, and it's likely to be a huge growth area in the future as our photos and Raspberry Pis become ever more capable.

But much of the software used to turn photos into 3D is proprietary, with RealityCapture and PhotoScan being two popular choices. There is an open source pipeline, using VisualSFM, MeshLab, and Blender, but it's difficult to use and takes plenty of work to create good results. Which is why it's brilliant to see a new piece of open source software in the 3D space, which is exactly what the excellently named Meshroom is. The application itself is easy to understand. This is because it's still only in the early phases of development, but also because of great design that doesn't overwhelm you with options. You start by importing your stack of images, which can then be refined to make sure you're making the best use of the available data. You can then just click *Start* to initiate the intense processing required to build the 3D mesh.

If you've imported hundreds of photos, which you need for a complex scene, processing can take hours and tens of gigabytes of memory. Simpler models can be generated from a dozen and 20 minutes of processing, but there is an inescapable requirement in the current version. You will need an Nvidia GPU and CUDA installed, as this does all the number crunching. Other GPUs are being evaluated, so hopefully more options will become available. Either way, even though processing can still take hours, the 3D view will become populated by 3D colored points as they're derived from your image set, along with triangulated viewports for the location where each photo was taken. The graph editor lets you select and edit each step in the process, and, cleverly, each step logs its own output to a separate folder, complete with any generated assets, such as depth maps or objects. This means when you change something small, these elements won't need to be calculated again. The final result can then be imported into your modelling or mesh reduction application, either as an object, as a full texture, or both.



For best results, take your photos in RAW mode and lock the exposure. If possible, use a DSLR with low depth of field.



Meshroom's output is a fully textured model that can be loaded into Blender and refactored and refined for final use in your scene or game.

Project Website

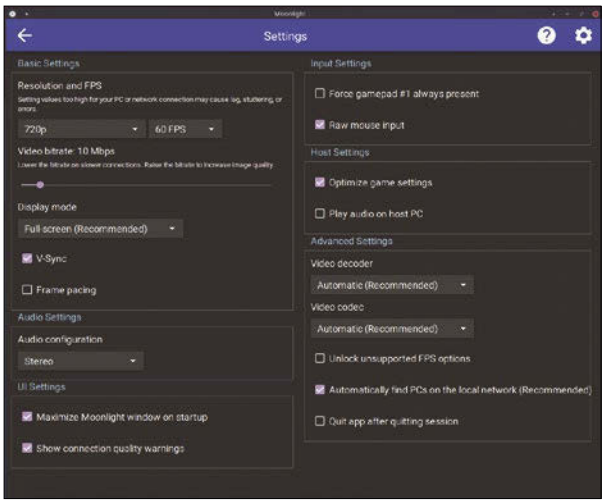
https://github.com/alicevision/meshroom

Game streaming

Moonlight

If you're into games, you may have heard of Valve's low cost Steam Link hardware. This was a wonderful little embedded (proprietary) box that on one side is connected to your network – ideally, via Ethernet – and on the other side is connected to your television and a games controller. You could then stream your games from a game-capable PC running Steam to the television via Steam Link, letting you play your favorite games from anywhere with good network connectivity. The most impressive part of all this was that it actually worked remarkably well, even at 1080p and 60 frames per second. There was no noticeable latency, allowing you to free yourself of an often noisy and constrained PC environment to play games on the sofa.

As you've likely guessed, the past tense of the first paragraph means that Valve's Steam Link is no more; Valve stopped selling them in late 2018. To mitigate this, alongside iOS and Android clients, Valve released its Steam Link proprietary streaming protocol via an app on the Raspberry Pi, helping you replace the old hardware with a newer and cheaper option. It works well, but it doesn't integrate well with other software, such as Kodi, and doesn't have an x86 build. If you want the same facility on your Linux box, you need to install, sign-in, and run the entire Steam stack. Which is where Moonlight comes in. Moonlight is an open source client for Nvidia's GameStream, another proprietary service for Nvidia's hardware that does the same thing Steam Link did, except Moonlight is better. It has an even lower latency, supports 4K resolutions and 120fps, and works on many different devices. It integrates well with Kodi (with an add-on), a Raspberry Pi, x86, and even an unlocked PlayStation Vita. Because it's open source, it can be easily built and run on almost any Linux system.



The main app has a Qt UI, but a separate version, Moonlight Embedded, runs headless for media centers.

Project Website
https://moonlight-stream.org/

Jigsaw game

Palapeli

There aren't many desktop games you can play at your own leisure, perhaps in between work or across days. Chess could almost work if you didn't have to cache your current position and your strategy in your brain before coming back to a game, and there are both text and graphical adventures with a very leisurely pace. But nothing quite matches a good old jigsaw. There's something about their simplicity and their level of distraction that can be very calming, and ironically, almost productive. Palapeli is a KDE-based desktop application that lets you solve a jigsaw puzzle from your desktop. You start a game by selecting an image from the installed collection. After doing this, the play space appears, littered with the jigsaw pieces you're going to need to fit together to reconstruct the image. There's lots of control over the appearance of the play space, and you can add 3D beveling to the pieces, drop-shadows, and replace the background, all of which we'd recommend for maximum legibility. You then drag pieces around to find the places where they fit, but you're free to use whatever strategy works best for you. When you get two fitting sides in close proximity, they'll snap together, but even this can be reconfigured.

By default, there's a clear area in the middle of the play space where you might start constructing edges, before moving them into position. At any time, you can select other pieces and rearrange them around the jigsaw, much as you would in real life. You can drag the play space with the right button and freely zoom in and out to get a better view. There's also an optional preview window to see the original image and a progress indicator. When you quit the application or go back to the image collection, your progress and play-space settings are automatically saved and restored again when you return.

As well as letting you solve built-in puzzles, you can use Palapeli to generate all kinds of new ones from many different piece shapes.

Project Website
https://github.com/KDE/palapeli

Making the most of WordPress

Blog Central

WordPress is one of the most popular content management systems. With the introduction of the new Gutenberg editor, now is the time to (re)learn WordPress.

BY MARCO FIORETTI

In 2018, WordPress [1] powered about 75 million websites [2]. WordPress knowledge is a valuable job skill, more future-proof than any "social media" expertise and generally the best answer when friends and family ask you to help them make a website.

When it was released in December 2018, WordPress 5 generated enough complaints and discussions to remind one of the "KDE vs. Gnome" feuds. Now, however, many tools have been ported to the new version, and many teething problems have been fixed. Learning WordPress 5, even if you are new to WordPress, is much easier now than it was six months ago. In this tutorial, I will describe how to install WordPress 5 on Linux, configure its main features, and use its new editor, Gutenberg, plus I'll highlight some lesser-known WordPress tools.

Installation

While WordPress boasts a five minute install, this will take a bit longer if your system doesn't have all the correct dependencies installed. Starting from scratch, the complete procedure to self-host WordPress on Linux requires the following steps:

1. Install and configure a recent version of the Apache or Nginx web servers.
2. Install and configure PHP v7.2 or higher.
3. Install MariaDB.
4. Create a WordPress database and database user.
5. Configure your web server to point to the WordPress folder.
6. Download and unpack the WordPress archive in the right folder, with the right permissions and ownership.
7. Point your browser at your website's URL to perform the five minute installation.

The last two steps do indeed take just five minutes and are all you need to install WordPress on any hosting service that officially supports it and provides a MySQL or MariaDB database.

To self-host on a barebones Linux server, you'll need the first five steps; this only takes 10 more minutes – unless you run into the same problem I encountered on my CentOS 7.6 server (which also may occur with other Linux distributions). Luckily, the solution is simple and distribution-independent.

The problem is that WordPress 5.x requires Apache v2.4 and PHP v7.2, which may be newer than the versions available in your distribution's default repositories, and adding optional ones may not be enough.

In my case, I had no problem enabling the EPEL repository for CentOS 7.6 and installing those packages from there. But when I pointed my browser to *fabriziofioretti.com* [3] after performing steps 3 to 6 above, Apache told me that the folder I wanted did not exist or was not accessible. The reason is that an Apache server only sees folders *below* the one defined as `DocumentRoot` in its own configuration file. I had put the WordPress files inside `/var/www/html/fabrizio`, but `DocumentRoot`'s default value in the EPEL Apache 2.4 package is `/opt/rh/httpd24/root/var/www/html`. Once I set that variable to `/var/www/html` and restarted the server, everything was fine.

To create the database in step 4, launch the command-line `mysql` client as root in a terminal window:

```
#> mysql -u root
```

and then, at the SQL prompt, type the following:

```
#> create database wp_database;
#> grant all privileges on wp_database.* to ➥
   wp_user@'localhost' identified by ➥
   'wp_password';
#> flush privileges;
#> exit;
```

changing `wp_database`, `wp_user`, and `wp_password` to suit your needs.

Step 5 (configuring Apache to serve the new website) was almost as fast. Thanks to the Let's Encrypt project [4], I created a self-signed SSL cer-

tificate for the domain with one call of the `certbot` command:

```
#> certbot -n --agree-tos --standalone certonly
  -d fabriziofioretti.com
```

and then added to Apache's `httpd.conf` file the `VirtualHost` section shown in Listing 1. Lines 1 to 4 tell all browsers connecting to *fabriziofioretti.com* through the default, unencrypted HTTP port (80) to use the secure HTTP protocol instead (HTTPS). Lines 6 to 9 tell Apache to use the files inside `/var/www/html/fabrizio` to handle all secure connections (on port 443) to the same website. Line 11 through line 15 specifies where to find the cryptographic certificates and keys created with `certbot` just for that purpose. Finally, line 17 sets the location of optional access control rules.

