FREE DVD

Timers
WILL SYSTEMD REPLACE CRON?

Systemd Stories
TRICKS WITH THE NEW LINUX INIT SYSTEM

LINUX PRO MAGAZINE

SEPTEMBER 2018

Systemd

Getting more from the elusive new Linux init system

10 Top PDF Readers

Arduino Programming for Open Hardware Projects

Homegrown Facial Recognition
Build facial recognition into your own Python scripts

Maker Tricks
Monitor a beehive with a Raspberry Pi

Cover Your Ass(ets)
Back up your system files with CYA

MiyoLinux
A lean Linux that still loves SysV init

Massimo Banzi
Arduino cofounder explains how it all got started

LINUXVOICE

- **maddog:** Adopting FOSS firmware can keep your hardware out of the landfill
- **Ncdu:** Trace down big files that are clogging your disk drive

FOSSPicks
- Cutter Disassembler
- Ctrlr MIDI Editor

Tutorials
- Doing science with BOINC
- DocFetcher: Easy and powerful local search tool

Issue 214
Sept 2018
US$ 15.99
CAN$ 17.99

HETZNER ONLINE

ULTIMATE PERFORMANCE
COMPLETE FLEXIBILITY

BRAND NAME HARDWARE FROM DELL

Dedicated Root Server DX152

- ✔ Dell PowerEdge™ R640
- ✔ 2 x Intel® Xeon® Silver 4114 10-Core (Skylake-SP)
- ✔ Incl. Hyper-Threading-Technology
- ✔ 64 GB DDR4 ECC RAM
  max. 768 GB at additional cost
- ✔ Up to 10 drives at additional cost
- ✔ No minimum contract
- ✔ Setup Fee $187

monthly from $ **187**

Dedicated Root Server DX292

- ✔ Dell PowerEdge™ R640
- ✔ 2 x Intel® Xeon® Gold 6130 16-Core (Skylake-SP)
- ✔ Incl. Hyper-Threading-Technology
- ✔ 64 GB DDR4 ECC RAM
  max. 768 GB at additional cost
- ✔ Up to 10 drives at additional cost
- ✔ No minimum contract
- ✔ Setup Fee $257

monthly from $ **257**

www.hetzner.com

TRIPLE AXIS

Dear Reader,

Our whole economy is based on the idea that competition leads to innovation. Interestingly, the cause of innovation is best served if no one ever *wins* that competition. A company that truly succeeds in defeating all its adversaries gets complacent, and innovation tends to dry up.

For much of the PC era, Intel and AMD dueled for market share in the competitive PC industry. Intel was always bigger – well, *way* bigger – but AMD found ways to stay competitive. A computer with an AMD processor cost just a little less, which gave a push to demand, and AMD could match Intel with some serious engineering chops, which kept the competition lively. Intel would come up with some new innovation, and AMD would find a way to do something similar, or at least close enough to achieve near-equivalent performance. And AMD always put the emphasis on economy, forcing Intel to stay real with pricing.

The competition continued for several years around the double axis of performance and price. But sometime a few years ago, AMD started to fall behind. A miscalculation? An economic downturn? Maybe Intel just got so big it sucked up all the attention? Or maybe the price for a PC processor got so low that it wasn't as easy for AMD to slide under the Intel price? For whatever reason, AMD stock dropped to $1.67 per share sometime in 2015 – a fraction of its previous value. Many of us who believed the presence of AMD was a very good thing for the industry started to wonder if AMD would have a way to keep marching, but the company did exactly what one would hope they would do at such a moment: they called their best minds together and came up with a plan.

That plan has been exciting to watch. AMD's Zen architecture arrived in 2016, and the Ryzen processor series debuted in 2017, which was, by all accounts, a very good year for AMD. Revenues are up, and a share of stock is up to $16 – 10 times the previous low. The Radeon Open Compute (ROCm) initiative unveiled a whole new open framework for GPU computing, which plays to AMD's strength in the graphics processor space.

I don't know how many high-tech companies have gotten to the edge of disaster in recent years and gone right on over the edge. Nokia CEO Stephen Elop wrote his famous "Burning Platform" memo in 2011 to rally the troops and save the company [1], but it didn't work: The recommended plunge into icy waters could not forestall the final combustion.

AMD and its CEO Lisa Su have my admiration for working out a plan and keeping their focus on bringing the company back to stability. Comebacks are easy to wish for and not so easy to accomplish in real life.

Meanwhile, though, a third contender has entered the ring. ARM processors are a common feature of cell phones and tablets, and now that phones and tablets claim an ever-larger share of work that was once performed on PCs, the ARM universe is getting ever-closer to direct competition with Intel and AMD.

And the ARM phenomenon is seriously spilling out of the once-isolated mobile space. At the end of 2017, Microsoft rolled out a new line of ARM-based Windows 10 laptops. Even the speed-conscious high-performance computing industry is starting to look seriously at ARM, and the first petascale ARM supercomputer went online earlier this year [2].

The best feature of ARM processors is their efficient power usage. Low power means that a smartphone can ride around in your pocket longer without a charge. It also means (at least theoretically) you can pack more processors into that supercomputer without it overheating or using all the available energy.

Intel and AMD never had to be serious about optimizing for power usage in past generations, but now it looks like they will have to. And if they do, a new triple axis of performance versus price versus power could lead to a new generation of better, less expensive processors.

Joe Casad,
Editor in Chief

Info

[1] Burning Platform memo: *https://www.engadget.com/2011/02/08/nokia-ceo-stephen-elop-rallies-troops-in-brutally-honest-burnin/*

[2] First petascale ARM computer: *https://www.top500.org/news/sandia-to-install-first-petascale-supercomputer-powered-by-arm-processors/*

LINUX MAGAZINE

WHAT'S INSIDE

The Systemd init system is here to stay. You probably know it is running on your system somewhere, but have you gotten inside and explored what you can do with it? This month we take a close look at Systemd.

Other highlights:

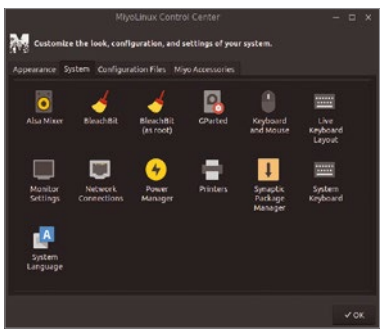
- **Facial Recognition** – Snapshot columnist Mike Schilli shows how to integrate a free facial recognition library with your Python scripts (page 56).
- **Bee Monitoring** – Beekeepers keep watch over their hive with a Raspberry Pi (page 72).

Check out LinuxVoice for a look at how you can participate in science with the BOINC crowd-source analysis framework (page 82).

SERVICE

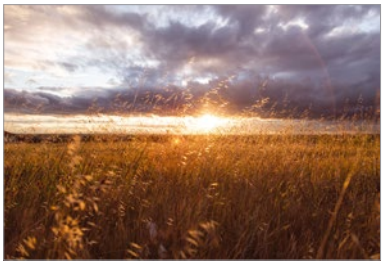
NEWS

REVIEWS



COVER STORIES

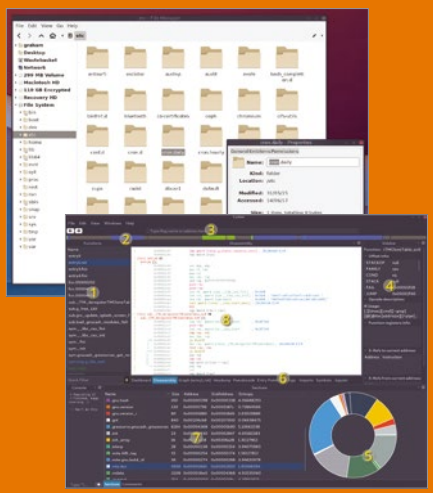
IN-DEPTH



MAKERSPACE

SEE PAGE 6 FOR DETAILS

LINUXVOICE



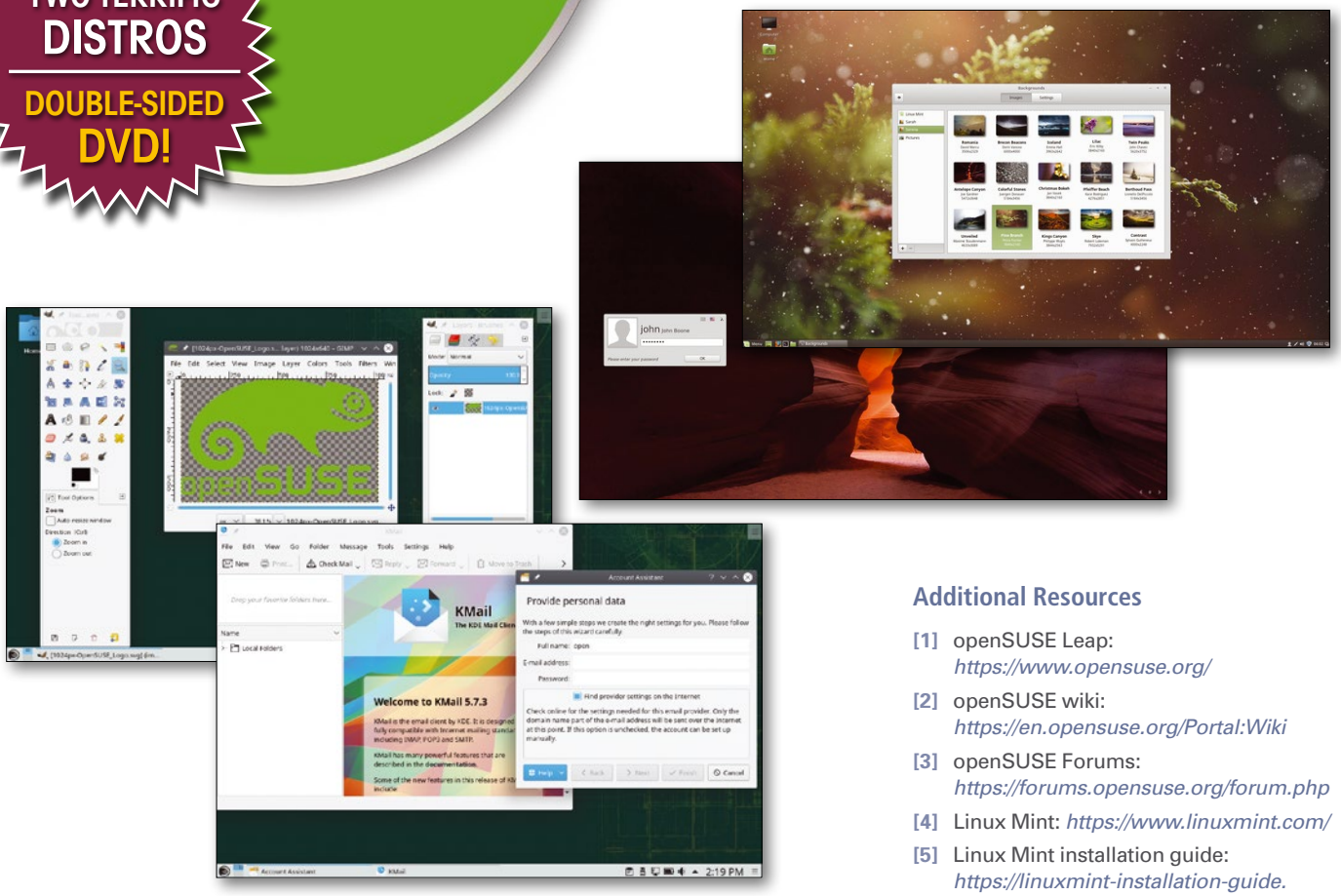
On the DVD

openSUSE Leap 15.0 (64-bit)

openSUSE is a community-driven Linux project sponsored by SUSE. The openSUSE "Leap" edition uses source code from the high-end SUSE Linux Enterprise (SLE), making it unusually stable and robust for a free community Linux. The latest version comes with a new feature that supports easy migration to SLE. Other changes include enhanced container support, new development tools, and several improvements to the YaST management feature.

Linux Mint 19 "Tara" Cinnamon Edition (64-bit)

Mint is a popular Linux desktop edition based on Ubuntu. The latest version is a long-term support release, with support provided until 2023. Mint 19 features the new Timeshift tool, which makes it easy to restore your system to a previous system snapshot. The edition included on this month's DVD features the Cinnamon desktop, which is based on Gnome 3. Cinnamon 3.8 offers better performance, improved sound control, and enhanced search capabilities.



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

Additional Resources

[1] openSUSE Leap: *https://www.opensuse.org/*

[2] openSUSE wiki: *https://en.opensuse.org/Portal:Wiki*

[3] openSUSE Forums: *https://forums.opensuse.org/forum.php*

[4] Linux Mint: *https://www.linuxmint.com/*

[5] Linux Mint installation guide: *https://linuxmint-installation-guide.readthedocs.io/en/latest/*

Get the $200 of on registration by registering at www.snia.org/SDcode and using code SDC18LINUXPRO.

Don't Miss This!

Storage Developer CONFERENCE

Santa Clara, CA, September 24-27, 2018

SDC 18

Produced since 1998, SNIA's Storage Developer Conference (SDC) is the only event created **BY** storage developers **FOR** storage developers.

Register at www.snia.org/SDcode.

Register to attend SDC for technical discussions and education on the latest storage technologies and standards.

Register at www.snia.org/SDcode and use code SDC18LINUXPRO to recieve **$200 off** registration.

SDC 2018 will deliver over 100 presentations on topics including Persistent Memory, File Systems, Orchestration, NVMe, Object Storage, Security, Performance, and more.

Learn from hundreds of experts in one place!

www.storagedeveloper.org

SNIA®

NEWS

Updates on technologies, trends, and tools

THIS MONTH'S NEWS

Honey I Shrunk Ubuntu

Canonical is tightening its focus on cloud and enterprise markets. The company has released a new version of Ubuntu, dubbed Minimal Ubuntu, which it claims is optimized for automated use at scale, with a tiny package set and minimal security cross-section.

Canonical claims that Minimal Ubuntu is the smallest Ubuntu base image for cloud operations. These images are less than 50% the size of the standard Ubuntu server image and boot up to 40% faster.

© Phil Wohlrab, 123RF.com

Despite its reduced size, Minimal Ubuntu retains full compatibility with standard Ubuntu. Any Ubuntu package can be installed on Minimal Ubuntu.

"The small footprint of Minimal Ubuntu, when deployed with fast VM provisioning from GCE, helps deliver drastically improved boot times, making them a great choice for developers looking to build their applications on Google Cloud Platform," said Paul Nash, Group Product Manager, Google Cloud.

Images of Minimal Ubuntu 16.04 LTS and 18.04 LTS are available for use now in Amazon EC2, Google Compute Engine (GCE), LXD, and KVM/OpenStack.

Source: *https://blog.ubuntu.com/2018/07/09/minimal-ubuntu-released*

Linux Mint 19 Released

The Linux Mint team has announced the release of Linux Mint 19, code named Tara. Linux Mint 19 comes in three flavors – Cinnamon, Mate, and Xfce. Linux Mint 19 is based on the Ubuntu LTS 18.04 release, which is supported until 2023.

Linux Mint gained popularity during the early days of Gnome 3 and the Unity desktop. Linux Mint created the Cinnamon desktop, which offered the good old WIMP (Windows, icons, mouse & pointer), as compared to touch-friendly, future proof Gnome 3 and Unity.

One of the highlights of Linux Mint 19 is Timeshift, a name and feature borrowed from Apple's Time Machine that enables users to create a system backup and restore the system if something goes wrong.

To make life easier for users, Linux Mint 19 simplifies the Update Manager. "The Update Manager no longer promotes vigilance and selective updates. It relies on Timeshift to guarantee the stability of your system and suggests to apply all available updates," said the official blog post.

The Linux Mint team has offered an update mechanism, so users running Linux Mint 18 can upgrade to Linux Mint 19.

Linux Mint has not been free of controversies. The team edits Firefox to remove Google as the default search engine and replace it with Yahoo!

You can download Linux Mint from the official download page: *https://www.linuxmint.com/release.php?id=32*

Red Hat Adds GPLv3 Cure Clause to Its Codebase

Red Hat has taken the next step to ensure that users of its open source software are protected from any GNU GPL violations (*https://www.redhat.com/en/blog/gpl-cooperation-commitment-and-red-hat-projects*). Many companies mix different open source codebases into their products and services. It could be challenging to keep up with the licenses used for each component, and any violation could lead to a court case and public shaming.

GNU GPLv3 added a cure clause that offers a grace period to violators to fix the violation and resume the right to use the codebase.

Red Hat said that all new Red Hat-initiated open source projects that opt to use GPLv2 or LGPLv2.1 will be expected to supplement the license with the cure commitment language of GPLv3. The cure language will live in a file in the project source tree and will function as an additional permission extended to users from the start.

There have been some cases where companies using GPL'd software were attacked by trolls and dragged into courts. However, the cure clause offers companies a grace period to fix any violation and avoid such a situation.

Red Hat said in a blog post, "We are extending the GPLv3 termination policy to users of our GPLv2/LGPLv2.1 code because we consider it the right thing to do. The cure permissions offer additional comfort level that users of our code have reasonable assurance of quiet use of that code, even if there is a temporary license noncompliance by a third party redistributing our code, due to misunderstanding or otherwise. We also believe that community adoption of these rights will reduce the opportunity for illegitimate forms of license enforcement. We hope that others will also join in this endeavor to reassure the open source community that good faith efforts to fix noncompliance will be embraced."

SUSE Linux Enterprise 15 Released

SUSE has announced the release of SUSE Linux Enterprise 15 (SLES 15). It's a major "leap" not only in terms of the operating system's architecture, but also the numbering. Thanks to some superstitions in its core markets, SUSE skipped numbers 13 and 14 and jumped to 15. Technically this would have been SLES 13.

SUSE calls SLES 15 a multimodal operating system that's designed to cater to both traditional and modern workloads – from data centers to the cloud.

"As organizations around the world transform their enterprise systems to embrace modern and agile technologies, multiple infrastructures for different workloads and applications are needed," said Thomas Di Giacomo, SUSE CTO. "This often means integrating cloud-based platforms into enterprise systems, merging containerized development with traditional development, or combining legacy applications with microservices. To bridge traditional and software-defined infrastructure, SUSE has built a multimodal operating system – SUSE Linux Enterprise 15."

With this release, SLES also accomplishes its modular architecture. Customers don't have to concern themselves with different versions of SLE for different workloads: There is only one installer; there is only one codebase. Users can install the desired version depending on the workload.

SLES 15 is complemented by two other components from the SUSE product line – SUSE Manager 3.2 and SUSE Linux Enterprise High Performance Computing 15, with a focus on helping customers innovate in this era of rapid digital transformation while meeting the needs of multimodal IT.

© iqoncept, 123RF.com

SUSE said in press releases that the latest release of SUSE Manager delivers new features focused on lowering costs, improving DevOps efficiency, and easily managing large, complex deployments across IoT, cloud, and container infrastructures. SUSE Manager also

MORE ONLINE

Linux Magazine
www.linux-magazine.com

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

Building Containers with HPC Container Maker • Jeff Layton
Building HPC applications for production systems is never easy, especially when containers are involved, but with Python and HPC Container Maker, you can describe the container you want quickly and easily without having to worry about the details.

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

Malware Analysis in the Sandbox
Matthias Wübbeling
In malware analysis, a sandbox can provide insight into the software and its run-time environment. While a sandbox can prevent the execution of malicious code with built-in detection mechanisms, malware developers can use countermeasures to take advantage of those same detection mechanisms.

Platform Independence with PowerShell Core
Thomas Wiefel
Microsoft has broken new ground with the release of PowerShell Core 6.0, which at heart is a complete reboot in terms of architecture and objectives. For the first time, a new version is not linked to the Windows operating system.

Protecting Samba File Servers in Heterogeneous Environments • Stefan Kania
Because Samba can be integrated easily into heterogeneous environments, a kind of heterogeneous administration is often necessary, and security falls by the wayside. We show you how to use a Samba file server securely in heterogeneous environments.

ADMIN DevOps Focus
http://www.admin-magazine.com/DevOps

Infrastructure as Code with Terraform
Chris Binnie
With the Terraform configuration management tools and the Amazon Route 53 DNS service, you can configure AWS to provide geographically diverse failover between two web servers.

helps customers improve DevOps efficiency and meet compliance requirements with a single tool that manages and maintains everything from edge devices to Kubernetes environments. SUSE Manager makes managing large, complex deployments easier with new extended forms-based UI capabilities.

GitLab Drops Pricing After Microsoft GitHub Acquisition

As the news broke that Microsoft was acquiring GitHub, panicked users started to move their accounts to GitLab, a fully open source implementation of Linus Torvalds' Git.

Many leading figures of the open source world argues that GitHub is actually now in a more accountable and reliable position compared to earlier, because Microsoft will be treading carefully so as to not stain the positive image the company has been building with the open source community. However, that didn't stop users from moving away from GitHub. Sensing an opportunity, GitLab dropped pricing for its self-hosted GitLab Ultimate plan and its hosted Gold plan; both plans are now available for free to open source projects and educational institutions.

In an interview with Frederic Lardinois of TechCrunch, GitLab CEO Sid Sijbrandij said, "Most education and open source projects don't have access to enhanced security or performance management tools for their software projects. At GitLab, we are happy to have achieved a level of success that allows us to extend the full set of features to these important communities by offering GitLab Ultimate & GitLab Gold plans for free."

A caveat: these prices have been dropped, but these users won't get any commercial support from GitLab like paying users do.

KDE Plasma 5.13 Is Here

The KDE Project has announced the release of Plasma 5.13, the latest version of its desktop environment (*https://www.kde.org/announcements/plasma-5.13.0.php*). KDE is known for its modular design and under-the-hood customization. However, at times these benefits come at the cost of resource efficiency. But as KDE is targeting mobile devices, this release takes advantage of that work and has been optimized to run smoothly on under-powered ARM laptops, high-end gaming PCs, and everything in between. Resource efficiency also means that on powerful machines, more resources will be free for applications instead of being consumed by the desktop itself.

Web browsers are the gateway to the Internet; Plasma 5.13 comes with browser integration that allows users to monitor and control supported browsers, including Chrome/Chromium and Firefox, from the desktop widget. Users will be able to play and pause media playing in web browsers, offering users better control over not only their own entertainment, but also to control annoying autoplaying videos embedded on websites.

The community has also improved the KDE Connect experience; users can now send links directly to a phone using KDE Connect. The Media Control Widget has been redesigned with added support for the MPRIS specification, which means media players can now be controlled from the media controls in the desktop tray or from a phone using KDE Connect.

On the security side, Vaults, Plasma's storage encryption utility, includes a new CryFS back end, better error reporting, a more polished interface, and the ability to remotely open and close vaults via KDE Connect.

KDE already had good multimonitor support, where you could even choose a customized layout for each monitor. The 5.13 release makes it easier to connect external monitors. When a new external monitor is connected, a dialog pops up offering the option to control the position of the additional monitor in correlation to the primary monitor.

The desktop has also received some visual upgrades, from the login screen to icons. Plasma 5.13 will appear in different distributions depending on their own release cycle, but users can test the latest release with KDE's own distribution called "neon" (*https://community.kde.org/Plasma/Live_Images*). openSUSE Tumbleweed and Arch Linux will be among the first to offer this release.

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

Zack's Kernel News

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

By Zack Brown

New NDS32 Port

New Linux ports come and go. Ideally, Linux will run on any piece of hardware that needs it, but some ports lose all their users and are eventually expunged from the kernel, while new hardware might come along, clamoring for a port in the kernel tree, and Linus Torvalds will say no because no one uses that hardware.

This time a promising port appeared on the Linux Kernel Mailing List for the NDS32 architecture. Greentime Hu posted patches that would successfully boot the hardware and would also pass "most LTP-2017 test suites in [the] NDS32 AE3XX platform." Arnd Bergmann liked Greentime's code and approved it for inclusion in the tree, and when Linus asked for some clarity on what the chip was actually for, Arnd said it was a plain low-end RISC architecture, generally used for systems-on-chip (SoC) products, and sat in the same category as ARM32, ARC, MIPS32, RISC-V, and Xtensa architectures.

Greentime added that billions of products had already shipped with his company's hardware, and their customers would get better Linux support if the code were in the main-line tree. In a situation like this, with support from Arnd, a recognizable set of similar chips, and many existing users, a port is likely to go quickly into the kernel, even if – as in this case – the code still barely runs on the hardware. In many other instances, code must be relatively spotless to make it into the tree, but for a port that is unlikely to have any effect on other parts of the kernel, even broken code is often acceptable at first.

Landlock Versus seccomp

Security debates are the hairiest. They often seem to come out of nowhere, and if you lose the debate, your patch is dead. It's not like other situations, where there are a clear set of interested parties and maybe your feature will have to be changed, but at least your use case will probably end up supported in some fashion. With security issues, you might have a perfectly valid need, and it will simply never be met.

Mickaël Salaün posted a patch for Landlock recently that ran into some trouble. The Landlock security module draws a hard line between a given process and the rest of the system. Even a completely compromised process, if it's landlocked, can't escape to infect the rest of the system. The dream is for user processes to landlock themselves, and then, presto, there is no possibility of a security problem.

But there are circumstances – maybe obviously so – where landlocked processes might want to communicate with each other. Mickaël's patch would set up a tightly constrained handshaking mechanism to allow landlocked processes to access each other under certain circumstances. For example, one process might debug another. Without any possibility of interprocess communication, this would remain impossible.

But Andy Lutomirski felt that these patches, and perhaps the entire Landlock module, were redundant with the `seccomp` security module. Instead of implementing a whole new set of security features, he argued, Mickaël should focus his efforts on coding these same features for `seccomp`. Although `seccomp` was originally intended to block system calls, Andy indicated he was open to Landlock-esque extensions.

However, Mickaël pointed out that his was actually the more sophisticated project. As he put it, "Landlock is more complex than `seccomp`, because of its different goal. `seccomp` is less restrictive, because it is more simple."

It did no good. Whatever argument Mickaël made in favor of Landlock – for example adding easy sandboxes to larger applications or creating whole sandbox managers – Andy simply said that if these features were useful, they would be useful as additions to `seccomp`. As Andy put it, `seccomp` was already a living project engaged in exactly this

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**.

sort of thing, and it would be wasteful to duplicate the effort and care going into it. Mickaël did agree that his Landlock features would be valuable for `seccomp`, though he still wanted to keep the projects separate.

It's rough when the gatekeeper for your patch wants you to refocus your efforts on their own pet project, but such things have happened before, and to some extent, that's the way it's supposed to be. Redundant work does need to be identified and its efforts redirected more productively. Sometimes this means a developer must redo many hours of work and accept design decisions and implementation assumptions that they disagree with.

Sometimes developers can disagree so strongly that they will maintain their own alternative for years without backing down. In the late 1990s, Richard Gooch maintained his Devfs project for years against staunch opposition from Greg Kroah-Hartman, who favored the udev project. Eventually the udev project won out, but not before years of flame wars and bitterness were logged forever in the mailing list archives.

New Features from Intel

Intel is not sitting still. While the rest of us suffer slowdowns and security risks because of their longstanding hardware flaws, they are hard at work trying to address those flaws and rebuild their reputation, which is currently a slippery mess on the soles of everybody's shoes.

Reinette Chatre recently posted some patches to support Cache Allocation Technology (CAT). However she couldn't get out of her own way trying to explain the use and value of those features while maintaining deniability regarding whether the technology would exist in future processors.

The upshot of CAT is that it lets users constrain the amount of cache available to a given process. But Thomas Gleixner wanted to know if the feature would still be available in future chips, and if not, what that would mean for applications that relied on it. Would cached data become unavailable to those processes? All Reinette could answer was that the feature was "model specific." Gavin Hindman, also of Intel, tried at one point to clarify, saying that they couldn't guarantee the feature would exist in future chips, although they would like it to.

Thomas really wanted a little more than this, though. If the feature would simply disappear in the future, it might not be worth shoving code into the kernel that would only help a small number of aging systems.

Gavin explained that the future of the patch – and the hardware feature – partly depended on whether the kernel would support it or not. If yes, it might gain more users, and Intel might be inclined to build more chips with that feature. As Gavin put it, "We are in a bit of chicken/egg [situation] where people aren't broadly using it because it's not architectural, and it's not architectural because people aren't broadly using it." Or as Thomas paraphrased, "what you are saying is that 'official' support should broaden the user base, which in turn might push it into the architectural realm."

Ultimately, the question of this and other patches will be decided on a case-by-case basis. Regardless of its damaged reputation, Intel's chips are still absolutely everywhere, and Linux needs to support them. It's good that Intel is posting patches, and it's good that their patches are being considered, but there will likely continue to be a significant amount of tension between the kernel developers and Intel, at least until Intel reveals its ultimate intentions regarding fixing some of its glaring hardware problems. Those intentions can only be revealed as new generations of CPUs roll off the assembly line. It'll be years.

Loading and Unloading Security Modules After Bootup

Sargun Dhillon recently posted some code to allow Security-Enhanced Linux (SELinux) to load and unload Linux security modules (LSMs) after bootup. He implemented this in the form of hooks in the callback chain. A callback chain is a way for code to get a look at how it was called, to determine whether it's running in one context or another. Depending on the context, a given piece of kernel code might behave differently.

Sargun's hooks were not new features. Hooks are general-purpose programming constructs for triggering certain actions at certain times, but his code added the capability to load and unload arbitrary security modules after bootup.