To copy the WordPress files on the server, unpack them, and change their ownership to the *apache* user running the web server, use the following:

```
#> cd /var/www/html/
#> wget https://wordpress.org/latest.tar.gz
#> tar xvf latest.tar.gz
#> mv wordpress fabrizio
#> rm latest.tar.gz
#> chown -R apache:apache fabrizio
```

Housekeeping

A word of warning: The WordPress installer automatically creates an administrator account. Never use this account to create content, even if you are sure you will be the website's *only* author. Instead, give the administrator account a hard to guess name and only use it for actual administrative tasks. Then create a separate account immediately after installation for publishing and managing content; give that account your real name and email address, but not administrator privileges! First of all, this makes brute force attacks on the most powerful account a bit harder than if it were named *admin*. Secondly, it will decrease the probability that you inadvertently will change some setting while using your author account.

After setting up accounts, go to *Tools* | *Site Health* for critical information about your WordPress setup (Figure 1); check out the status report and follow its advice. Repeat this procedure every week or so.

Getting Started

Before jumping into the WordPress dashboard, sketch out *on paper* your website's basic components and how visitors should access them, including things like menus, a search field, RSS news feeds, or a slider. Determine whether you want a home page with static content or a list of posts, with the newest first. Do you need a banner or a background image? Will you accept reader comments and for how long? The better you do this initial homework, the happier you will be with your finished product. However, do not worry about design or where to place those components at this stage.

Now you are ready to use WordPress' dashboard. Configure the components you have decided upon by going through the sections in the dashboard's left panel in order (with the exception of *Appearance* | *Themes*, as you can waste a lot of time checking out the various theme options before configuring the main components).

Listing 1: Configuring Apache

```
01 <VirtualHost fabriziofioretti.com:80>
02   ServerName fabriziofioretti.com
03   Redirect permanent / https://fabriziofioretti.com/
04 </VirtualHost>
05
06 <VirtualHost fabriziofioretti.com:443>
07 ServerAdmin mfioretti@nexaima.net
08 DocumentRoot /var/www/html/fabrizio
09 ServerName fabriziofioretti.com
10
11 SSLEngine on
12 Include /etc/letsencrypt/options-ssl-apache.conf
13 SSLCertificateFile /etc/letsencrypt/live/fabriziofioretti.com/cert.pem
14 SSLCertificateKeyFile /etc/letsencrypt/live/fabriziofioretti.com/
     privkey.pem
15 SSLCertificateChainFile /etc/letsencrypt/live/fabriziofioretti.com/
     chain.pem
16
17 AccessFileName .htaccess
18
19 </VirtualHost>
```

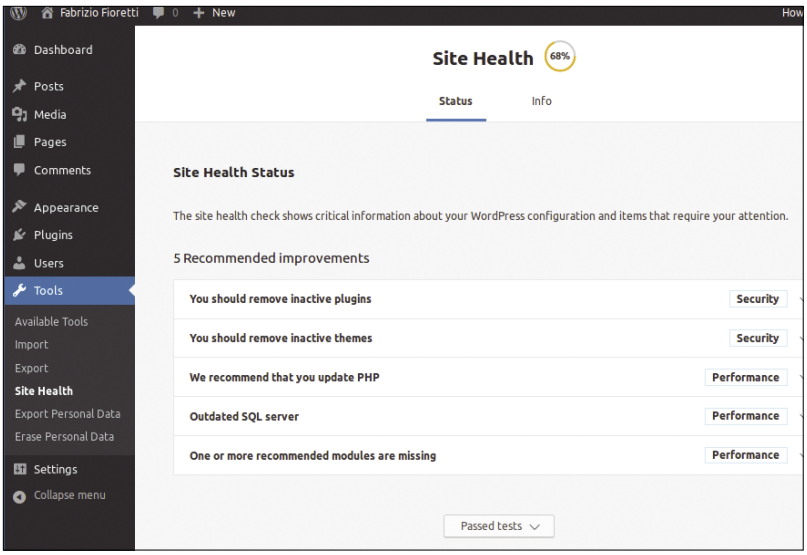


Figure 1: The Site Health Status report makes recommendations for improving your site security.

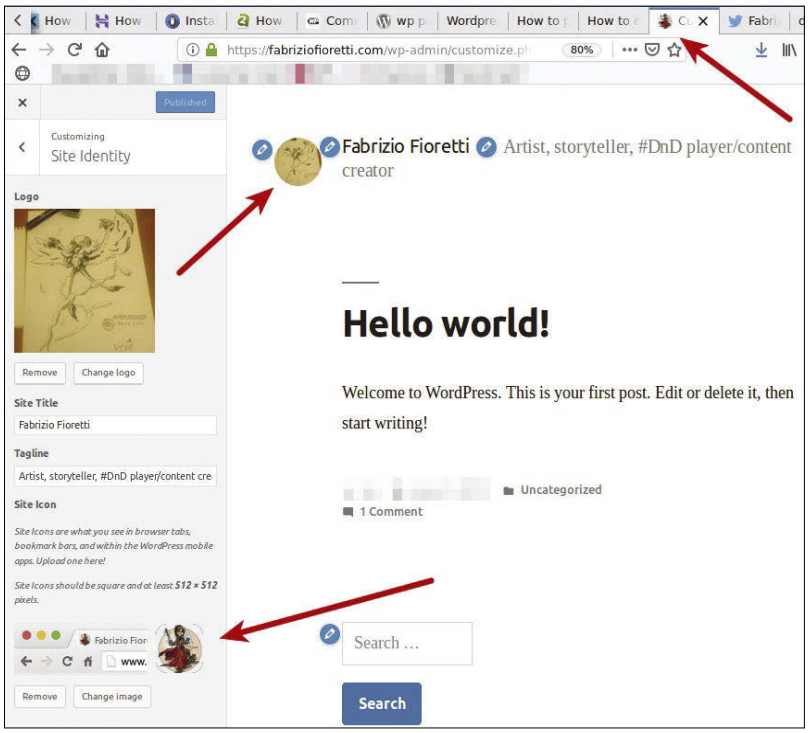


Figure 2: You will want to customize the logo and site icon.

The first components that you will want to customize are the logo and site icon, under *Appearance | Customize* (Figure 2). The logo is what visitors see (depending on the chosen theme) somewhere in each page's header, and the site icon is the small picture that identifies the browser tab that contains your website pages.

Next, define your homepage settings. As your homepage, WordPress can display the titles and abstracts of all your posts, in reverse chronological order, or a fixed, static page. In WordPress, pages should contain permanent, timeless general information (e.g., your mission or how to contribute), whereas posts are time-sensitive content (e.g., travel reports, announcements, etc.).

Menus, essential to navigating any website, are easy to create and rearrange in WordPress (Figure 3). You can also configure WordPress to automatically add each newly created page to the menus. Keep in mind that menu placement depends on the theme, and not all themes offer the same options.

After its plugins and themes, the most flexible (and easy to abuse) WordPress building component is widgets, which are boxes that can contain static or dynamic content, from greetings or embedded videos to tag clouds and titles of most recent posts. Figure 4 shows WordPress's default widgets, but other plugins and themes add more options.

Clicking on *Settings* in the dashboard loads the interface to configure variables like website URL, language, membership, date format, comment management, and more. An often overlooked part of these settings is *Permalinks* (the URL structure of each page or post of your website). You can tell WordPress to automatically include in each URL its category, author name, or the date, in several formats. Good permalinks improve the usability of your website and above all its forward-compatibility. Permalinks with a structure like `mywebsite.com/YEAR/MONTH/post-title`, for example, are supported by any CMS. Adopt that structure, and none of the links to your posts will break, even if you decided to replace WordPress with some other product. For more information on how WordPress handles permalinks, see [5].

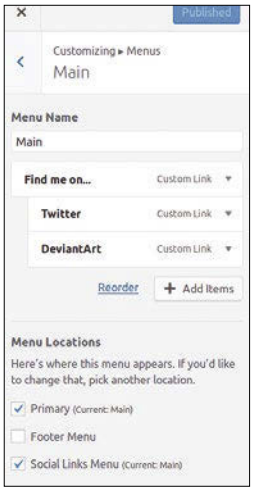


Figure 3: Building WordPress site menus is a snap, but remember that their location depends on your chosen theme.

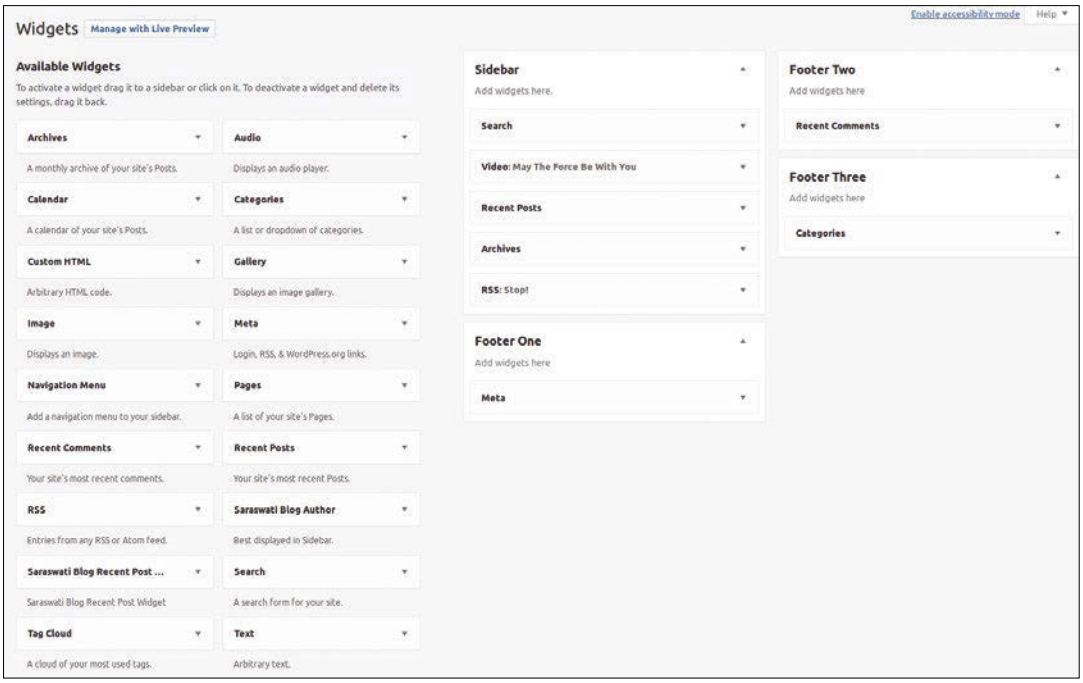


Figure 4: WordPress 5 widgets facilitate adding static or dynamic boxes to your pages.