Tetsuo Handa offered some technical suggestions and a patch to apply on top of Sargun's code, mostly renaming things, restricting the scope of some variables, and making other minor changes. However, he also felt that Sargun's hooks could be misused so as to produce unexpected results – specifically, in the case where the user configured a module to no longer be unloadable, but then tried to unload it anyway. To prevent this, Tetsuo ripped out the run-time module-unloading code.

Sargun felt that it might be better to keep the code and instead produce bug alerts and warnings if the user tried something like that, since it would essentially be user error at that point anyway. But Tetsuo felt, in this case, warnings wouldn't be enough. If the user wanted to unload a security module, but was unable to, even if it was their own fault, the situation should trigger a kernel panic and crash the system.

Sargun found the situation interesting and still had doubts. He asked what the SELinux folks themselves would prefer. He said, "What do you think the behavior should be? If `allow_unload_modules`/`allow_unregister_module` is set, do you want to be able to call `security_delete_hooks`? What do you think the right action should be if it fails?"

Stephen Smalley of the NSA said the primary goal should be to avoid changing the behaviors expected by existing users. In other words, a kernel panic would not be best, nor would removing the module unloading code. He explained:

I personally am in favor of killing SELinux support for run-time disable aka CONFIG_SECURITY_SELINUX_DISABLE; the only reason it exists is that Red Hat originally insisted that bootloader configuration is too painful to modify/update on certain platforms and therefore the selinux=0 boot parameter is insufficient as a mechanism for disabling SELinux.

However, we can't break existing users. User space should still attempt to proceed even if run-time disable fails, just with SELinux left in permissive mode and no policy loaded. That generally should work, but does retain the performance overhead of the SELinux

hook function processing, unlike a real disable.

I don't think we particularly care about allow_unload_modules/allow_unregister_module since there is no existing user space or configurations relying on it.

Paul Moore said he also favored removing the SELinux run-time disable code, but he said it wasn't a trivial operation and had a bunch of technical prerequisites before it could be considered, which made it a non-trivial problem.

The discussion ended there. It's not much of a debate, although there were three opposing viewpoints in the discussion. In this particular case, there seems to be support for the overall goal of letting the user load and unload security modules after bootup and also to prevent those things in certain cases, but there are a few little details constraining the exact way the features must be implemented and the way in which they might change over time, as other problems fall away.

An interesting element is the effort to put control directly in the hands of the user. This is in contrast to other "security" projects, which try to put control in the hands of the product vendor and specifically exclude users from altering their systems after bootup.

Splitting Up Security Projects

Sometimes security patches are clandestine efforts to take control away from the user and put it in the hands of the vendor; sometimes they only seem to be trying to do that.

David Howells recently posted some patches to "lock down" the kernel, preventing it from being changed in any way. With his code, it would be possible to stop users from loading and unloading modules, suspend the system to disk, access `/dev/mem` or various registers, or perform direct memory access (DMA). From the user's perspective, he added two new config options – one to allow the user to lock down the kernel or end a lockdown, and the other to make lockdown mandatory and irreversible until the next boot sequence. He added, "if lockdown mode is enabled, the kernel will not be able to use certain drivers as the ability to manually configure hardware parameters would then be prohibited. This primarily applies to ISA hardware devices."

There were a few technical comments and suggestions, and Linus Torvalds also came into the discussion, pointing out his own problems with the direction of this particular area of the kernel. He said:

I think many of these things should simply be done as separate config options.

For example, if the distro is sure that it doesn't need /dev/mem, then why the hell is this tied to "lockdown" that then may have to be disabled because other changes may not be acceptable (e.g., people may need that device DMA, or whatever).

If that /dev/mem access prevention was just instead done as an even stricter mode of the existing CONFIG_STRICT_DEVMEM, it could just be enabled unconditionally.

So none of these patches raise my hackles per se. But what continues to makes me very, very uncomfortable is how this is all tied together.

Why is this one magical mode that then – because it has such a big impact – has to be enabled/disabled as a single magical mode and with very odd rules?

I think a lot of people would be happier if this wasn't so incestuous and mixing together independent things under one name and one flag.

He added:

I would seriously ask that the distros that have been using these patches look at which parts of lockdown they could make unconditional (because it doesn't break machines), and which ones need that escape clause.

He went on:

I get the feeling that not a lot of people have actually been testing this, because "turn off secure boot" is such a universal thing when people boot Linux.

So it's really the whole claim that distributions have been running for this for the last five years that I wonder about, and how often people end up being told: "just disable secure boot."

But if people really don't need DEVMEM/DEVKMEM/DEVPORT, maybe we should just disable them in the default configs and consider them legacy.

I'm just surprised. I suspect a lot of people end up actually using devmem as a fallback for dmidecode, etc. Maybe those people don't boot with EFI secure mode, but if so that just shows that this whole "hardening" is just security theater.

In terms of how many people were just told to disable secure boot, Justin Forbes said (somewhat vaguely), "Very rarely in my experience. And the one time that we sent a kernel to updates testing that was signed with the test key instead of the real key, we had a surprisingly high number of reports from users that it was broken before the update even got synched to mirrors. So we don't have actual numbers of users running active secure boot with Fedora, but we do know it is more than we expected. The majority of people who do run into issues are those running out of tree modules, who haven't imported any sort of key for local signing. This isn't like SELinux was at launch where it was so invasive that a large number of users instinctively turned it off with every installation; I would guess even people who turned it off in the past don't even think about it when they get a new machine and leave it on."

Regarding the possibility of splitting the config options up into separate items, he added, "it might make sense to add separate config options for each of these pieces, which can be unconditionally enabled, and a separate option for secure boot, which selects all of them?"

Linus replied, saying again that he'd love to deprecate `/dev/{mem,kmem,port}` if possible. If they could be separated from the rest of these patches and simply left off, then, Linus said, "they'd *stay* disabled even if somebody cannot handle the other limitations, like DMA, etc."

This seems to be a case where features that seem to be attempts to lock users out of controlling their own system may simply be legitimate security patches, and by splitting the config options out into separate pieces, the kernel will in fact put more control in the hands of the user, who would then be able to make fine-grained decisions about what their particular setup needs and does not need. This would also theoretically put an end to the grinding debates over lockdown and secure boot, which have pitted developers against kernel maintainers for a while now. If these macro-features are designed as multiple smaller options, it might be relatively easy to decide the fate of this or that individual option, rather than wrangle over a given project as a whole. ■■■


OPEN SOURCE BACKUP
CONFERENCE
SEPTEMBER 26, 2018 | COLOGNE
CHECK OUT
WHO'S IN
&
BACKUP
YOUR SEAT!
REGISTER NOW
OSBCONF.ORG

Arduino's cofounder describes the quest for simplicity

Keep It Simple

By Swapnil Bhartiya

I am a massive fan of Arduino. Every year, during Christmas and Halloween, I build some exciting projects for outdoor display. All my projects are powered by Arduino and Raspberry Pi boards. Even my fully open source Original Prusa MK2 3D printer is powered by an Arduino board. When I found that Massimo Banzi, the Arduino project's cofounder, was at the same event I was attending, I reached out to him, and we sat down for an hour-long interview.

Banzi also teaches interaction design at many universities in Europe. That's how Arduino started. Back in 2002, Banzi went to teach at a design school in Northwest Italy called the Interaction Design Institute. The objective of teaching was to apply design principles to the way we interact with things made of digital technologies. It's no longer about rotating dials and pushing buttons. It's about sensors, web interfaces, and touch screens. It could be a challenge to borrow and bring ideas from the physical world into a digital world. As a result, some of these things can be really easy, simple, and pleasurable to use, and some can be very annoying.

"We try to make it simple and easy for people to use things that we design using modern technologies. We worked at trying to figure out how to make electronics simple enough that anyone could pick it up in a few weeks and use it as a creative tool," he said. "I had been working on a number of projects to achieve that, and the last project summarized all of that research. That project was Arduino."

Why Open Source?

Massimo and other cofounders of the project decided to develop Arduino as an open source project. As a long time Linux user, he had seen the power of the open source communities driven by countless passionate users. "When these technologies start to solve problems in people's lives, they look for new ways to use them. You start seeing these technologies in places you didn't even imagine in the first place. Traditionally, it would be very difficult to reach those places with a top-down approach, but open source makes adoption extremely easy."

Initially, Arduino did not manufacture hardware. They were giving away designs and people were building their own hardware. "Then we realized that we needed to simplify this by providing them with ready-made hardware, so they could start creating the project they wanted to do instead of wasting their resources on hardware," said Banzi.

To support hardware manufacturing, marketing, sales, and management, the Arduino project launched a company with the same name. "Everything we do is open source. Whether it's hardware, software, or IDE; even our documentation is released under a Creative Commons license. The only thing that's protected by the company is the brand Arduino. Our technology and knowledge are available for everybody. We make Arduino-branded hardware that supports all the work that we do on the platform. Open source enables users to add features to Arduino that they want in their hardware," he said.

I often come across people from within the desktop Linux community who don't approve of the commercialization of open source. I asked Banzi for his thoughts on it. "A lot of people don't realize all important open source projects, including Linux, have a majority of contributors that are paid by some company to do that work. Yes, there are volunteers who contribute because they use and feel like contributing, but the bulk of the contribution these days comes from paid contributors. So our business model enables us to support the open source side of things," he said.

Arduino Triggered a Revolution

A lot of people never thought that they would be embedded developers, but

they discovered the Arduino and started to build things. "One of the tricks was to basically not tell them that they were embedded developers. Enable them to use electronics as a creative medium," Banzi told me. "The first people who used Arduino were designers, artists, musicians, [and] performers. It started off as a highly creative tool, and then it became a tool for people to also innovate on products. Arduino enabled thousands of people to start using electronics as a tool to innovate and create companies."

Arduino is being used in so many use cases that it's virtually impossible to keep a tab on all of the adoptions. At the beginning of Arduino's history, the goal was to teach people how to use microcontrollers to build things. And then the Internet of Things (IoT) happened. It created a new problem – how to make it easier for people to build IoT projects. At the same time as many other open source projects, single board computers happened. These devices run Linux and Linux, still requires quite a bit of knowledge, especially embedded Linux.

"Recently, I was teaching a class on how to apply machine learning and artificial intelligence creatively, and I realized that we are still in a world where technology is becoming extremely powerful, but the tools that we use are very complex. There is still room to make it all simpler," he said.

Arduino went back to its core mission and decided to create tools that are simple to use – tools that support classic use cases so people can easily build functioning IoT projects without having to understand every single technology along the chain.

"We are going to be releasing a ton of open source software, but our focus is always on the user experience. Our focus is how to make the experience of building an IoT application very simple – all the way from the node that's connected to your sensor, to the edge gateway, to the cloud, to the way you view the data. Everything needs to be seamless and simple to use. So that's one of the things we're focusing on right now," he said.

The result of all this work is a new development platform called Arduino Cloud. "Arduino Cloud started off with us taking our development environment and putting it into the cloud. All you need is just a browser and you can edit code. You can compile the code. You can download it into different Arduino boards," said Banzi, "Then we started to add a section where people can publish very detailed tutorials on how to build things so that if you see someone else's project and you want to build it, just press a button and that becomes a project in your cloud. It allows easy sharing of knowledge and collaboration."

Millions of people are interested in Arduino but don't know how to program. Arduino Cloud provides them with hardware, a platform, and a knowledge base so they can start building IoT projects. Arduino Cloud uses Amazon Web Services (AWS) and is available for free of cost. However, Arduino is also working on a paid version that will offer additional services.

It's a natural evolution for the Arduino project, which started with the goal of keeping it simple. ■■■

Understanding systemd units

On the Unit

Systemd units use files to control resources that Systemd manages. *By Ferdinand Thommes*

Whether you like it or not: Systemd has become ubiquitous. Linux distributions that rely on other init systems are becoming increasingly rare. If you run one of the mainstream distributions, you need to familiarize yourself with the concepts and working methods of systemd.

Systemd *units* and their corresponding configuration files require close attention. The term "unit" means any type of resource that cooperates with Systemd – which includes timers, mountpoints, network resources, sockets, partitions, and devices on top of services.

Configuration units, known as unit files, let you define how and when a service starts, which resources it is allowed to access, and which dependencies need to be met. Unit files are similar in function to the init scripts in SysVinit or Upstart (Figure 1), but they are usually easier to create and easier to maintain. They follow the conventions of simple INI files (Figure 2).

```
root@siductionbox:/lib/systemd/system# cat /etc/rc3.d/S02autofs
#! /bin/sh
#

### BEGIN INIT INFO
# Provides: autofs
# Required-Start: $network $remote_fs $syslog
# Required-Stop: $network $remote_fs $syslog
# Should-Start: ypbind nslcd slapd
# Should-Stop: ypbind nslcd slapd
# Default-Start: 2 3 4 5
# Default-Stop: 0 1 6
# Short-Description: Automounts filesystems on demand
# Description: Automounts filesystems on demand
### END INIT INFO

# Location of the automount daemon and the init directory
#
PROG=automount
DAEMON=/usr/sbin/$PROG
NAME=autofs
PIDFILE="/var/run/$NAME.pid"

test -e $DAEMON || exit 0

PATH=/sbin:/usr/sbin:/bin:/usr/bin
export PATH

. /lib/lsb/init-functions

#
```

Figure 1: Init scripts from SysVinit are monolithic, long, and difficult to read. Autofs contains over 100 lines.

The naming convention for unit files follows the pattern `Name.type`. Table 1 shows a selection of the most frequently encountered types. As you can see, Systemd manages many different unit types.

Many of the unit types work together to extend functionality. Some units are used to trigger other units and activate services and targets. Each type has its own man page – named according to the pattern `systemd.type`.

Table 1: Unit Types

Type	Function
`.service`	Start, monitor and stop services
`.device`	Create device files
`.mount`	Mount and unmount mountpoints
`.automount`	Automatically mount and unmount mountpoints
`.target`	Define a group of units
`.timer`	Define recurring tasks (like Cron)
`.socket`	Establish connections between processes
`.network`	Configure networks
`.path`	Execute service units as a function of changes

```
root@siductionbox:/lib/systemd/system# cat autofs.service
[Unit]
Description=Automounts filesystems on demand
After=network.target ypbind.service sssd.service network-online.target remote-fs.target
Wants=network-online.target

[Service]
Type=forking
PIDFile=/var/run/autofs.pid
EnvironmentFile=-/etc/default/autofs
ExecStart=/usr/sbin/automount $OPTIONS --pid-file /var/run/autofs.pid
ExecReload=/bin/kill -HUP $MAINPID
TimeoutSec=180

[Install]
WantedBy=multi-user.target
```

Figure 2: The autofs unit file is clear, concise, and relatively easy to read.

Distributed

Units are found in several places on the system. Under `/lib/systemd/system/` are files pre-installed by the system. Units you created yourself, or units you edited, are found in `/etc/systemd/system/`. If you want to change an existing unit, it is best to copy it there first and edit it there. Finally, certain units relevant for the runtime are located below `/run/system/system/`. The order of parsing is `/etc/`, `/run/`, `/lib/`.

When you look at the unit files gathered in `/lib/systemd/system/`, you will see files with different extensions that represent the different types. Extensions like `.network`, `.timer`, `.mount`, or `.device` are self-explanatory. However, the service units that determine the behavior of the services on the computer are the most common type (Figure 3).

Tripartite

An example helps to explain the structure of the files. This example is based on the `autofs.service` service shown in Figure 2. `autofs.service` is a service for starting external drives or network shares, and it uses a reasonably clear unit file.

A unit is divided into three sections `[Unit]`, `[Type]`, and `[Install]`; the unit type varies. In this case, it says `[Service]` because it is a unit for controlling a service. Often several unit types belong together, such as a service and a timer. The service file defines the service itself, and the timer controls its recurring execution (Figure 4).

In the example from Listing 1, the directives used with the unit take the form of key-value pairs. This section is typically used to define metadata for the unit and to configure the relationship of the unit to other units.

The `Description` key describes the service. You are free to choose the value, but it should clearly state the purpose of the service. The `After` key contains the services and targets that this service expects. Targets are groups of services (Figure 5).

The last key you see is `Wants`, which you use to denote optional dependencies. You indicate a hard dependency with `Require`. If the service registered there does not start, the service to which this unit belongs fails. If you specify `Wants`, it will still start. Now, if you're wondering what the extra `After` is for: Its absence would mean that both units would start in parallel, which would not make sense in this case.

```
root@siductionbox:/lib/systemd/system# ls
accounts-daemon.service                 rcS.service
acpi-fakekey.service                    reboot.service
acpi-fakekey.socket                     reboot.target
alsa-restore.service                    reboot.target.wants
alsa-state.service                      remote-cryptsetup.target
alsa-utils.service                      remote-fs-pre.target
apparmor.service                        remote-fs.target
apt-daily.service                       remote-fs.target.wants
apt-daily.timer                         rescue.service
apt-daily-upgrade.service               rescue.target
apt-daily-upgrade.timer                 rescue.target.wants
atd.service                             resolvconf.service
auth-rpcgss-module.service              rmnologin.service
autofs.service                          rpcbind.service
autovt@.service                         rpcbind.socket
avahi-daemon.service                    rpcbind.target
avahi-daemon.socket                     rpc-gssd.service
basic.target                            rpc-statd-notify.service
basic.target.wants                      rpc-statd.service
binfmt-support.service                  rpc-svcgssd.service
blk-availability.service                rsync.service
bluetooth.service                       runlevel0.target
bluetooth.target                        runlevel1.target
bootlogd.service                        runlevel1.target.wants
bootlogs.service                        runlevel2.target
bootmisc.service                        runlevel2.target.wants
checkfs.service                         runlevel3.target
checkroot-bootclean.service             runlevel3.target.wants
checkroot.service                       runlevel4.target
clean-mount-point@.service              runlevel4.target.wants
```

Figure 3: Unit files for services ending in .service are the most common.

```
root@siductionbox:/etc/systemd/user# systemctl list-unit-files --type timer
UNIT FILE                    STATE
apt-daily-upgrade.timer      enabled
apt-daily.timer              enabled
etckeeper.timer              enabled
fstrim.timer                 enabled
snapd.refresh.timer          enabled
snapd.snap-repair.timer      enabled
systemd-tmpfiles-clean.timer static

7 unit files listed.
```

Figure 4: Self-created timers like etckeeper and fstrim complement the timers created by Systemd.

Listing 1: Unit Section

```
[Unit]
Description=Automounts filesystems on demand
After=network.target ypbind.service sssd.service network-online.target remote-fs.target
Wants=network-online.target
```

```
root@siductionbox:/lib/systemd/system# cat multi-user.target
#  SPDX-License-Identifier: LGPL-2.1+
#
#  This file is part of systemd.
#
#  systemd is free software; you can redistribute it and/or modify it
#  under the terms of the GNU Lesser General Public License as published by
#  the Free Software Foundation; either version 2.1 of the License, or
#  (at your option) any later version.

[Unit]
Description=Multi-User System
Documentation=man:systemd.special(7)
Requires=basic.target
Conflicts=rescue.service rescue.target
After=basic.target rescue.service rescue.target
AllowIsolate=yes
```

Figure 5: Targets group several units and may themselves be based on other targets. The AllowIsolate=yes directive allows this target to act as a boot target.

In the `Unit` section, you can use further keywords like `Description`, `Documentation`, `BindsTo`, `Conflicts`, `Condition`, `Assert` and others. For more information, see the `systemd.units` man page.

The `[Service]` section plays a central role (Listing 2). First, you need to define the service type. The default is `Type=simple`. The `simple` type means that the service does not fork after startup.

The example in Listing 2 is a `forking` type service. Systemd considers the service to have started as soon as the process disappears into the background and the higher-level system terminates. This type is often used for legacy daemons. You need to specify the key-value pair `PIDFile=File`, so that the system can continue to follow the main process.

You might encounter a few other service types when you look at the unit files on your computer. `Type=oneshot` is used for scripts that do a single job and then exit. `Type=notify` is similar to `Type=simple`, with the difference that the daemon sends a signal to Systemd when it is ready. In the case of `Type=dbus`, the service is considered ready if the specified `BusName` appears on the D-bus system bus. In the case of `Type=idle`, Systemd delays the execution of the service until it has completed all other pending jobs.

Another key from this example, `EnvironmentFile=`, appears in the `[Service]` section. This value refers to a file from which the service loads environmental variables if required. `ExecStart=` contains the command that the system executes when the unit starts, and `ExecReload=` reloads the configuration of the service if necessary.

Finally, `TimeoutSec=` specifies the maximum time the service will run. All service types and keys for the `[Service]` section are explained in detail in the `systemd.service` man page.

The `[Install]` section in this example has only a single entry (Listing 3). The key-value pair used here is present in almost every service file. The `WantedBy` key determines when the unit starts. The `multi-user.target` value is the default for a multi-user system. These targets correspond to the run levels in SysVinit, where `multi-user.target` stands for run level 3.

You can determine the run level in Systemd using `systemctl get-default` in the terminal. In a graphical environment, it goes by the name of `graphical.target` (SysVinit: Run-Level 5). All available run levels (or, more correctly, targets) are displayed by the command

```
ls -al /lib/systemd/system/runlevel*
```

See Figure 6.

Listing 2: Service Section

```
[Service]
Type=forking
PIDFile=/var/run/autofs.pid
EnvironmentFile=-/etc/default/autofs
ExecStart=/usr/sbin/automount
ExecReload=/bin/kill -HUP $MAINPID
TimeoutSec=180
```

Listing 3: Install Section

```
[Install]
WantedBy=multi-user.target
```

Author It Yourself?

You may have wondered what practical reason (besides pure curiosity) would make you want to create a Systemd unit.

Suppose you install some software that runs a service that only has an init file, but not a unit file: It makes sense to create a unit for it yourself. A good starting point is the simple template in Listing 4.

A useful way to fill the `[Service]` section is to look at similar existing unit files. The `systemctl list-unit-files` command creates a list of all unit files on the system with their current statuses. You can restrict this list to services by specifying `systemctl list-unit-files --type service`.

However, the first command can return other interesting unit types of the same name. These could be target units, for example: They are used to link and group other units to describe a

```
root@siductionbox:/lib/systemd/system# ls -al /lib/systemd/system/runlevel*
lrwxrwxrwx 1 root root   15 Apr  1 13:02 /lib/systemd/system/runlevel0.target -> poweroff.target
lrwxrwxrwx 1 root root   13 Apr  1 13:02 /lib/systemd/system/runlevel1.target -> rescue.target
lrwxrwxrwx 1 root root   17 Apr  1 13:02 /lib/systemd/system/runlevel2.target -> multi-user.target
lrwxrwxrwx 1 root root   17 Apr  1 13:02 /lib/systemd/system/runlevel3.target -> multi-user.target
lrwxrwxrwx 1 root root   17 Apr  1 13:02 /lib/systemd/system/runlevel4.target -> multi-user.target
lrwxrwxrwx 1 root root   16 Apr  1 13:02 /lib/systemd/system/runlevel5.target -> graphical.target
lrwxrwxrwx 1 root root   13 Apr  1 13:02 /lib/systemd/system/runlevel6.target -> reboot.target
```

Figure 6: The runlevels used in previous init systems find a counterpart in Systemd targets.

LIBRECON

powered by CEBIT®

THE REFERENCE EVENT FOR OPEN TECHNOLOGIES IN THE SOUTH OF EUROPE

21+22 Nov · 2018
Bilbao (Spain)

LATEST TRENDS IN OPEN SOURCE APPLIED TO:

INDUSTRY

The so-called fourth industrial revolution is written in open source.

PUBLIC AUTHORITIES

The most transparent administrations are based on open technologies.

FINANCE

The global stock exchanges have begun to adapt their technology to the open source.

JOIN UP to the unique event powered by CEBIT in the South of Europe

ORGANIZED BY:

[esle] ASOLIF

POWERED BY:

CEBIT® Deutsche Messe

FURTHER INFORMATION:

www.librecon.io

info@librecon.io
twitter.com/librecon
facebook.com/librecon
es.linkedin.com/in/librecon

Listing 4: A Template

```
[Unit]
Description=My_Unit
Documentation=man:<i>optional reference to man files<I>
After=<I>starts after XY<I>
Wants=optional <I>dependencies<I>

[Service]
<I>Key-value pairs for the unit type in question<I>

[Install]
WantedBy=multi-user.target
```

desired state of the system. Some of these units would then be services, others perhaps additional targets with their own groups of units.

Users who rely on KDE Plasma enjoy greater convenience: This desktop lists all units and their statuses clearly in the system settings below the *System* item. You can also use a combo box to filter by unit type or use a search function (Figure 7).

To learn more about a unit similar to the one you are creating, use the `systemctl status service_name` command in addition to the unit file. This command provides information about runtime, memory usage, process ID (PID), and possible error messages since the last start of the service (Figure 8).

Once you have finished the unit file and put it in the right place, use `systemctl status service_name` to check the status. Additional information may be provided by the `journalctl --unit=service_name` command.

To enable and start the service, use the `systemctl start service_name` command. The `systemctl` man page provides additional commands and options.

Conclusions

Creating systemd units yourself is not rocket science. What may seem a little confused at first glance is actually a well organized combination of interacting components. Compared to the `init` files in SysVinit, system units are easier to create and maintain. In particular, unlike with some other init systems, you do not need to be familiar with a scripting language to create unit files, since these are simple INI files.

Plenty of documentation is available for Systemd in general and system units in particular. The DigitalOcean [1] website provides a good overview of the relationships between units. Other reliable points of contact on Systemd and its components are Freedesktop [2] and Systemd-creator Lennart Poettering's blog site "Pid Eins" [3]. ■■■

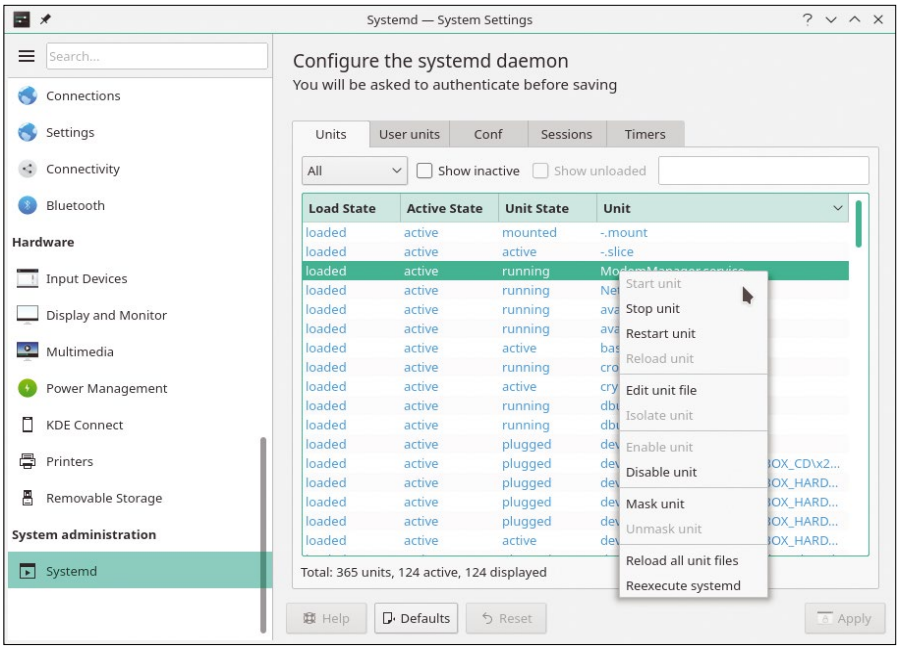


Figure 7: The Systemd menu item in the KDE Plasma settings provides an overview of units running on the system and their current statuses. You can use the context menu to control units, as you would with the systemctl command-line utility.

TIP

When determining the optimal start time for a service, take a look at the output of `systemd-analyze blame`, which breaks down the boot process.

Author

Ferdinand Thommes lives and works as a Linux developer, freelance writer, and tour guide in Berlin.

Info

[1] DigitalOcean: *https://www.digitalocean.com/community/tutorials/understanding-systemd-units-and-unit-files*

[2] Freedesktop: *https://www.freedesktop.org/wiki/Software/systemd/*

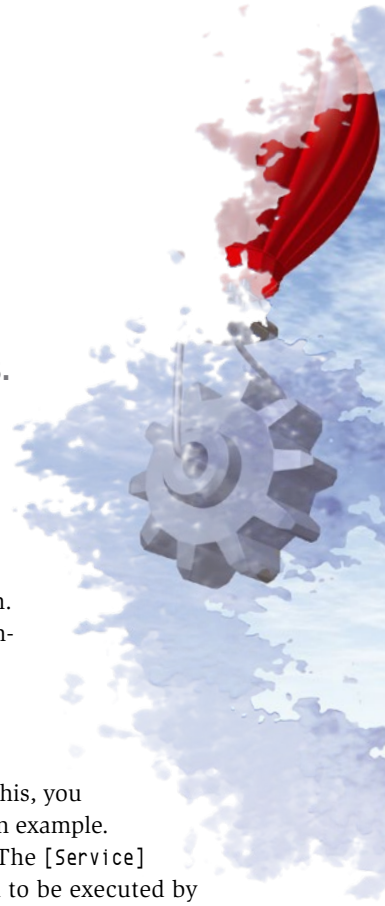
[3] Systemd blog: *http://0pointer.de/blog/projects/systemd-for-admins-1.html*

```
root@siductionbox:/etc/systemd/user# systemctl status autofs.service
● autofs.service - Automounts filesystems on demand
   Loaded: loaded (/lib/systemd/system/autofs.service; enabled; vendor preset: enabled
   Active: active (running) since Tue 2018-05-08 06:03:53 CEST; 1 weeks 6 days ago
 Main PID: 1292 (automount)
    Tasks: 4 (limit: 4915)
   Memory: 7.2M
   CGroup: /system.slice/autofs.service
           └─1292 /usr/sbin/automount --pid-file /var/run/autofs.pid
```

Figure 8: Querying the status of a unit facilitates troubleshooting.