Test Post

Once you have configured all your components (minus the themes), publish a test post. The content doesn't matter; just make sure that it contains headers, lists, at least one image, and possibly even an embedded video. Then, verify how that test post and the website structure (not the design) works with the default WordPress theme. If everything is in place, then you can move on to visual appearance.

Finally, Themes!

A website's perfect visual theme, in WordPress or anything else, is whatever maximizes its usability while satisfying the taste (or company policy) of its webmaster. Because WordPress makes it so easy to download and switch themes, I have one piece of advice: See how a theme looks on many different operating systems and devices (from smartphones to big monitors) as well as how quickly it loads. Then, choose a theme that has an open source license *and* the greatest number of users. This will decrease the possibility that development stops before you get tired of the theme. Finally, choose a theme that is already fully compatible with WordPress 5's new editor, Gutenberg, because posts using such themes look exactly the same in Gutenberg as they will look when published.

Enter Gutenberg

Gutenberg, introduced with WordPress 5, replaces TinyMCE and is responsible for most of the complaints. If you prefer TinyMCE, just run the Classic Editor plugin – until 2021, that is, when you will have to use Gutenberg anyway.

If you have never used TinyMCE, Gutenberg won't be a big deal, but it has been described as "the biggest transition in WordPress history." Gutenberg's goal is to make writing in WordPress as productive and pleasant as it is in platforms like Medium or Ghost. In Gutenberg, for example, you can work in Fullscreen or Spotlight mode and generally have much more control of your content's layout. Clicking on the wheel in the top-right corner opens tabs with all the options and settings for the entire post or the part you are currently editing.

Unlike TinyMCE, which handles each post or page as one monolithic entity, Gutenberg is a *block* editor (the source of most complaints). In Gutenberg, headers, paragraphs, buttons, tables, lists, videos, images, e-commerce, or polling widgets are each a separate block that you can see and format independently from other blocks. Also, you can drag and drop blocks to change each post's layout.

To add a new block, just click on the plus sign in the main window and choose its type. If you dislike clicking, type a slash, followed by the first letters of the type's name, and an autocompletion function will help you find and add that type block quickly. Access the full list of Gutenberg keyboard shortcuts by typing Shift+Alt+H. I strongly suggest bookmarking and printing WordPress 5's Ultimate Cheat Sheet [6].

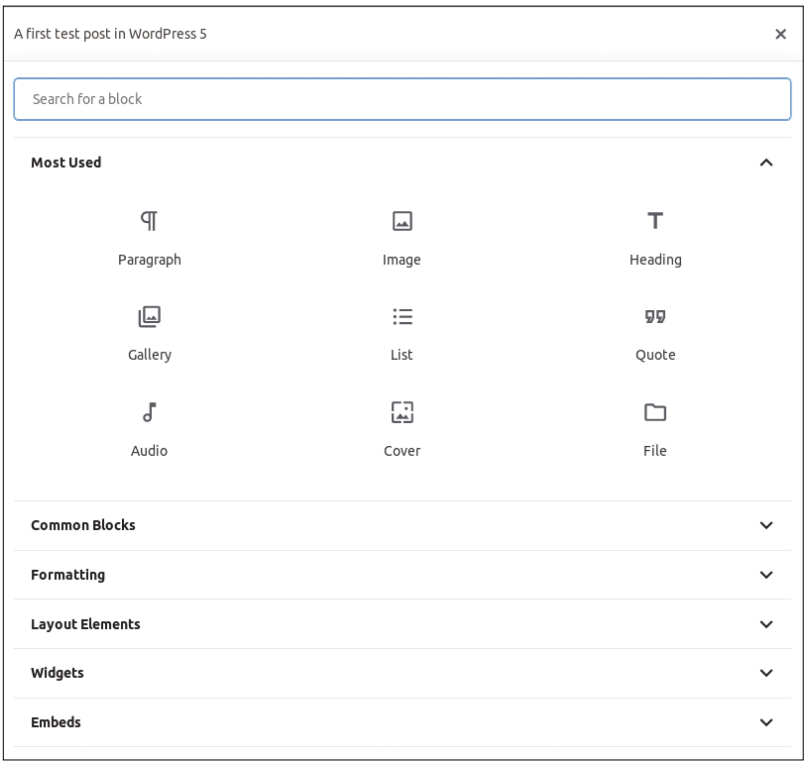


Figure 5: Gutenberg's nine "most used blocks" are enough for 90 percent of its users.

In practice, most authors just need the *Most Used* or *Common* blocks (Figure 5), but many more are available. You will find tables and pull quotes in the *Formatting* section, and buttons, separators, and columns in *Layout Elements*. Use *Widgets* for short codes and "meta components" like the latest posts or categories. To embed video, tweets, and other similar content from external sources, see *Embeds*.

You can embed images or video clips previously uploaded in your WordPress Media Library or just type the URL (if they are publicly accessible online). If you already publish content on Vimeo, YouTube, Dailymotion, or Flickr, no typing is necessary: Install and activate the Remote Media Libraries plugin and give it your user ID for those services. You then will be able to see the list of all the corresponding content right inside WordPress and embed them with one click.

The most popular source for additional blocks is the Otter Blocks plugin. Among other things, it includes social media sharing icons and a block to embed Google Maps.

A post consists of three blocks: a header, a text paragraph, and an image. When you select a block, Gutenberg activates a toolbar with the main

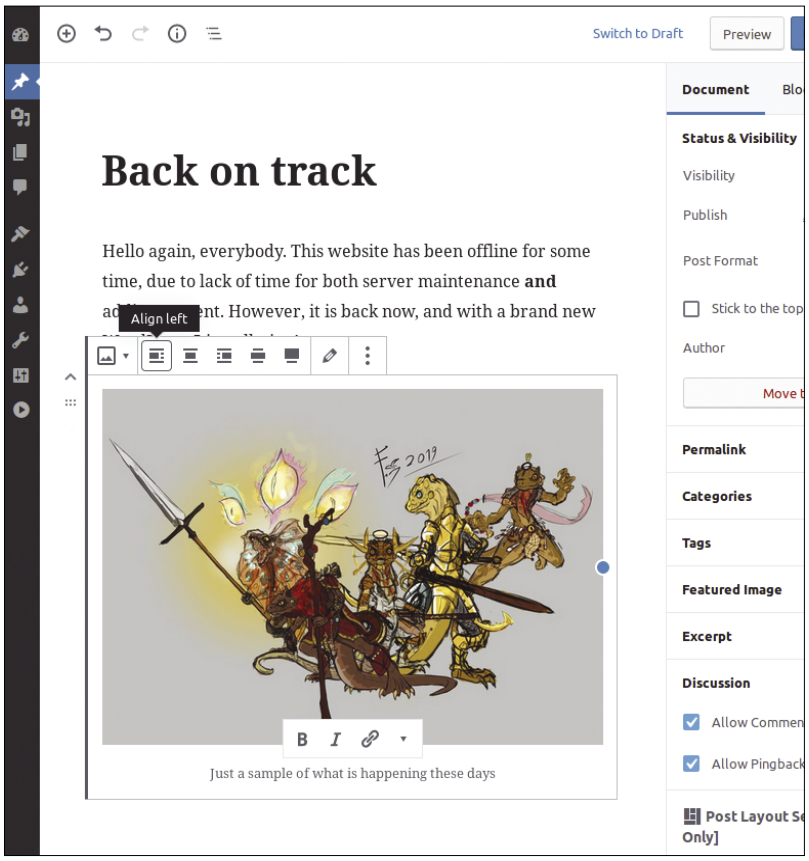


Figure 6: Clicking on a block activates a toolbar that shows the editing options for that type of block.

editing options for that block type (Figure 6). More advanced settings are accessible by opening the *Block* tab in the right panel (Figure 7). Posts previously created with TinyMCE will be treated as a single block, but you can split them into Gutenberg blocks by clicking on the three dots icon in the block toolbar and selecting *Convert to Blocks*. Do this only if the content doesn't translate well in the new WordPress 5 themes.

Clicking on the three dots icon also lets you access and edit the current block's source code.

Once you are comfortable with block-based writing, you can save your most frequently used block combinations as templates, which you can then insert in a post as a single block. To create a template, select the corresponding blocks, click on the three dots icon and select *Add to Reusable blocks*.

The Verdict on Gutenberg

For the most part, I like Gutenberg. Seriously. Still, I see a couple of issues in the underlying philosophy behind *anything* like Gutenberg that may create future problems. Gutenberg's official mission statement is to "go beyond the post into page templates and ultimately, full-site customization." In practice, Gutenberg was developed to prevent WordPress from losing users to services like Medium and Squarespace or page builders like Elementor. This is a noble and necessary goal, but one that can potentially backfire.