Triggering regular tasks with Systemd

Alarm Clock



Systemd can start timers that automatically perform tasks at specified times. The configuration files are known as timer units. *By Tim Schürmann*

You might want to use your Linux system to automatically create a backup every evening and rotate the log files at regular intervals. In most distributions, time-controlled tasks are handled by the Cron daemon. But Systemd is an interesting alternative to Cron. Systemd controls the startup process of most distributions, and it can also trigger time-controlled and recurring tasks.

Service Providers

The first task is to tell Systemd which task to perform. To do this, you create a configuration file, the Service Unit. Listing 1 shows an example.

A service unit is a text file divided into several sections. The `[Service]` section is required. `ExecStart=` is followed by the command to be executed by the system. In Listing 1, Systemd would simply run a script that backs up the system to the `/mnt` directory. The `[Unit]` section adds some metadata. In the simplest case, `Description=` is followed by a description of the task.

Service Units usually tell Systemd which services to boot when the system starts. (See the article on Systemd units elsewhere in this issue.) Systemd also supports additional sections and settings. However, since the system just needs to schedule the task, these settings are not (absolutely) necessary. In particular, you can leave out the complete `[Install]` section.

Save the newly created service unit to `/etc/systemd/system`. The filename corresponds to the (internal) name of the service unit. It must be unique among all service units and end with `.service`, as in `backup.service`. Systemd can also start existing service units or service units supplied by the distribution on a time-controlled basis. In this case, simply make a note of the filename of the service file.

Tick-Tock

To avoid burning the cake to a crisp, most hobby bakers set a kitchen timer. In a similar way, you need to set a separate timer for a task you wish to assign to Systemd.

First, create a new text file in the `/etc/system/system` subdirectory. The text file should have the same filename as the service unit you created earlier, but it ends with `.timer`. In the example, the file would be named `backup.timer`. In Systemd speak, the file with the `.timer` extension is known as the timer unit. In the timer unit, you describe when the timer should "go off," at which point, Systemd will start the backup.

Listing 1: Service Unit

```
[Unit]
Description=Create a backup of the system

[Service]
ExecStart=/usr/bin/backup.sh /mnt
```

The structure of a timer unit is very similar to that of a service unit. As the example from Listing 2 shows, it typically consists of three sections: `[Unit]` is followed by general information about the timer. In Listing 2, this information would include a `Description=`

that serves mainly as a reminder for the user. Make a note on why the timer exists and what actions it triggers.

Current Events

In the next section, `[Timer]`, you tell Systemd when to start the task. Make a note of this time after `OnCalendar=` in the notation *weekday year-month-day hour:minutes:* seconds. The setting `OnCalendar=Fr 2018-11-30 12:00:00` tells Systemd to create the backup on Friday, November 30, 2018 at noon precisely. You can omit unnecessary information, such as the day of the week or the seconds.

Normally, you will not want Systemd to run the task once only, but repeat it. To set up a repeating event, you can simply list the corresponding days, dates, and times separated by commas. In the example from the first line of Listing 3, Systemd starts the backup November 30, 2018 at 1AM and 12 Noon.

Listing 2: Timer Unit

```
[Unit]
Description=Create a daily backup of the system

[Timer]
OnCalendar=*-*-* 18:15:00
Persistent=true
RandomizedDelaySec=2h

[Install]
ZWantedBy=timers.target
```

You can also abbreviate the number ranges with two dots `..`, which means that you do not have to list all the months, for example. The entry from the second line of Listing 3, tells Systemd to take action on the first day of each month. If the statement applies to all months, you can also use the wildcard `*` (line three).

The `*-*-*` entry from Listing 2 tells Systemd to run the backup every day at 18:15 in every month and every year.

Extremely Hesitant

If the computer is not running at the selected time, Systemd cannot create a backup. In Listing 2, the `Persistent=true` setting ensures that Systemd catches up with the task as quickly as possible in such situations. However, if several actions start simultaneously, they can slow down the system or even interfere with each other.

To prevent a traffic jam, Systemd randomly delays execution by a few seconds if necessary. The maximum number of seconds it can wait before executing is stated after `RandomizedDelaySec=`. Systemd interprets the number as minutes for a trailing `m` and as hours for an `h`. In Table 1, you will find all other supported time units; you can also combine these. Systemd would delay the backup by a maximum of 90 seconds if you state `RandomizedDelaySec="1m 30s"`.

Repetitions

Systemd lets you schedule a task to occur at some recurring interval without specifying an exact time – for example, every 15 minutes or once a week. Use the `OnCalendar=weekly` option to start a weekly backup. In addition to `weekly`, you'll find options for `minutely`, `hourly`, `daily`, `monthly`, `yearly`, `quarterly`, and `semiannually`.

If you want to run a task 15 minutes after system startup, use the following settings instead of `OnCalendar=...`:

```
OnBootSec=15m
OnUnitActiveSec=1w
```

`OnBootSec=` specifies how many seconds after system startup Systemd should execute the task. In the example, the timer

Listing 3: Date and Time

```
OnCalendar=2018-11-30 01,12:00:00
OnCalendar=2018-01..12-01 01,12:00:00
OnCalendar= 2018-*-01 01,12:00:00
```

Table 1: Units Used by Systemd

Unit	Long forms	Meaning	Example
s	seconds, second, sec	second	5s
m	minutes, minute, min	minute	10m
h	hours, hour, hr	hours	2h
d	days, day	day	7d
w	weeks, week	week	2w
M	months, month	month	6M
y	years, year	year	4y

Table 2: Monotonic Timers

Setting	Refers to the moment when…
OnActiveSec=	… the timer was activated.
OnBootSec=	… the computer was booted.
OnStartupSec=	… Systemd started.
OnUnitActiveSec=	… the unit that activates the timer was last activated.
OnUnitInactiveSec=	… the unit that activates the timer was last deactivated.

goes off 15 minutes after the system startup. The second setting, `OnUnitActiveSec=`, tells Systemd the time intervals at which it should repeat the task. In the example, Systemd would run the backup 15 minutes after system startup and then every week.

With both settings, you can use the units from the Table 1 and combine the information. For example, the `OnBootSec="5m 30s"` setting would execute the task five and a half minutes after system startup.

If a timer is based on a (calendar) date, as per Listing 2, it is known as a "Calendar Timer." If, on the other hand, a timer starts after a specified period of time relative to an event, such as a system start, Systemd refers to it as a "monotonic timer." Such timers work independently of the time zone.

The timer is not only triggered shortly after system startup, but also responds to other events listed in Table 2. As in the previous example, several settings can be combined with each other; each setting must have its own line.

Relationship Helper

The `systemd-analyze` tool helps you figure out the correct times. If you pass it the `calendar` parameter, `systemd-analyze` converts the relative time specifications into other formats (Figure 1). The following command tells you, for example, which day of the week `weekly` corresponds to:

```
$ systemd-analyze calendar weekly
```

By default, Systemd guarantees one-minute timer accuracy. You can therefore expect the backup not to start punctually at 6:00 pm, but at 6:01 pm. If you need greater accuracy, add the line `AccuracySec=30s` to the `[Timer]` section. The time specification determines the desired accuracy; in the example, the action would be no later than 30 seconds after the assigned date. For such time entries, you can again use the units from Table 1.

```
tim@ubuntu:~$ systemd-analyze calendar weekly
  Original form: weekly
Normalized form: Mon *-*-* 00:00:00
    Next elapse: Mon 2018-05-28 00:00:00 CEST
       (in UTC): Sun 2018-05-27 22:00:00 UTC
       From now: 5 days left
```

Figure 1: A timer starting weekly would execute at midnight every Monday. The next event will be in exactly five days.

Timers also let you wake up the computer from suspend mode on a time-controlled basis. To do this, add the line `WakeSystem=true` to the `[Timer]` section. Systemd only wakes the system when it is in sleep mode and if the hardware and the BIOS/UEFI of the computer support the process. Systemd is currently unable to put the computer to sleep on a time-controlled basis.

Systemd assigns the timer unit to the appropriate service unit based on the filenames. In the example, the timer `backup.timer` automatically starts the command from the service unit `backup.service`. Alternatively, in the `[Timer]` section, you can explicitly specify the name of the service unit that you want Systemd to execute using the `Unit=` setting. This is especially useful if you want to start an existing service unit with a new timer.

Winding Up the Clock

If you want Systemd to activate the timer directly at system startup, you need an `[Install]` section in the timer unit. The `WantedBy=` setting tells which other units the timer should start with. In Listing 2, the `WantedBy=timers.target` setting ensures that Systemd starts the timer together with all other timers at the regular system startup time.

If you want Systemd to start the timer at startup time, you have to enable it explicitly (Listing 4, first line). Alternatively, you can start the timer manually (second line). All currently configured timers are listed by the `systemctl list-timers` command (Figure 2).

In the table under `Next`, you can read when the system timer will execute the task the next time. The time remaining until then is in the `Left` column. Similarly, you can see under `Load` when `systemd-timer` last executed the task. How long ago that was is shown in the `Passed` column. Under `Unit`, you will find the name of the corresponding timer and thus its configuration file.

You can end the display by pressing [Q]. By default, Systemctl only presents timers that are currently enabled. You can display the inactive timers on screen by appending the `--all` parameter.

```
NEXT                         LEFT          LAST                         PASSED       UNIT
Wed 2018-05-23 00:02:37 CEST  35min left    Tue 2018-05-22 23:02:02 CEST  24min ago    anacron.timer
Wed 2018-05-23 01:10:51 CEST  1h 43min left Tue 2018-05-22 11:00:48 CEST  12h ago      apt-daily.timer
Wed 2018-05-23 06:55:48 CEST  7h left       Tue 2018-05-22 11:00:48 CEST  12h ago      apt-daily-upgrade.timer
Wed 2018-05-23 11:32:51 CEST  12h left      Tue 2018-05-22 23:18:02 CEST  8min ago     motd-news.timer
Wed 2018-05-23 22:33:44 CEST  23h left      Tue 2018-05-22 22:33:44 CEST  53min ago    systemd-tmpfiles-clean.timer
Mon 2018-05-28 00:00:00 CEST  5 days left   Mon 2018-05-21 00:00:13 CEST  1 day 23h ago fstrim.timer

6 timers listed.
Pass --all to see loaded but inactive timers, too.
lines 1-10/10 (END)
```

Figure 2: Systemctl displays all timers currently running. The display requires the widest possible terminal; alternatively, you can use systemctl list-timers --no-pager to output the information to the standard output.

Listing 4: Enabling at Startup

```
$ systemctl enable backup.timer
$ systemctl start backup.timer
```

Listing 5: Manual Stop

```
$ sudo systemctl stop my.timer
$ sudo systemctl disable my.timer
```

Listing 6: Examples

```
$ systemd-run --on-active=30m /usr/bin/backup.sh /mnt
$ systemd-run --on-calendar=weekly --unit backup.service
```

Snooze Button

If required, each timer can be stopped manually (Listing 5, first line) and disabled (second line). The manpage [1], which goes by the name of `systemd.timer`, provides explanations for all presented settings. `man systemd.time` provides further information on the format of dates and times and offers numerous additional examples.

Short-Term Alarm

If you want Systemd to make a single backup in exactly 30 minutes, use `systemd-run`. The command looks like the first line of Listing 6. The `/usr/bin/backup.sh /mnt` command appended there is executed by Systemd at the specified time. Use the parameter `--on-active` to tell it the waiting time.

The time units again correspond to those in Table 1. In the example, Systemd interprets the `30m` as half an hour. Alternatively, use `--on-calendar=` to enter a specific date. The details are again provided in the same way as in the timer unit. With appropriate time specifications such as `weekly`, the action can execute repeatedly.

```
tim@ubuntu:~$ systemd-run --on-active=2m /usr/bin/backup.sh /mnt
Running timer as unit: run-u127.timer
Will run service as unit: run-u127.service
```

Figure 3: The timers generated by systemd-run have cryptic names that typically do not indicate the task solved by the timer.

In any case, `systemd-run` creates a new timer in the background without you needing to create a service file (Figure 3). If a suitable service unit already exists, you can alternatively let `systemd-run` launch it. To do this, simply pass in the name of the service unit using the `--unit` parameter. The example from the second line of Listing 6 starts the task stored in the `backup.service` service unit every week.

The timers generated by `systemd-run` only exist temporarily. If you use the `--on-active` parameter, the timer disappears immediately after the action has been executed; in any case, it disappears after rebooting the system. `systemd-run` only creates a timer for a service unit if no suitable timer unit exists.

Conclusions

Compared to good old Cron, the timers from Systemd offer a number of benefits. Systemd is now included with most distributions. The actions can also be started independently of the timer, executed in a very specific environment, assigned to groups, and made dependent on other units.

However, as the simple example from Listing 2 shows, the configuration overhead is significantly higher for Systemd timers. A short line is all it takes with Cron, but with Systemd, you first have to write a complete timer unit. Furthermore, unlike Cron, Systemd cannot send you an email in case of an error and thus draw attention to problems. Systemd timers should therefore not replace Cron, at least not in the near future, but merely supplement it. ■■■

Info

[1] Manpage for Systemd timer units: *https://www.freedesktop.org/software/systemd/man/systemd.timer.html*

shop.linuxnewmedia.com

Manage systemd with graphical tools

Command Center

Graphical frontends make it easier to take full advantage of the Systemd process manager. We examine some leading tools for the KDE environment. *By Erik Bärwaldt*

For administrators as well as power users, the Systemd init daemon [1] requires training due to its many options and innovations. However, if you are in a hurry – or if you are accustomed working within a GUI interface – you can speed up the learning curve by using one of Systemd's graphical frontends. Systemd has several GUI options, each with its own range of functions. Most of the Systemd GUIs are based on established desktop environments such as KDE Plasma or Gnome, but cross-desktop solutions are also available.

Kcmsystemd [2], Systemd-kcm [3], and SystemdGenie [4] are three powerful tools for managing Systemd in KDE and other Qt-based desktop environments. In addition to displaying current system status, these frontend tools serve as a convenient interface for viewing and modifying information stored in Systemd configuration files (see the box entitled "Background.")