My issues with Gutenberg are the same ones I have with desktop office suites, "user-styled content" in GeoCities, and the `<blink>` HTML tags. First, mixing content and styling in one interface can make content (especially web content) much less portable to other publishing systems in the long run. Themes should be compatible with open, platform-independent standards, not with any one editor. Second, Gutenberg's "block-based editing" is as easy to use as it is to misuse (the Internet contains tens of millions of really embarrassing PowerPoint slideshows and MySpace pages).

However, if this frees users from Facebook, Medium, and the like, more power to Gutenberg. Just be very, very careful with it and avoid bloated themes!

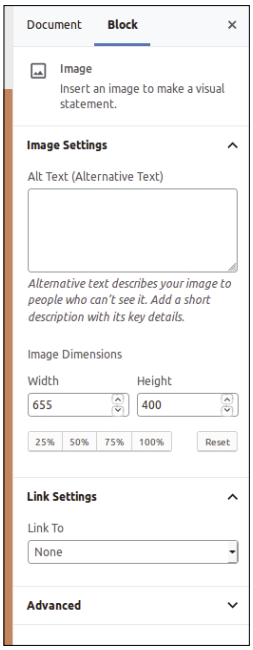


Figure 7: The *Block* tab lets you customize options for the selected block.

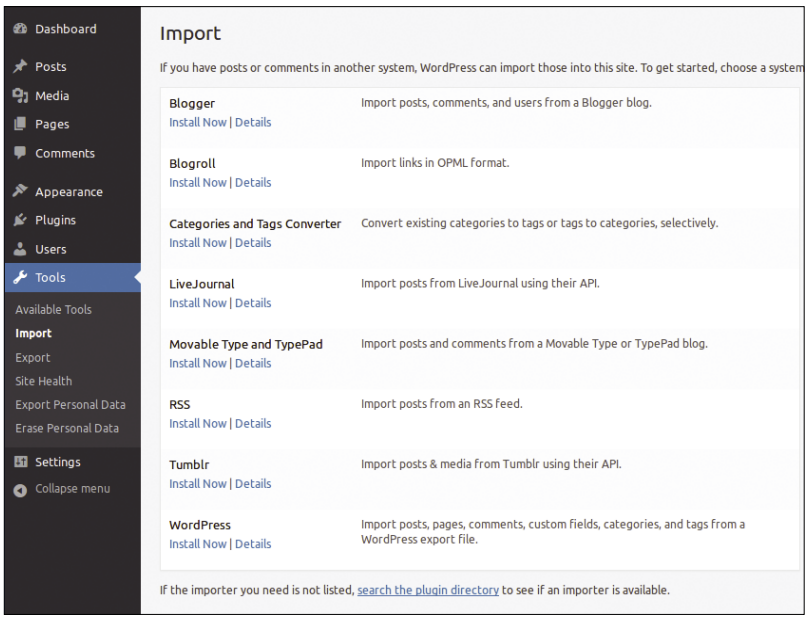


Figure 8: WordPress 5 can import your existing posts from many sources, though some may require extra manual work.

Advanced Topics

WordPress 5 includes three often ignored tools that make it even more powerful and flexible. First, *Tools | Import* (Figure 8) lists all the plugins available to semi-automatically import content from other WordPress blogs or other sources. While I haven't tested all of these options, some of these plugins have saved me a lot of time.

An even more powerful WordPress tool is

WP-CLI [7], the command-line interface for WordPress, which I have been using since 2011 [8]. Technically speaking, WP-CLI is a PHP command-line utility that works with any version of WordPress 4 and 5. Installing it only takes a couple of minutes at a Linux prompt, and it can be a timesaver. The WP-CLI commands are simple, well-documented, and can do anything from installing WordPress and WordPress plugins to publishing posts, moderating comments, cleaning the WordPress cache, and managing menus. Inside a shell script, WP-CLI can clone a fully customized WordPress site or import thousands of posts from any database or set of text files while you take a nap. Try it!

The final ignored WordPress tool is a plugin that does something that might seem like pure idiocy: Using WordPress to generate a completely static website!

The first websites were nothing but collections of static images and text files in HTML format. Consequently, every visitor saw exactly the same pages as everybody else and could not interact with the website with the exception of basic forms or JavaScript-based menus.

Dynamic websites like WordPress changed all that, because they are written in PHP or similar languages, which are executed by the web server every time a user requests a page. This architecture is what lets us post comments, fill forms, make purchases, browse images, complete surveys, and so on.

However, a drawback to dynamic websites is speed: Fast themes and plugins cannot do miracles. Unless you spend a lot of money on hosting, dynamic websites can be much slower than static ones, even if the latter are on free hosting accounts. Even more important is maintenance: It may take just a few minutes per week, on average, but if you don't always keep your WordPress installation as clean and up-to-date as possible, sooner or later it will stop working or be hacked.

While the money and/or time is justified in most scenarios, there are a couple of surprisingly common ones where a static site is preferable.

The first one is when your WordPress website reaches its end of life (usually because a project finishes). At that point, you may want the website to remain online, even if you are sure that nothing will be added or modified. In this case, continuing to run WordPress is both a waste of resources and a guarantee that sooner or later it will be hacked.

The other reason to make static versions of your WordPress website is if you want WordPress for *yourself* and not your users. When you want WordPress's themes, plugins, dashboard, and editor, but do not need user comments or any other interaction (search fields, dynamic calendars). In this case, you can:

1. Install and run WordPress on your laptop with the WP2Static [9] plugin, which works better for this kind of job than web mirroring tools like `wget`.
2. Disable anything that may not work as a static copy (sliders, search fields, etc.).
3. Run the plugin every time you add or modify your content or theme.
4. Automatically copy the static version to your static hosting account, with rsync or similar tools.
5. Enjoy your website made with all the power of WordPress, but much faster and more resistant to attacks.

Keep It Simple

Less is more and lasts longer, so use as few plugins as possible and the cleanest (but mobile-friendly) theme you can tolerate. Follow that rule, and WordPress 5 will give you, and all your website visitors, plenty of satisfaction. ■■■

Info

[1] WordPress: *https://wordpress.org*

[2] 25 Astonishing WordPress Facts: *https://www.whoishostingthis.com/compare/wordpress/stats/*

[3] The WordPress blog used in this tutorial belongs to my son.

[4] Let's Encrypt: *https://letsencrypt.org*

[5] Using permalinks: *https://wordpress.org/support/article/using-permalinks/*

[6] WordPress 5 Ultimate Cheat Sheet: *https://www.codeinwp.com/blog/wordpress-5-0-cheat-sheet/*

[7] WP-CLI: *https://wp-cli.org/*

[8] How to post content to a WordPress blog from the command line: *http://freesoftware.zona-m.net/how-to-post-content-to-a-wordpress-blog-from-the-command-line/*

[9] WP2Static: *https://wordpress.org/plugins/static-html-output-plugin/*

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*).

Tools to monitor your 3D prints

Printing Proper

One last step remains in our 3D-printing voyage: actually printing something. This issue, we'll tackle how to print and monitor your print at the same time.

BY PAUL BROWN

Over the last four issues, we have looked at how to design a piece for printing [1, 2, 3] and then how to slice it [4]. What we haven't done is any printing proper, or at least not using a controlling/monitoring program.

Why would you need a monitoring program? After all, you could just save your G-code file to an SD card, insert it into the printer, and print from there. The answer is that many of the low-end 3D printers (the ones you and I have sitting on our desks or in our garage) are quite dangerous: They have hot, moving parts out in the open and, at least in hobbyist set ups, are often near flammable things, such as wooden desktops and curtains.

Although it is recommended that you never move very far from your printer while it is working, a monitoring program, especially if it lets you live-feed video from the process, will give you a bit more freedom of movement.

There is another reason to use a printer-monitoring program: convenience. Setting up and starting a job directly from the printer is a bit like programming a 1980s Casio watch: It involves a lot of pressing and twisting of one single knob that beeps annoyingly from time to time as you navigate menus and options. It is much easier to use sliders and text boxes from a full-screen graphical interface.

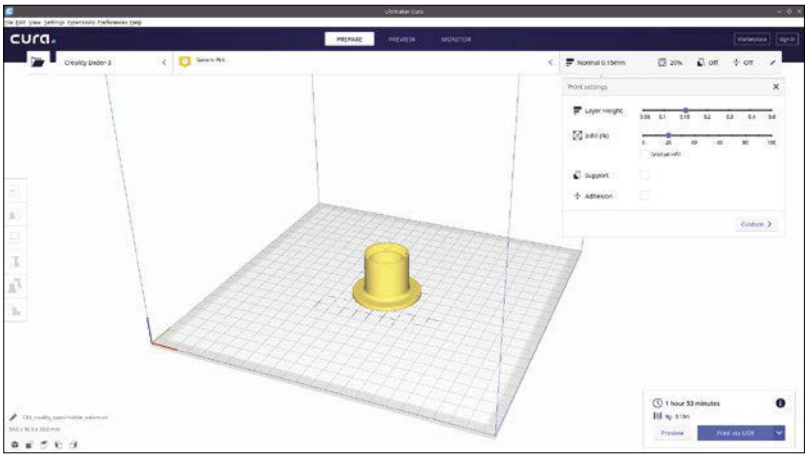


Figure 1: By connecting your printer directly to your computer, you can use Cura to print from your desktop.

Cura

In last month's issue, we covered Cura [5], primarily a slicer application [4]. But once you have sliced your object, you can also use Cura as a controller/monitor program for your printing jobs.

To start using Cura as a controller, make sure it supports your printer by picking it from the drop-down menu in the upper left-hand corner of the main pane or by visiting *Preferences* | *Configure Cura…* and clicking on the *Printers* option in the dialog.