Trilateral Test

Kcmsystemd and Systemd-kcm were originally developed for the KDE desktop. Kcmsystemd was designed for KDE SC 4.x interfaces, and Systemd-kcm was created for

Background

As you will learn in the other articles in this issue, the Systemd universe is organized into entities called *units*. Systemd units initialize individual processes, mount drives, manage swap memory, and create new devices. Systemd is capable of parallelizing the individual units. Simultaneous execution of several tasks significantly speeds up system startup, especially on hardware with modern multicore processors.

The file extension tells you which unit fulfills which task: The `.service` extension describes service units that manage processes – similar to the init scripts of SysVinit. Units with the `.mount` extension manage drives and filesystems. Units ending in `.device` create device files, and units ending in `.timer` handle recurring tasks. There are also `.socket` and `.target` units: `.target` combine individual units and can therefore be compared with the runlevels under SysVinit, while the `.socket` units open individual network sockets.

Since the configuration information of each unit resides in text files, each unit can be modified using the `systemctl edit --full Unit_name` command. To view the central log file of the system, you do not open the `/var/log/messages` file – which was the case on SysVinit systems. Instead, Systemd provides the `journalctl` command, which you can use to open the log file created in binary format. Parameters of the command can be used to view individual subareas.

Plasma 5 environments. SystemdGenie is similar to Systemd-kcm in its scope of features, but it works as an individual application.

Functionally and visually, the two tools integrated in KDE differ only slightly, whereas SystemdGenie differs from both of the others. All three tools are also available for other desktop environments based on the Qt libraries, such as LXQt.

Kcmsystem

After launching, Kcmsystemd comes up with a simple and clear program window: In the main segment, all processes of the system appear in a multicolumn tabular view in the *Units* tab, which is active by default. Use this tab to view the status of the individual processes (Figure 1).

To display only certain unit types, limit the display to the desired option by selecting an entry in the *All unit types* checkbox top-left in the program window. If you also want inactive units to appear in the list, check *Show inactive*. Inactive units now appear in red; all other processes are in green.

Info Center

Kcmsystemd shows detailed information about each unit as soon as you click on the process in the list view. The processes appear in the form of a table in the lower window segment. Below them are four switches, not all of which can be activated, depending on the selected process. The switches allow you to stop or start the relevant unit, initiate a restart of the process, or reload the configuration.

If you hover over a unit line for a while with your mouse pointer, a window opens, displaying information about the dependencies of the current process. The dialog lists dependencies for the current unit, as well as for units that require the selected process.

You can also edit the text files used for configuring the individual units. Right-click the unit, and a context menu appears. Select the first option *Edit unit file*. After entering the root password, KWrite, the default text editor under KDE, opens and lets you edit the configuration file (Figure 2). (BTW: This function failed under OpenSuse Leap 42.3 due to an incorrect path entry for the text editor, which cannot be changed manually.)

Since incorrectly configured units can have considerable consequences for the system, it is a good idea to read the documentation [5] before editing a configuration. To enable the changes, restart the unit after editing.

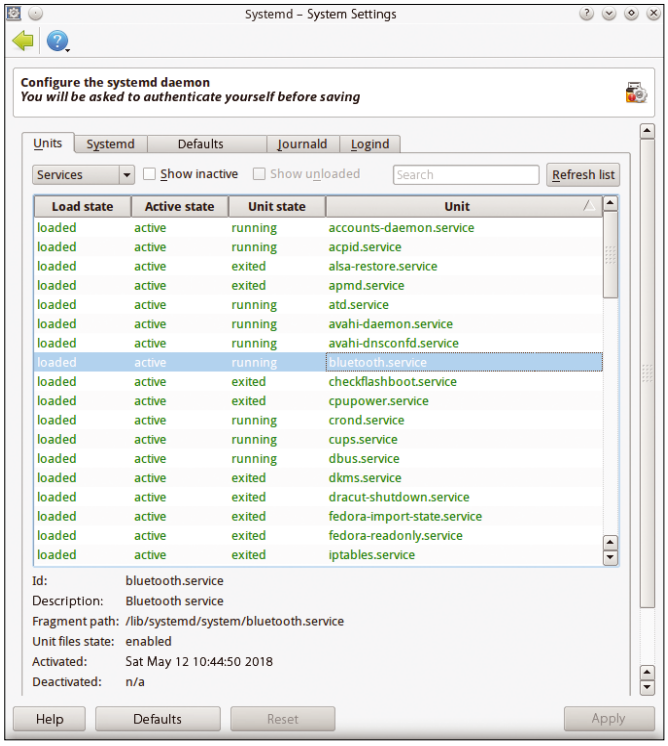


Figure 1: Kcmsystemd summarizes process information in a well organized list.

```
[Unit]
Description=Bluetooth service
Documentation=man:bluetoothd(8)

[Service]
Type=dbus
BusName=org.bluez
ExecStart=/usr/lib64/bluetooth/bluetoothd
NotifyAccess=main
#WatchdogSec=10
#Restart=on-failure
CapabilityBoundingSet=CAP_NET_ADMIN CAP_NET_BIND_SERVICE
LimitNPROC=1

[Install]
WantedBy=bluetooth.target
Alias=dbus-org.bluez.service
```

Figure 2: The Kcmsystemd editor lets you adapt individual configuration files.

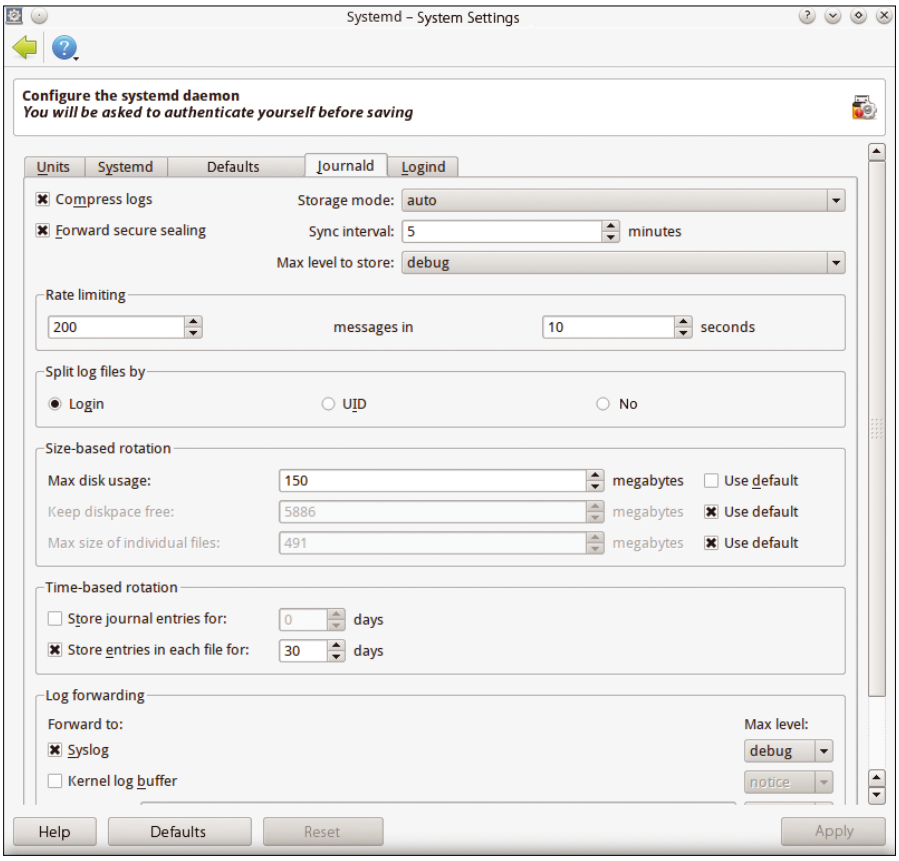


Figure 3: Kcmsystemd offers several options for configuring the journal.

All Encompassing

Both Kcmsystemd and Systemd-kcm have additional tabs that let you to configure Systemd further. The *Systemd*, *Preferences*, *Journald*, and *Logind* tabs are arranged horizontally from left to right in the program window. The *Systemd* tab lets you configure individual settings for logging, starting the system, and dealing with a crash; under *Preferences*, you will find options for timing when to start and stop individual units.

The *Journald* tab contains setting options for the journal. Systemd keeps journals, but, unlike SysVinit, Systemd's journals are in binary form. Therefore, you can't view the entries using conventional commands such as `cat`, `tail`, or `less`, but only with the help of Journalctl.

In the *Journald tab*, you can define the rentention period for the journal entries, the maximum size of the journal file, and whether the system compresses it. In this section, you also specify the maximum number of entries in the journal within a defined period (Figure 3). You can avoid oversized and therefore confusing journal files by making sensible settings.

The *Logind* tab offers some settings of the login daemon for virtual terminals and for power management. Changes to these settings require a system reboot.

Systemd-kcm and SystemdGenie

At first glance, the Systemd-kcm and SystemdGenie frontends, which were developed for the Plasma 5 desktop, differ only slightly from the KDE-SC 4.x version: They also provide a tabular list of the processes, along with selection and search fields.

SystemdGenie provides the buttons for reloading the configuration and starting and disabling units in the *Table* View, rather than at the bottom of the window. The tool also offers a menu line and uses a different tab structure.

Both of the newer Systemd GUIs retain the overlapping information window when you hover the mouse pointer over a unit. However, the window displays significantly less information than the previous version (Figure 4).

With Systemd-kcm, you can control the individual units via a context menu that opens after a right-click on the unit. You can also edit the configuration file, not in KWrite, but using an integrated, rudimentary editor.

The System-KCM developers have also modified the tab structure: The *Setup* section summarizes the configuration of the login and journald processes. After activating the tab, you

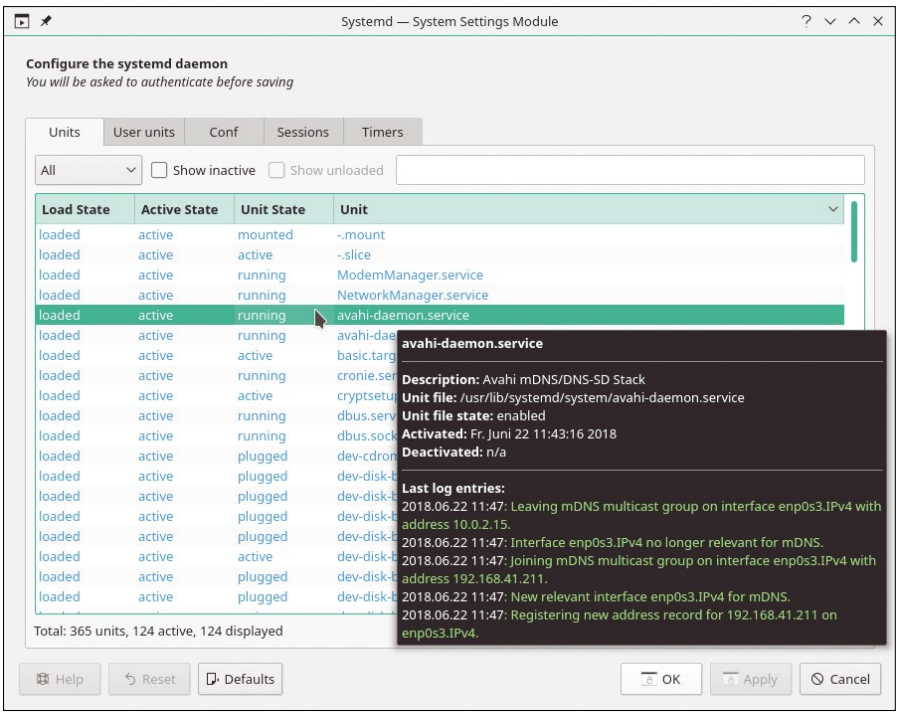


Figure 4: The software displays information on the active unit in an overlapping window.

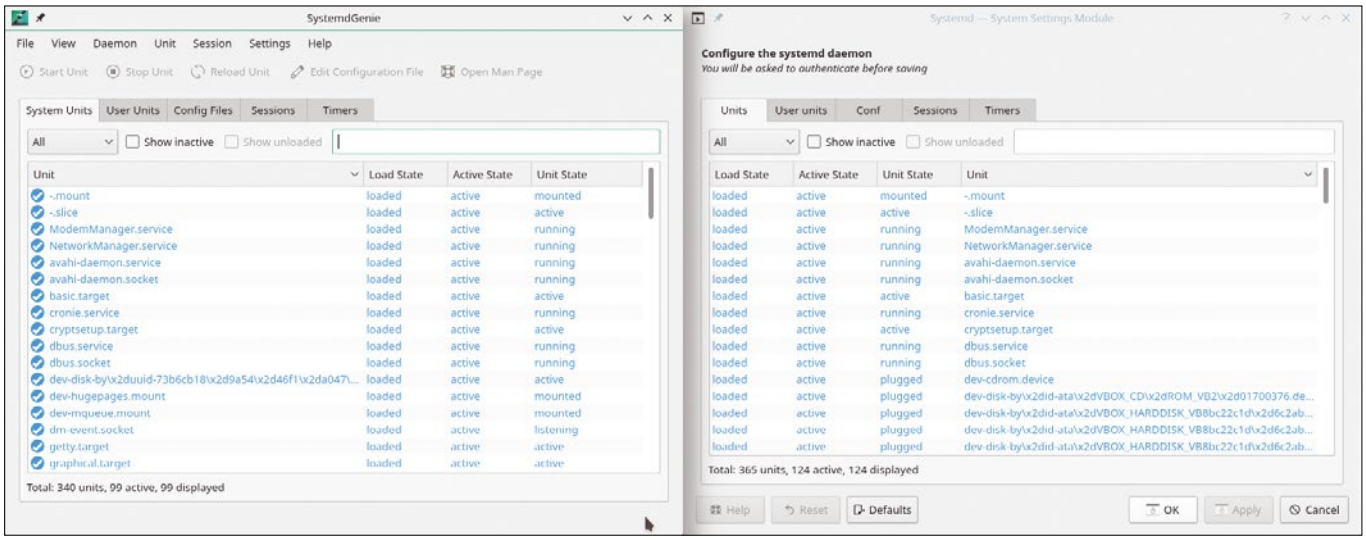


Figure 5: SystemdGenie and Systemd-KCM look almost like identical twins.

select the desired configuration file from a selection field in the upper-right corner. A table of the existing parameters then appears below.

The small info windows that appear when hovering the mouse pointer over the options are very helpful. The individual parameters appear in a selection field after clicking on the option; this approach helps to avoid misconfiguration. The *Sessions* and *Timers* tabs are for information purposes only, and the *User Units* tab contains processes started by the user (Figure 5).

SystemdGenie adopts the tab structure of Systemd-kcm and also offers the option of editing configuration files. You can modify the configuration of individual units using the context menu. Unlike Systemd-kcm, SystemdGenie calls help texts either from context menus or via the *Open Man Page* button, which make it easier for inexperienced users to enter the parameters correctly. You can use *Unit* from the menu bar to manage and edit the individual processes. The *Daemon* menu offers the option to completely reload or rerun the system.