Connect your printer to your computer using a USB cable, and Cura should pick it up automatically.

Load and slice your object as usual, and the label on the slicer button will change from *Slice* to *Print via USB* (Figure 1).

But before you hit the print button, visit the *Monitor* tab located at the top middle of the workspace. From there, you can regulate the temperature of the bed and hotend, as well as the location of the printing head, with a clear and easy interface (Figure 2). Check the ideal temperature for the material you will be using (this information is usually printed on the spool of filament), and use that information in the *Extruder 1* and *Build plate* fields to start heating up the printer.

Although it is convenient to slice and print from the same application, Cura is not very good for the latter. Apart from a progress bar, it does not seem to have any way to monitor how the print is going. It doesn't show the slice you are printing, and I was unable to get it to bring up a video feed from the camera I had trained on the printer (the big empty space on the left in Figure 2 is supposed to show that feed). It seems that some Ultimaker printers come with built-in cameras, and it may be that the feature only works with those.

Changing options during printing also made Cura crash a lot. Since each time it crashes, printing stops, forcing you to start over, you would be better off just using Cura as the very good slicer that it is.

Atelier/AtCore

Atelier/AtCore [6], on the other hand, is an excellent choice for controlling and monitoring your 3D printer. Part of the KDE suite of applications, AtCore is the back end, the API that allows graphical (or non-graphical) applications to connect and control your printer. Atelier (Figure 3) is one such application developed as a proof of concept by the Atelier/AtCore team.

To get started, first configure your printer for Atelier. In the Settings menu, choose *Profiles*. You will need to have your printer's specifications handy. Specifically, you need to know:

- Whether you printer is Cartesian (square base) or delta (circular base)
- The width, depth, and height of the printing area
- The maximum temperature your bed and hotend can reach
- The speed at which your printer and computer communicate
- The firmware used by your printer's controller board

Give your profile a name, something like the make and model of your printer, and save it (Figure 4).

In the pane on the right of Atelier's main window, choose the profile you just created and click the *Connect* button. The pane will immediately fill with information on the printer's status as shown in Figure 3. From the *Controls* tab, you can move the printing head around and set the temperatures of the bed and hotend.

The *Advanced* tab shows you detailed information in log form and lets you set things like the fan and printer speeds. The *Sd Card* tab supposedly allows you to browse (and print) the files on an SD card inserted into your printer, but that did not work for me.

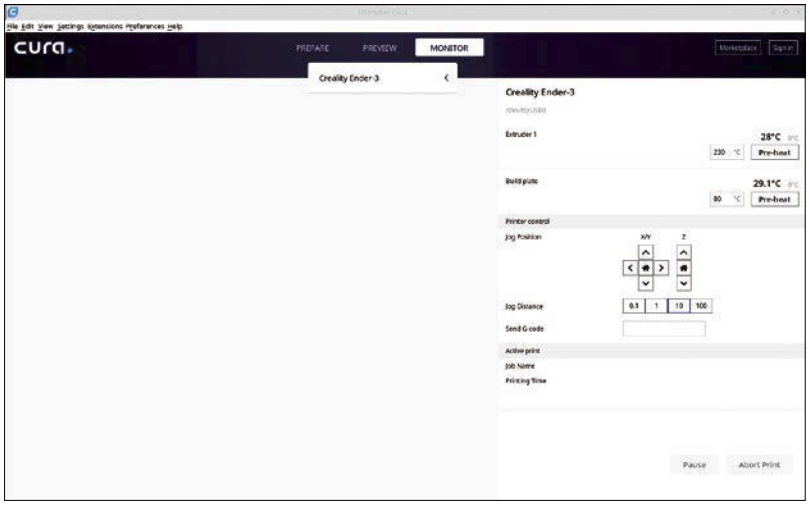


Figure 2: Control the temperature and the position of your extruder directly from Cura.

I should point out that Atelier/AtCore is under heavy development and some features don't work yet or get broken when a new version comes out – I tested Atelier 0.70.0 with AtCore 1.0.70. That said, the main thing Atelier/AtCore is meant to do (i.e., allow you to monitor and control a print from the comfort of your desktop), it does well, and the software is stable and does not crash.

Continuing on with the exploration of the interface, in a vertical toolbar on the left of the window, clicking the cube icon brings up a 3D view of the object you want to print. The lighting in the version I tested did not work, as you can see in Figure 3. Although you can make out the shape of the object, you cannot make out its features. It is also quite hard to rotate the object – only the left mouse button works for this, and there is no panning or zooming at the moment. Developers told

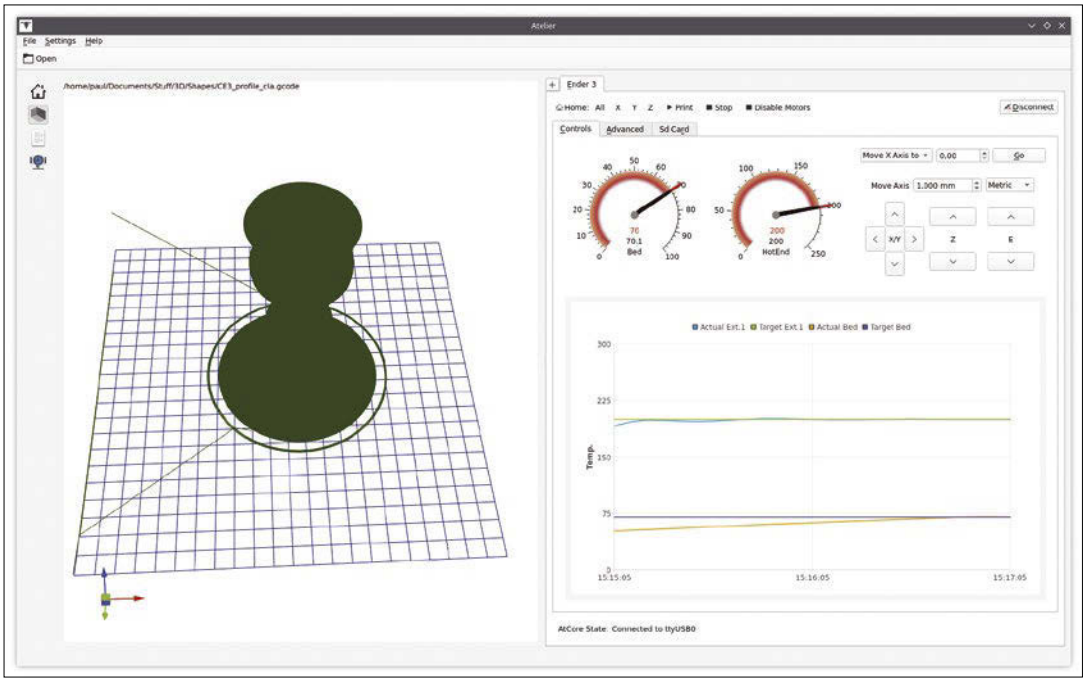


Figure 3: Atelier is an AtCore-based graphical application that allows you to control and monitor your 3D prints.

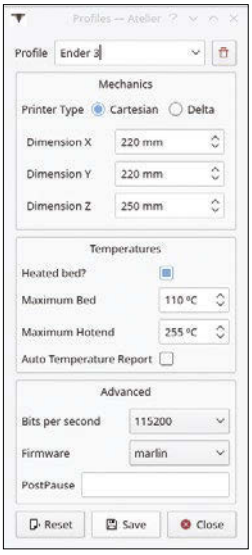


Figure 4: Configuring Atelier for the Creality Ender 3 printer.

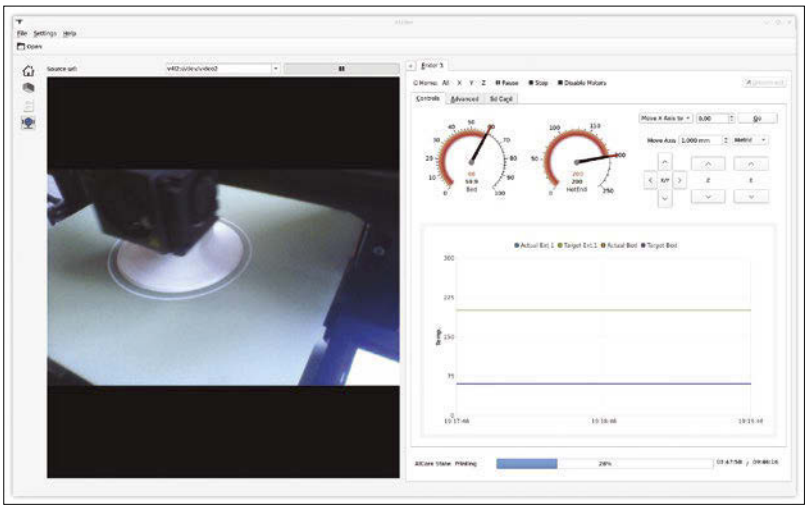


Figure 5: Atelier lets you monitor the print with a camera connected to your computer.

me the 3D viewport had a lot of the features stripped, because it made Atelier unstable. They are rewriting the whole thing and will re-implement it in the near future.

Clicking on the text page icon will bring up the file's G-code code, as in the list of commands used to render the 3D object. If you know what you are doing, you can modify the code by hand; this is terribly risky, though.

The camera icon at the bottom of the vertical toolbar opens up a pane where you can pick the camera you have trained on the print (Figure 5). Pick the device from the drop-down menu at the top of the pane and press the play button to start up the video feed.

Across the top of the pane on the right, there's another menubar that allows you to move the printing head to the *Home* position, start a print job, stop a print job, or disable your printer's motors – which allows you to move the bed and printing head around freely with your hand.