Conclusion

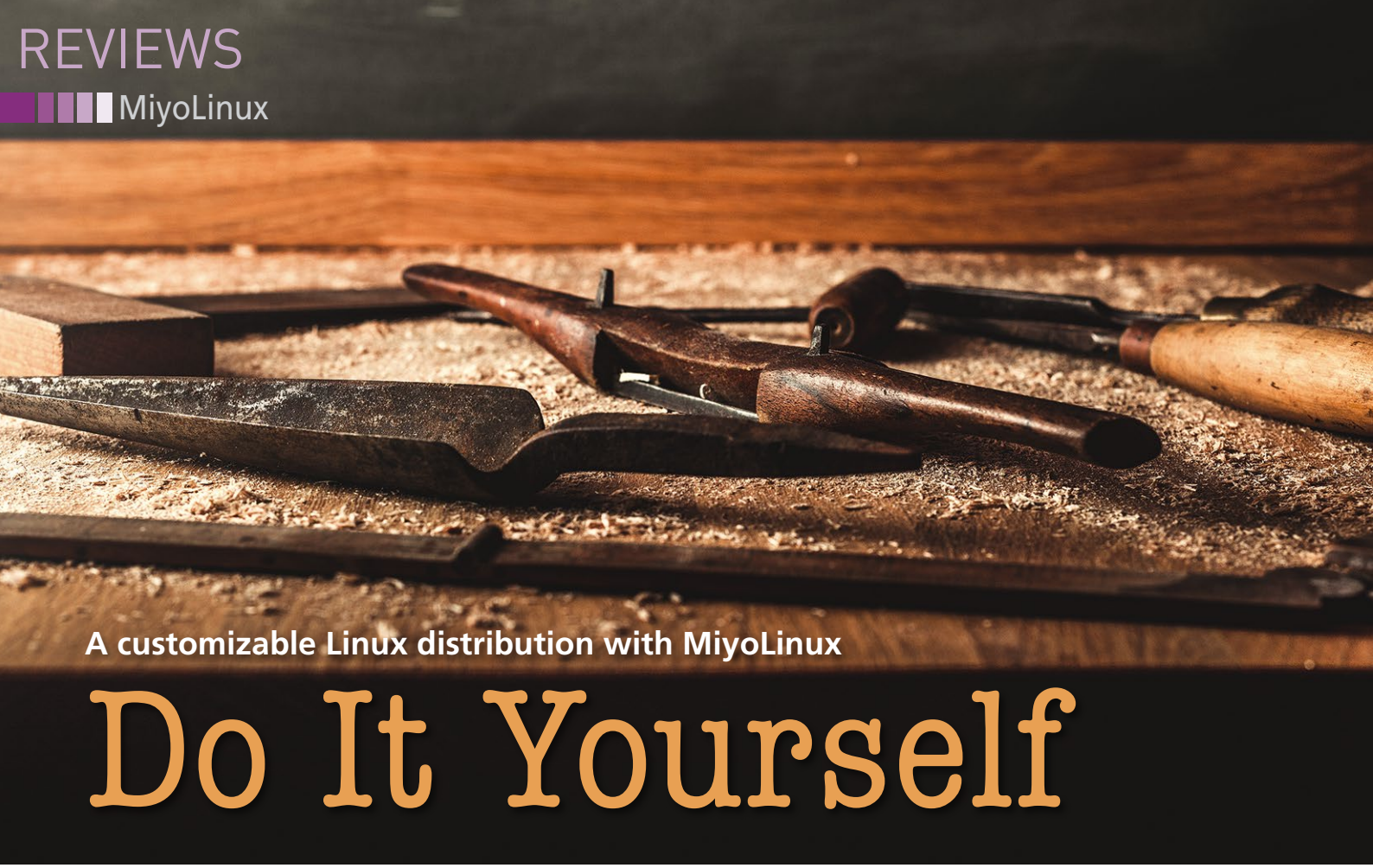
The three Systemd graphical interfaces developed for the KDE desktop are very similar visually, but they differ considerably in their functions. Kcmsystemd, Systemd-kcm, and SystemdGenie all provide a range of options that goes far beyond starting and stopping individual units. If you prefer Gnome instead of KDE, see the box entitled "Alternatives" for a summary of Systemd GUI options. ■■■

Alternatives

Systemd System Manager and Systemd Manager are graphical management tools for Gnome and other GTK3+-based desktop environments. However, their functionality is significantly less than that of the tools designed for KDE Plasma. Another option is Cockpit [6], a web-based administration tool that is not bound to any desktop environment. Cockpit, which left a favorable impression, is similar in functionality to the KDE tools and works with any popular web browser.

Info

[1] Systemd: *https://www.freedesktop.org/wiki/Software/systemd/*
[2] Kcmsystemd: *https://github.com/rthomsen/kcmsystemd*
[3] Systemd-KCM: *https://github.com/KDE/systemd-kcm*
[4] SystemdGenie: *https://github.com/KDE/systemdgenie*
[5] Manpages for the system processes: *https://www.freedesktop.org/software/systemd/man/index.html*
[6] Cockpit: *https://cockpit-project.org*



A customizable Linux distribution with MiyoLinux

Do It Yourself

The lean MiyoLinux is aimed at System V lovers who aren't afraid to customize. *By Erik Bärwaldt*

The Linux world is home to several lean distros that are intentionally designed to be lightweight and easy on system resources. Most of these systems avoid the bloat of KDE or Gnome, opting for a lighter and simpler desktop. They typically offer a scaled-down lineup of user applications, with the philosophy that, if you really want something else, you can always install it.

The primary use case for these lean distros is to run on old hardware, fulfilling Linux's role as the system that still runs on legacy computers. On the other hand, some users on conventional systems simply prefer a lean distro, because they don't like bloat. They find that a smaller footprint makes the system more responsive and easier to use.

Many lean systems are based on the universal Linux uber-distro known as Debian. The stable and well-tested Debian Linux is also the foundation for other Linux systems, including Ubuntu and Knoppix.

A huge controversy erupted in the Debian community over systemd, which became the default initialization daemon in Debian systems starting in 2015. Many of the Debian faithful objected to systemd, which they considered too complex and inconsistent with the Unix principle of simplicity. A group of developers eventually forked Debian to found the Devuan project [1] and develop a Debian-like distribution that retains the System V init and its surrounding components.

If you like Debian but prefer System V, you can use Devuan instead, but what if you want to use a lean distro based on Debian? Is there an answer for the Devuan/System V crowd?

The answer is yes. If you're looking for a scaled-down Linux based on Devuan and System V, you'll be happy discover MiyoLinux [2]. Miyo is an acronym for "Make It Your Own," which underscores the philosophy of starting with a bare-bones system and customizing it to fit your needs.

MiyoLinux uses the Openbox [3] window manager, which has been under development for more than 15 years and is considered extremely lightweight and frugal.

Although Devuan uses the DEB package format, it cannot offer the enormous software pool available through Debian. However, if you really want to access the full range of applications available through Debian, you probably don't want a system like MiyoLinux anyway. The whole point of MiyoLinux is to let you be selective.

Alive

The MiyoLinux ISO image for 32- or 64-bit hardware is about 600MB – not as small as truly tiny distros like Damn Small Linux, but significantly smaller than the latest generation of mainstream flavors (the latest Ubuntu weighs in at 1.8GB).

Burn the ISO image to an optical disc or USB memory stick and boot. (Remember: you'll need to use the option for burning a boot medium – you can't just copy the file.)

MiyoLinux opens a simple GRUB screen, which offers the option of copying the entire operating system into RAM and using it from there. This is particularly recommended for older computers or for starting from a slow optical data medium.

After a quick start, MiyoLinux welcomes you with a simple desktop (Figure 1). It

Photo by Alexander Andrews on Unsplash.com

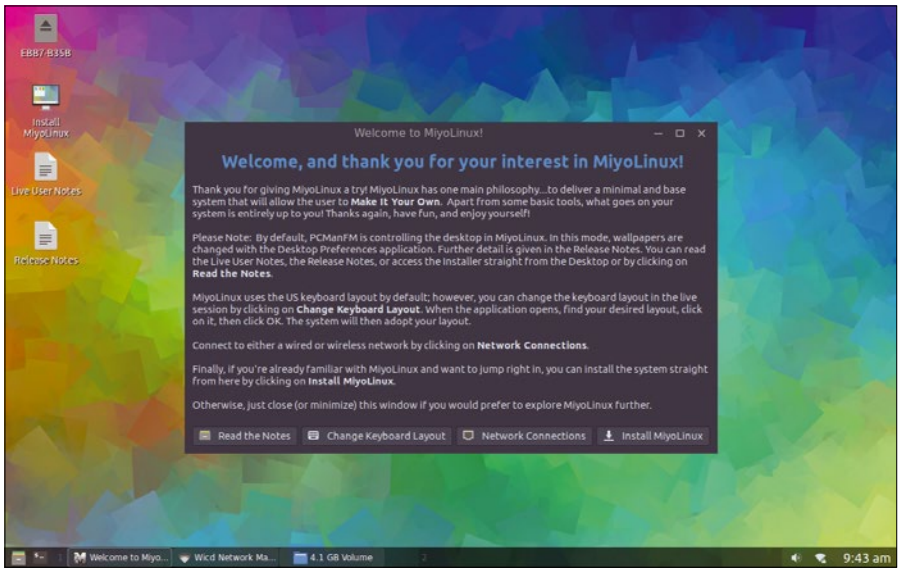


Figure 1: MiyoLinux starts in next to no time showing a simple desktop with a panel and welcome window.

only shows two icons for calling instruction files and a starter for the installation. A tint2 panel is at the bottom of the screen. A welcome window introduces you to the system.

Unlike most other distributions, MiyoLinux does not have a conventional menu button in the panel bar; instead, it has a starter for the PCManFM file manager. In the slim Devuan derivative, it not only acts as a file manager, but also – represented by the Applications menu – as a program launcher.

Various subfolders of the Applications menu let you access the few integrated applications. These tools include htop, BleachBit, and GParted, as well as the small text editor Leaflet. A web browser is missing, as are other ubiquitous heavyweights like LibreOffice, Thunderbird, VLC, and Gimp.

On initial startup, MiyoLinux uses *miyo* as the default username, password, and root password.

Setup

The welcome screen lets you customize the system using the *Change Keyboard Layout* and *Network Connections* buttons. If you are not connected to the network via a wired DSL connection, the Wicd wireless manager enables network access. One positive aspect is that MiyoLinux integrates proprietary firmware files for common wireless components; wireless Internet access thus works immediately (Figure 2).

Use the *Install MiyoLinux* starter on the desktop or the button of the same name in the welcome window to open the graphical installation wizard and install the operating system on the hard disk. The Refracta [4] installation wizard largely dispenses with graphics gimmicks and requires much more work from the user than, for example, Ubiquity or YaST, but it gets to the point very quickly. For partitioning, it uses either GParted or cfdisk.

In a small options dialog, you can also adjust certain configurations that differ from the default settings by setting or removing a checkmark (Figure 3). If you have already configured Internet access in live operation and entered some programs into the system using the integrated Synaptic package manager, the setup wizard takes over these applications. This saves you from having to reinstall the software later. The Synaptic package manager lists about 51,800 packages, or about 10 percent less than Debian.

Adaptable

Even after installation, the application folders only contain the tools you need

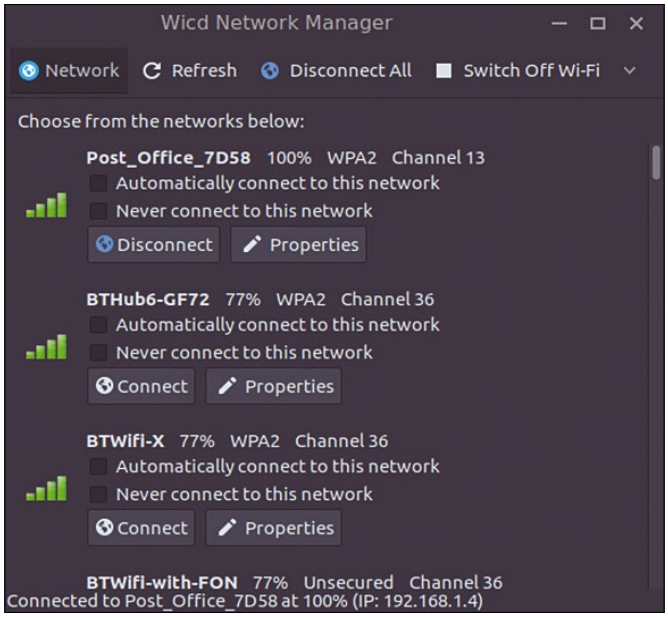


Figure 2: Thanks to Wicd, you can set up your wireless connection with just a few mouse clicks.

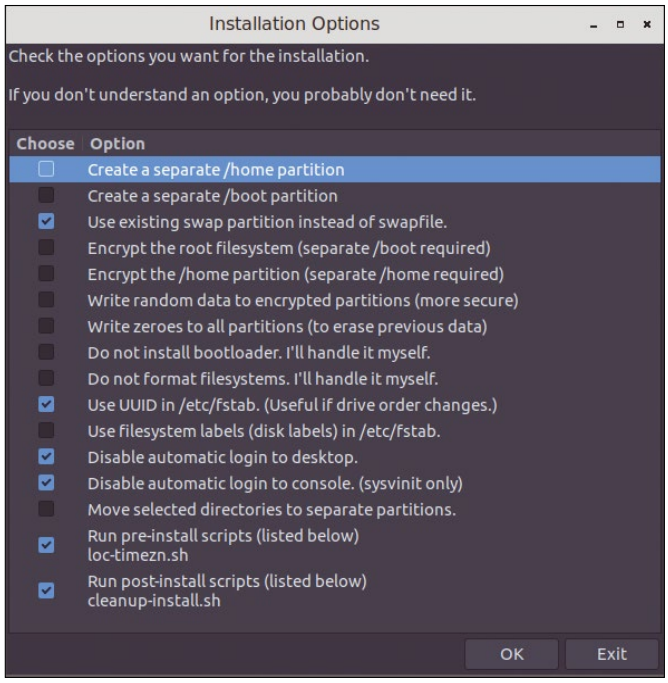


Figure 3: With Refracta, you can install MiyoLinux on your mass storage.

to assemble an individual system. MiyoLinux also borrows from other desktop environments: The Xfce power manager, for example, is a program for controlling the system's energy consumption that significantly increases the battery life of mobile computers, given suitable settings.

MiyoLinux also has the Gufw graphical tool for configuring the firewall from Ubuntu's software pool. You are not limited to Gtk applications: From Synaptic, applications can be installed from any desktop environment, including KDE Plasma; the package manager automatically resolves all the dependencies. MiyoLinux then adds appropriate starters to the `Applications` directory subfolders, which you need to call individually in PCManFM.