Notice that the right pane is actually a set of tabs. Click on the tab with the + sign, and you can open a new connection to another printer and manage several printers all at the same time and from one convenient application.

The Atelier/AtCore suite comes with another more compact client called AtCore Test Client (Figure 6). It has less functionalities than Atelier (there is, for example, no 3D visualizer or webcam support), but it does the job and is a good solution for a smaller screen. Indeed, AtCore Test Client is designed for embedded devices, like Raspberry Pis and other single-board computers (SBCs), with a touchscreen mounted directly onto your printer.

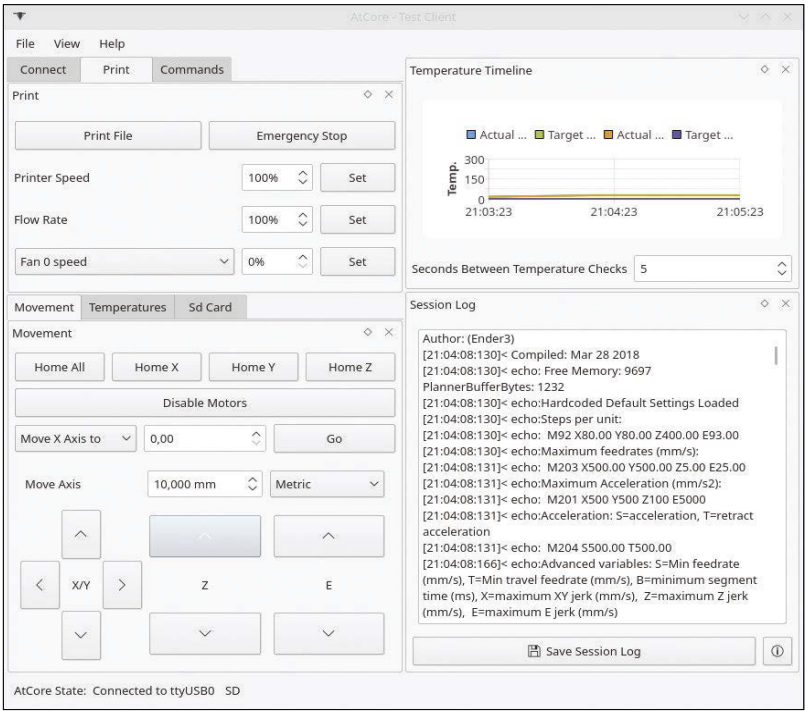


Figure 6: AtCore Test Client is another AtCore-based application ideal for small screens.

OctoPrint/OctoPi

OctoPrint [7] is probably the most used monitoring and controlling software and for good reason. You can install OctoPrint on your personal computer, and, when you run it, it will set up a local website that you can visit and use to control your printer (Figure 7). This is super convenient, as it means you can access and control your printer from any device on your local network, even your phone.

You could also make it available from the Internet; the configuration allows you to do this, but that is a huge security risk. Besides, I did say that you should not stray very far from a printer while it is printing, remember? If you must set things up so your printer is available remotely, be very careful!

OctoPrint is written in Python, so it does not need compiling, but that is about the only simple thing about its setup. To be fair, following the instructions carefully in the "Installing from source" section online [8], you can get the server up and running after multiple steps. However, then configuring things like the webcam or slicing capabilities is a bit of a nightmare and involves knowledge of streaming to web and compiling obsolete libraries that go way beyond the scope of this article.

So if OctoPrint is so fiddly, why is it so popular? OctoPi is why.

OctoPi is a version of Raspbian, the most popular operating system for the Raspberry Pi, and it does one thing and one thing alone: Run a perfectly configured version of OctoPrint. Download the image from [8], `dd` it to a microSD card, pop it in your Rasp Pi (preferably a version 3 upwards), plug your camera and printer into the Pi, power up and … Hey presto! You have a full-featured monitoring device.

To configure the WiFi, make sure your Rasp Pi is connected by a network cable to your network,

find its IP, and `ssh` into your Pi – username *pi*, password *raspberry*. Run

```
sudo raspi-config
```

and navigate to *2 Network options* and then *N2 Wi-fi*. Fill in the SSID of your WiFi network, select *Ok*, and fill in the password. Again select *Ok* when you are done. In the next screen, choose *Finish* and `raspi-config` will ask whether you want to reboot. Say yes.

On reboot, you will have a WiFi-enabled device that you can zip-tie to your printer. This means your printer is now portable, WiFi enabled, and cable-free (except for the power cord, of course)! You can still move it off your desk to anywhere in your house where the WiFi reaches, then connect to it, and send it jobs from the comfort of your laptop.

Open a browser and input *octopi.local* in the address bar, and your browser will take you to your local OctoPi instance.

If you have read through the configuration of Atelier/AtCore, configuring OctoPrint isn't much different. In fact, once you log into the website for the first time (again, username is *pi* and password is *raspberry*), OctoPrint will run a wizard that will help you get everything set up (Figure 8). If you ever need to reconfigure, choose the wrench icon (Settings) from the toolbar at the top of the web interface.

The wizard takes you through *Access Control* to OctoPrint's web interface, metrics that allow the OctoPrint project to know how many people are using the software (you can opt out, of course), connectivity to the Internet that allows OctoPrint to check for updates, setting up plugins, and setting up your printer's profile – here again, you will need your hardware specs.

Once you're done configuring, you will get back to OctoPrint's main screen. Click *Connect* to link up OctoPrint to your printer, and the graph on the right will come to life with data from the bed and hotend. You can adjust the temperatures of each using the buttons and drop-down menus under the graph. You can also preconfigure temperatures for different materials (PLA, ABS, PET, etc.) by clicking on the wrench icon at the top of the page and choosing *Temperatures* in the *Printer* section. Preconfigured values will appear in the blue drop-down menu to the right of the target temperature text box.

On the left, below the box showing the state of the printer, you have a list of recently used files you can print. You can access files both on the SD card in your printer and on your hard drives. Scroll down and you will see an *Upload* button at the bottom of the box. Press it and you can navigate through your files until you find what you want to print.

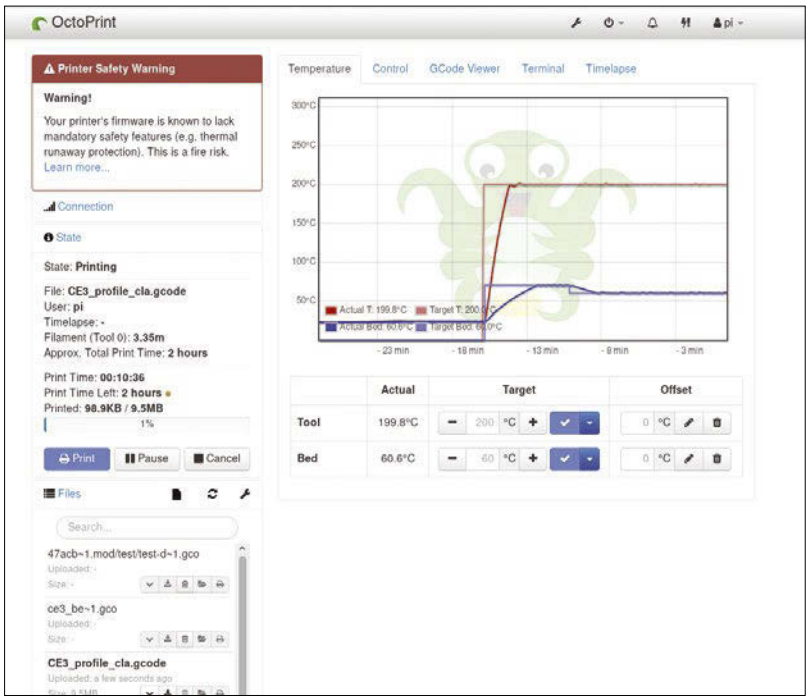


Figure 7: OctoPrint is probably the most popular monitoring software for 3D printers.

Across the top of the main pane, you have the *Temperature*, *Control*, *GCode Viewer*, *Terminal*, and *Timelapse* tabs. I have already talked about the *Temperature* tab, as it is the default tab you land on when you visit OctoPrint for the first time. The *Control* tab shows a shot from the camera and controls for moving about and disabling motors and fans (Figure 9). Use this to remotely monitor your print and place the printing head.

The *GCode* tab (Figure 10) does not show a 3D view of a loaded object as you would expect,

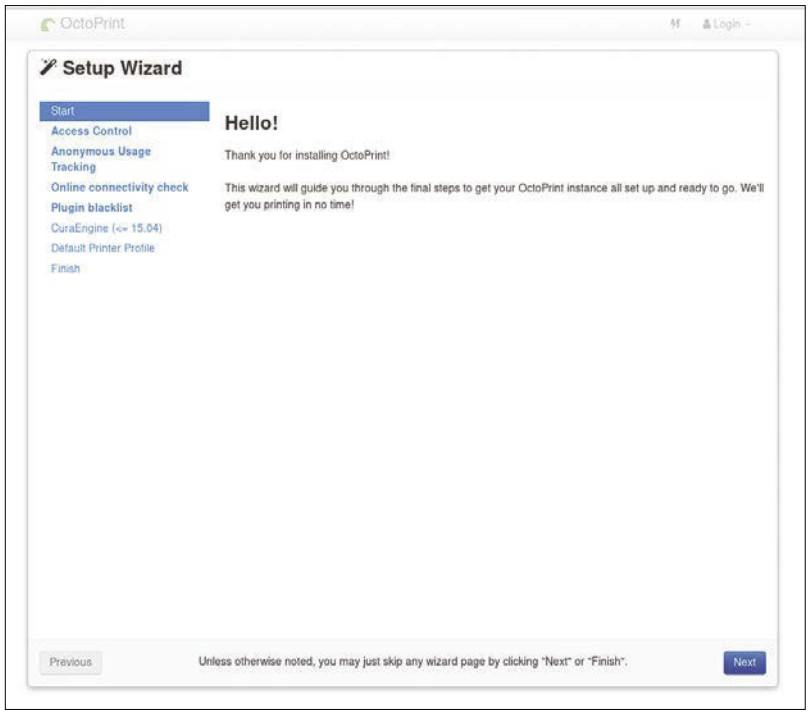


Figure 8: OctoPrint runs a configuration wizard the first time you log in.

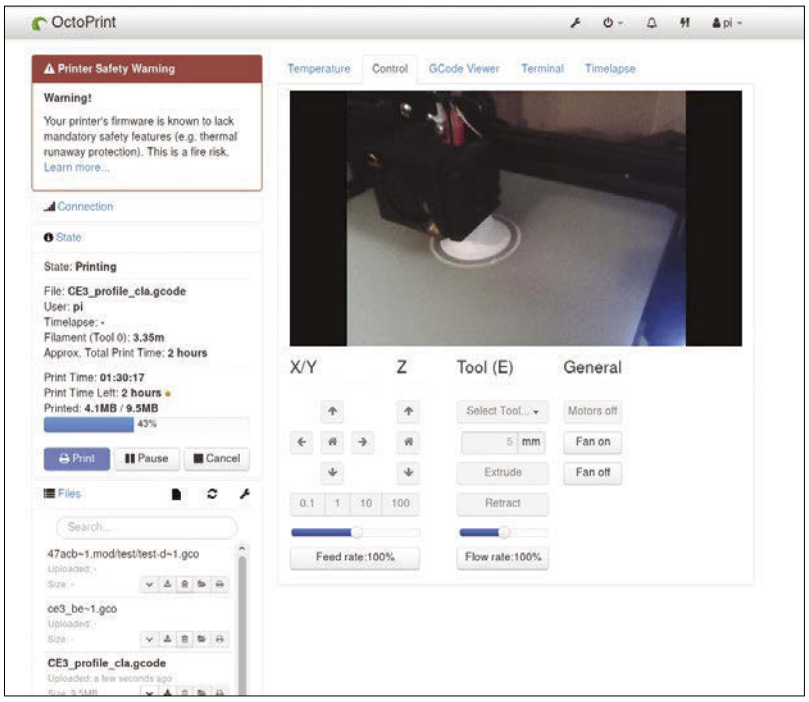


Figure 9: The *Control* tab shows the camera and buttons to move the motors.

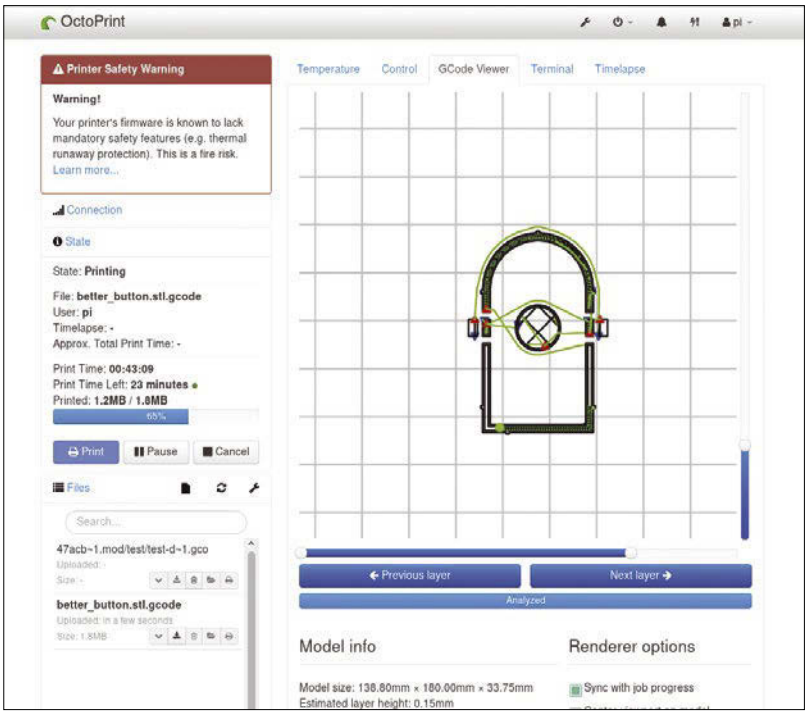
but instead it shows the layers of the selected object. While not printing, you can use the slider on the right to scroll through all the layers that will make up the object. The slider on the bottom will show you the path the extruder will follow to print the slice. While printing, the *GCode* tab will show you an animation of the print's progress.

The *Terminal* tab shows information received from the printer in a logging-like format. If anything goes wrong, you can use the *Copy all* link to copy all the contents to your clipboard. You can later copy the contents to a text file, so you can analyze and troubleshoot the print.

Timelapse lets you record a frame every certain amount of seconds to create cool movies of your prints.

To start printing, select an object from the *File* list, set the temperature of the bed and extruder in the *Temperature* tab, and, when the bed and extruder have reached the correct temperature, click on *Print*. You can *Cancel* a print that is going wrong, or you can *Pause* it, say, to change to another filament and print in several colors – that said, successfully stopping and starting a print is tricky, so be careful.

Figure 10: The GCode tab shows the selected object's layers.



Conclusion

Both Atelier/AtCore and OctoPrint are fine pieces of software that you can use to monitor your printer. If you have a Rasp Pi handy, OctoPi is probably the most convenient for remote monitoring of single printers. It also makes it really easy to integrate the controlling hardware (i.e., a Rasp Pi) into the printer.

Atelier, on the other hand, is great for controlling several or a whole farm of printers. Its smaller cousin, the AtCore Test Client, is ideal if you want to mount a control panel using an SBC (like a Rasp Pi) and touchscreen to your printer. ■■■

Info

[1] "3D Designer" by Paul Brown, *Linux Magazine*, issue 222, May 2019, pp. 90-94, *http://www.linux-magazine.com/Issues/2019/222/Designing-your-own-stuff-with-OpenSCAD*

[2] "Using OpenSCAD to build custom 3D pieces" by Paul Brown, *Linux Magazine*, issue 223, June 2019, pp. 90-94, *http://www.linux-magazine.com/Issues/2019/223/Tutorials-OpenSCAD*

[3] "Technical 3D design using FreeCAD" by Paul Brown, *Linux Magazine*, issue 224, July 2019, pp. 90-95, *http://www.linux-magazine.com/Issues/2019/224/Mother-of-Invention*

[4] "Preparing an object for 3D printing" by Paul Brown, *Linux Magazine*, issue 225, August 2019, pp. 90-95, *http://www.linux-magazine.com/Issues/2019/225/Print-Ready*

[5] Cura slicer: *https://ultimaker.com/software/ultimaker-cura*

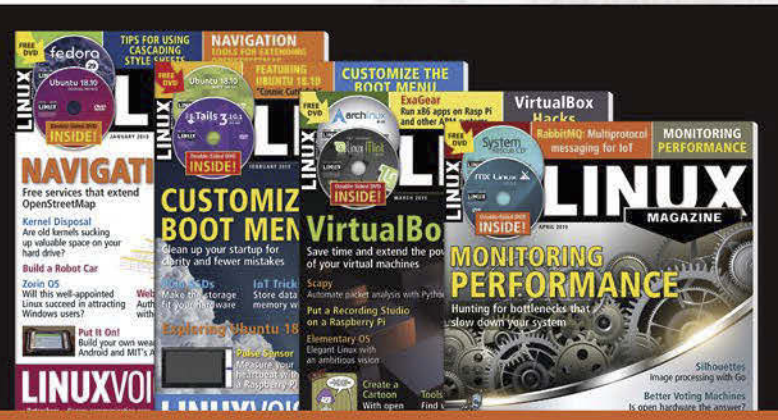
[6] Atelier/AtCore: *https://atelier.kde.org/*

[7] OctoPrint: *https://octoprint.org*

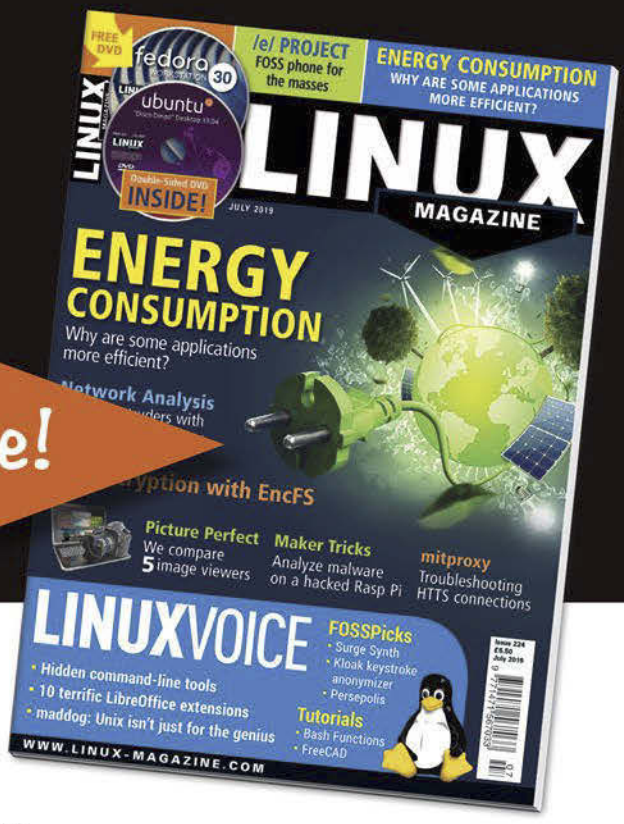
[8] OctoPi download: *https://octoprint.org/download/*

What?!

I can get my issues SOONER?



Available anywhere, anytime!



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

shop.linuxnewmedia.com/digisub

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*.

JAX London

Date: October 7-10, 2019

Location: London, England

Website: *https://jaxlondon.com/*

JAX London is the conference for Java and software innovation. Join us in London for four days with cutting edge software engineers and enterprise-level professionals innovating in the fields of JAVA, microservices, continuous delivery, and DevOps.

All Things Open

Date: October 13-15, 2019

Location: Raleigh, North Carolina

Website: *https://allthingsopen.org/*

All Things Open is the largest "open" technology event on the East Coast of the United States. The seventh year of All Things Open features some of the top technologists and decision makers in the world discussing the topics of open source, open technology, and the open web in the enterprise.

LISA19

Date: October 28-30, 2019

Location: Portland, Oregon

Website: *https://www.usenix.org/conference/lisa19*

LISA is the premier conference for operations professionals. Sys admins, systems engineers, IT operations professionals, SRE practitioners, developers, IT managers, and academic researchers will share real-world knowledge about designing, building, securing, and maintaining critical systems.

Events

Guadec 2019	August 23-28	Thessaloniki, Greece	https://2019.guadec.org/
Akademy 2019	September 7-13	Milan, Italy	https://akademy.kde.org/2019
Linux Plumbers Conference	September 9-11	Lisbon, Portugal	https://linuxplumbersconf.org/
Cloud Foundry Summit Europe	September 11-12	The Hague, Netherlands	https://www.cloudfoundry.org/event/summit/
HPC on Wall Street	September 11-12	New York, New York	https://www.hpconwallstreet.com/
The Linux Kernel Maintainer Summit	September 12	Lisbon, Portugal	https://events.linuxfoundation.org/events/linux-kernel-maintainer-summit-2019/
All System Go!	September 20-22	Berlin, Germany	https://all-systems-go.io/
Open Networking Summit Europe 2019	September 23-25	Antwerp, Belgium	https://events.linuxfoundation.org/events/open-networking-summit-europe-2019/
Storage Developer Conference	September 23-26	Santa Clara, California	https://www.snia.org/events/storage-developer
Deutsche OpenStack Tage 2019	September 24-26	Berlin, Germany	https://openstack-tage.de/
JAX London 2019	October 7-10	London, United Kingdom	https://jaxlondon.com/
All Things Open	October 13-15	Raleigh, North Carolina	https://allthingsopen.org/
Open Source Summit + Embedded Linux Conference Europe	October 28-30	Lyon, France	https://events.linuxfoundation.org/events/open-source-summit-europe-2019/
LISA19	October 28-30	Portland, Oregon	https://www.usenix.org/conference/lisa19
DrupalCon Amsterdam 2019	October 28-31	Amsterdam, Netherlands	https://events.drupal.org/amsterdam2019
Linux Security Summit Europe	Oct 31 - Nov 1	Lyon, France	https://events.linuxfoundation.org/events/linux-security-summit-europe-2019/

CALL FOR PAPERS

We are always looking for good articles on Linux and the tools of the Linux environment. Although we will consider any topic, the following themes are of special interest:

- System administration
- Useful tips and tools
- Security, both news and techniques
- Product reviews, especially from real-world experience
- Community news and projects

If you have an idea, send a proposal with an outline, an estimate of the length, a description of your background, and contact information to *edit@linux-magazine.com*.

The technical level of the article should be consistent with what you normally read in *Linux Magazine*. Remember that *Linux Magazine* is read in many countries, and your article may be translated into one of our sister publications. Therefore, it is best to avoid using slang and idioms that might not be understood by all readers.

Be careful when referring to dates or events in the future. Many weeks could pass between your manuscript submission and the final copy reaching the reader's hands. When submitting proposals or manuscripts, please use a subject line in your email message that helps us identify your message as an article proposal. Screenshots and other supporting materials are always welcome.

Additional information is available at:
http://www.linux-magazine.com/contact/write_for_us.

NOW PRINTED ON recycled paper from 100% post-consumer waste; no chlorine bleach is used in the production process.

Authors

Contact Info

Editor in Chief
Joe Casad, jcasad@linux-magazine.com

Copy Editor
Amy Pettle

News Editor
Swapnil Bhartiya

Editor Emerita Nomadica
Rita L Sooby

Managing Editor
Lori White

Localization & Translation
Ian Travis

Layout
Dena Friesen, Lori White

Cover Design
Dena Friesen

Cover Image
© Ievgenii Skryzhak & macrovector, 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.

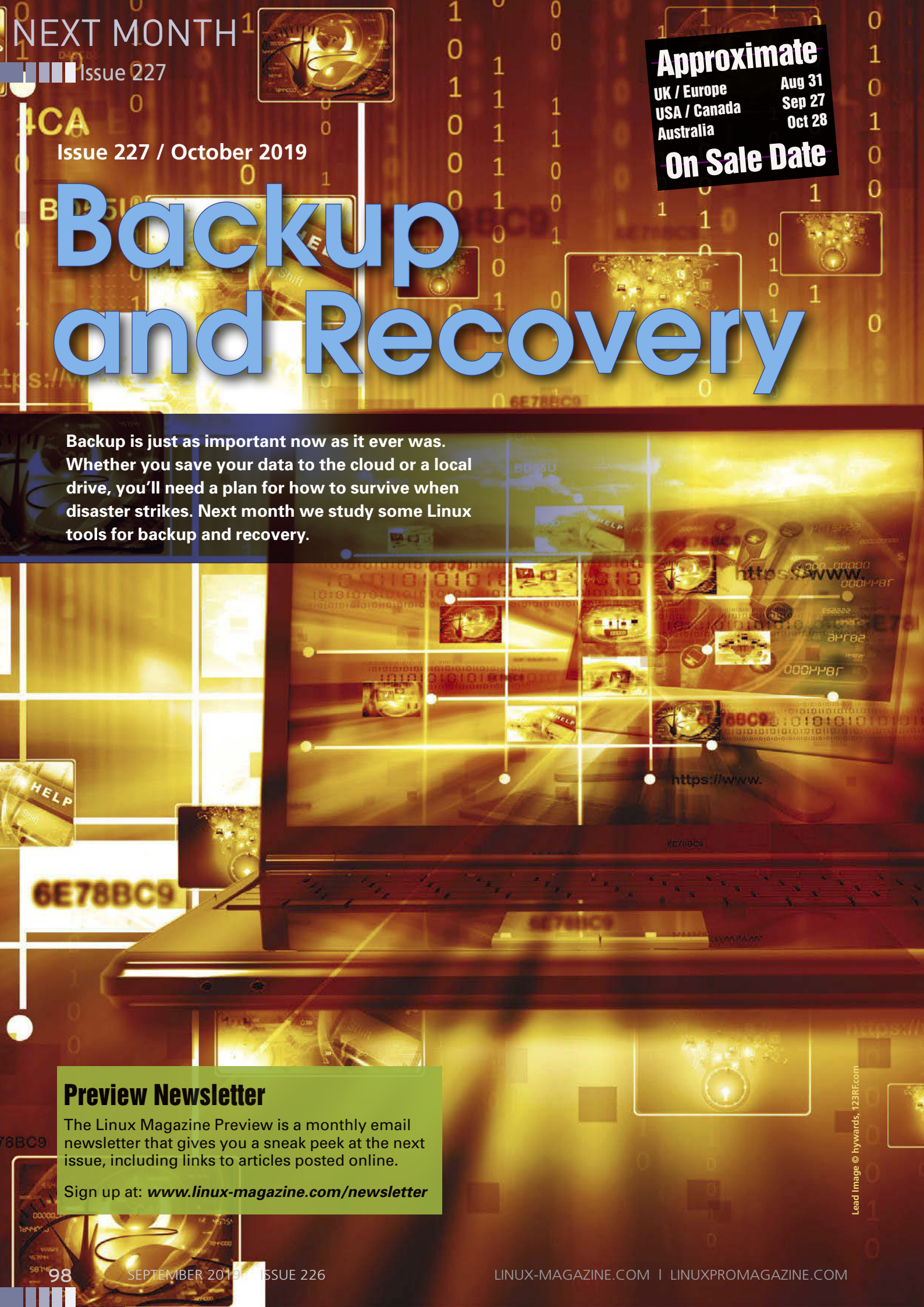
Issue 227 / October 2019

Backup and Recovery

Approximate On Sale Date

UK / Europe	Aug 31
USA / Canada	Sep 27
Australia	Oct 28

Backup is just as important now as it ever was. Whether you save your data to the cloud or a local drive, you'll need a plan for how to survive when disaster strikes. Next month we study some Linux tools for backup and recovery.



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***

LISA is the premier conference for operations professionals, where sysadmins, systems engineers, IT operations professionals, SRE practitioners, developers, IT managers, and academic researchers share real-world knowledge about designing, building, securing, and maintaining the critical systems of our interconnected world.

Registration and the program will be available in August 2019.

Sign up to receive updates at www.usenix.org/lisa/linuxpro, and follow @LISAconference on Twitter.

usenix
THE ADVANCED COMPUTING SYSTEMS ASSOCIATION

SUPERMICRO

Better. Faster. Greener.

Expect Better Data Center Performance, TCO & Impact on the Environment

35% Faster | Up to 50% TCO Reduction | Reduce E-Waste
Featuring 2nd Generation Intel® Xeon® Scalable Processors

intel XEON PLATINUM inside

Learn More at www.supermicro.com/X11

SUPERMICRO