By default, MiyoLinux speaks English in all dialogs. If you would like to use another language, say German, you need to set up the packages *task-german* and *task-german-desktop* in Synaptic. Here, Synaptic pulls a total of 59 packages and occupies about 390MB of additional space on mass storage. In doing so, LibreOffice also gets onto the hard disk after all, with German help and menu navigation.

Even with other large standard programs, such as Firefox, you have to learn about the appropriate localization files that Synaptic lists separately. Localization, especially of the configuration tools, is not complete – which is unsurprising for a still quite recent project like MiyoLinux – and should improve in the course of time for users who prefer a different language setup.

Configuration

Openbox is one of the most configurable window managers. To avoid system administration being restricted to manually editing text files, the developers of MiyoLinux have integrated two configuration tools into the system, which combine several otherwise individual dialogs. Under *Applications* | *Preferences*, you will find settings for the tint2 panel bar, mouse, and keyboard, as well as appearance and energy management.

Further setting options are distributed among various subfolders, so sorting of the individual tools is somewhat confusing. It is much easier to configure the system by right-clicking on the desktop. Select the *MiyoLinux Control Center* option from the resulting context menu. The tool structures the individual dialogs in the tabs *Appearance*, *Configuration Files*, *System*, and *Miyo Accessories*.

The dialogs not only combine the tools already known from other subfolders in the Applications menu, but also introduce additional options. Under *System*, you will find the *Language Selector*, which allows a partial adaptation to other locales. The same tab also contains the *MiyoLinux Panel Switcher*, which replaces the tint2 panel with either the Xfce4 panel or a LXPanel at the push of a button. Neither use PCManFM as the application starter for scripts, but rather a conventional menu hierarchy (Figure 4).

The *Conky Controls* and *Compton Controls* dialogs in the *Miyo Accessories* tab can be used to enhance the desktop visually with various effects. The system comes with the Compton composite manager, which you can activate by clicking the *On* button. This turns on various desktop effects, such as shadows around each window or the darkening of inactive windows. You can switch the Conky system monitor on and off in the same way. It displays data such as time, date, RAM, CPU, and hard drive usage at the bottom of the desktop.

Because of the lean design, neither effect leads to noticeable performance losses. Together with an individual adjustment of the appearance, however, they turn MiyoLinux into a real eye-catcher.

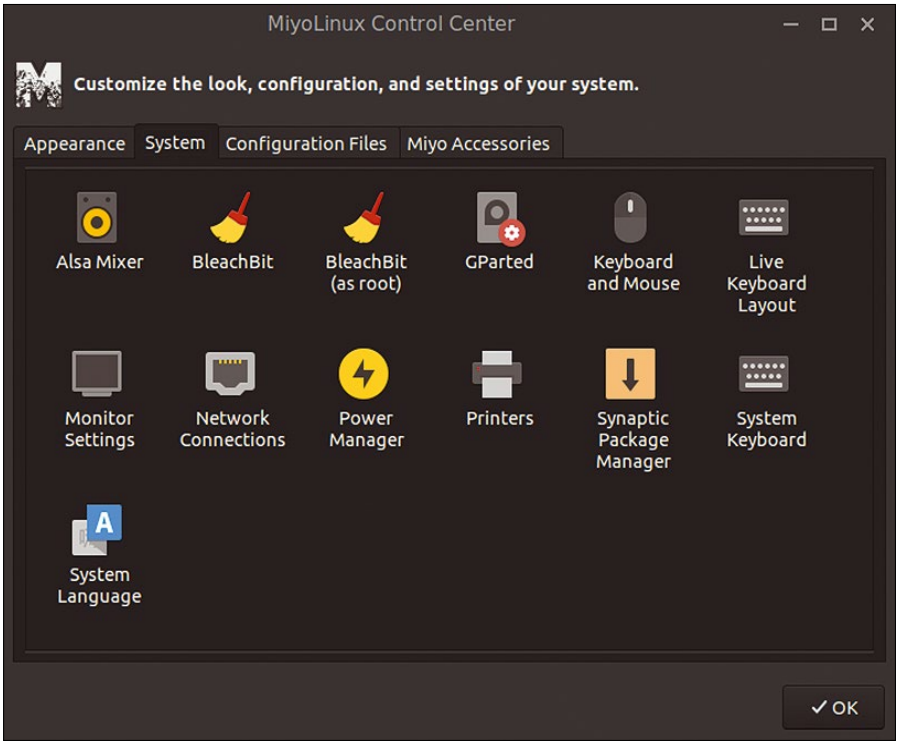


Figure 4: The MiyoLinux Control Center simplifies system configuration enormously.

Conclusions

The young MiyoLinux shows the potential of older window managers like Openbox. By combining numerous lightweight programs, MiyoLinux is every bit the equal of its bigger siblings. Although it is missing a few of the space guzzlers available through heavyweights like KDE, Gnome, and Xfce, MiyoLinux integrates many smaller tools and is a viable alternative for older hardware environments.

Devuan provides a solid foundation for MiyoLinux and provides a ready-made solution for distros to keep using System V. ■■■

Info

[1] Devuan: *https://devuan.org/*
[2] MiyoLinux: *https://sourceforge.net/projects/miyolinux/*
[3] Openbox: *http://openbox.org/wiki/Main_Page*
[4] Refracta: *https://www.ibiblio.org/refracta/*

Ten PDF Readers

FLEXIBLE TRANSPORT

PDF readers are easy to find in the Linux space. We tested 10 candidates, including a few fast and minimal alternatives, as well as others that offer convenience and additional features.

By Erik Bärwaldt

The Portable Document Format (PDF) [1], published by Adobe in 1993, quickly became the standard for exchanging electronic documents. Today, you can choose from a large selection of viewers for the Linux desktop.

In this article, I discuss 10 user-friendly and flexible PDF readers. In addition to standard functions, such as full-text search and bookmarks, the test team looked at the tools for marking and commenting. We also checked whether the readers can fill out forms and display file attachments. Table 1 summarizes the results.

Atril

Atril emerged from the Evince PDF viewer in 2011 and displays PDF, PostScript (PS), and EPUB files. Evince was the standard reader under Gnome 2.x at that time. Atril [2] is part of the Mate desktop [3] but is also available as a standalone application. The program usually appears in the *Applications* | *Office* menu. The interface is divided into three parts: In addition to the menu at the top and the toolbar at the bottom, a thumbnail previews the pages of the opened document on the left and the display area on the right.

Atril offers some additional functions. For example, it has a bookmark feature that is especially useful for large documents. The software also offers full-text searching and displays the results in green. In presentation mode, the reader

Table 1: PDF Readers Compared

	Atril	Evince	Foxit Reader	GSView	GV	MµPDF	Okular	qpdfview	Xpdf	Zathura
License	GPL	GPL	Proprietary	Proprietary	GPL	AGPL	GPL	GPL	GPL	zlib
Functions										
Text search	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes	Yes
Bookmarks	Yes	Yes	Yes	No	No	Yes	Yes	Yes	No	Yes
Forms	Yes	Yes	Yes	No	No	No	Yes	Yes	No	No
Comments	Yes	Yes	Yes	No	No	No	Yes	Yes	No	No
Markup	Yes	Yes	Yes	No	No	No	Yes	Conditional	No	No
Attachments	Yes	Yes	Yes	No	No	No	Yes	No	No	Conditional
Presentation	Yes	Yes	Yes	No	Yes	No	Yes	Yes	Conditional	Yes
Read aloud	No	No	No	No	No	No	Yes	No	No	No
Control										
Tabs	No	No	Yes	No	No	No	Yes	Yes	No	No
Thumbnails	Yes	Yes	Yes	Yes	No	No	Yes	Yes	No	No

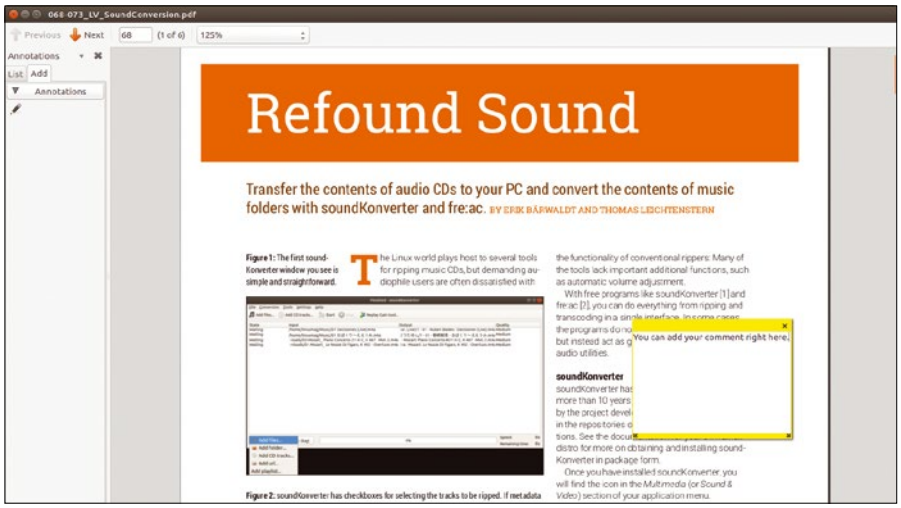


Figure 1: Atril offers a comment function enabled by the small pencil icon.

takes up the entire screen and is controlled with the keyboard. Alternatively, a Bluetooth controller can be used.

To insert comments and text markers into the document, you open the *Thumbnails* drop-down menu and select *Annotations*. In the left bar, Atril replaces the thumbnails with the *List* tab, which displays the annotations. New annotations can be added to the PDF from the *Add* tab with the pencil icon. The mouse pointer becomes crosshairs, with which you select the desired location to place the comment. Atril then displays a yellow balloon and opens an input box with the name of the user (Figure 1).

You can stop annotating by clicking on the pencil icon again and then closing the input window. The speech bubble remains visible at all times, and double-clicking it reopens the annotation. If you want to save the document and its comments, use the *Save copy* option from the File menu. After reopening, the left sidebar displays the comments along with the page number, date, time, and name of editor, if the *List* tab is enabled.

The viewer also fills out forms by clicking on the form fields. The selected field then turns into an input line where you can type your text.

Evince

The default viewer on current Gnome desktops is Evince [4]. Although its functionality is similar to Atril, the developers have visually adapted the reader to the conventions of Gnome 3.x. Therefore, you will no longer find a menubar, and the few buttons are integrated into the titlebar (Figure 2). Much like Atril, the left column displays thumbnails for quick navigation, comments, and attachments.

Comments and colored marks can be added to the document by clicking on the pencil icon. Evince also displays annotations, bookmarks, and markup from other programs without complaint. If you have added an attachment to a PDF, the reader can display the attachment and, depending on the file type, use a different Gnome program for viewing.

Evince can fill out and save PDF forms, although significant latency and display errors occurred during input. However, the completed forms were fine after saving and could be opened in other readers.

Foxit Reader

US-based Foxit Software has been developing and maintaining its Foxit Reader [5] since 2001 and offers it as freeware for free download. The manufacturer advertises it as a lightweight alternative to the sluggish Adobe Reader and provides a Linux edition (32- and 64-bit) for download. After unpacking the archive, you call a script that integrates the reader into the system with the help of a wizard and creates suitable menu entries.

Some older versions were noticeably impaired by ads and, above all, by security gaps, but the manufacturer has generally overhauled Foxit Reader and modernized its appearance. The program impresses with a tidy interface. A column on the left side offers quick access to documents, and thumbnails on the right show previously opened files.

The toolbar supports quick access to the navigation elements and searching. The three buttons, *View*, *Comment*, and *Connect*, open menus with frequently used functions. Foxit Reader is one of two test candidates that arrange new documents in tabs. If you miss the page preview thumbnails, you can enable the feature with *View* | *Navigation Panels* | *Pages*. The View menu also provides access to functions for rotating, enlarging, and reducing PDFs.

Foxit Reader shows its full potential when editing documents. Besides bookmarks, comments, and markers, the program also adds attachments to open files. Clicking on the paper clip in the left column displays and opens attachments. The program is not limited to the PDF format; it also stores audio and video. The reader does not open foreign formats; you save them with a right-click by choosing *Save attachment as*.

The test team was impressed with the comment function: Foxit Reader not only contains tools for commenting, but for crossing out, underlining, circling, and more. You can choose from several col-

Figure 2: Evince is the PDF viewer of current Gnome work environments and visually matches the desktop.

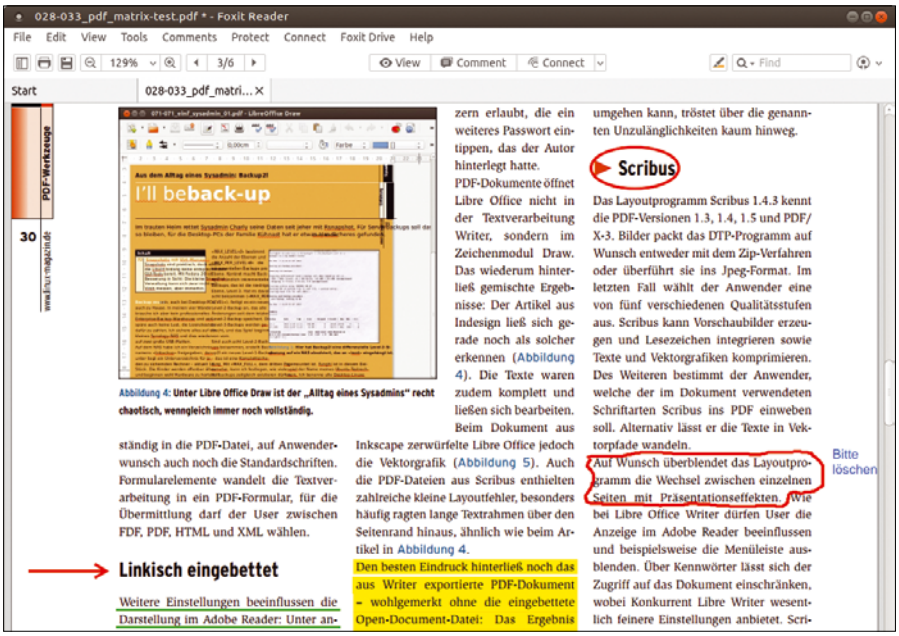


Figure 3: Foxit Reader offers many tools for marking parts of PDF documents or adding comments.

ors and geometric shapes (Figure 3). In forms, the program highlights fields that allow text entry in light blue and enters the letters directly into the boxes.

The *Connect* button provides functions for collaborative work, for which a free account and a commercial cloud connection to the manufacturer are required.

GSView

The non-free GSView [6], developed by US-based Ghostscript specialist Artifex Software, is primarily a user-friendly viewer for various formats. Besides PDF and PS documents, the program also opens JPEG, PNG, and EPUB files (version 2, without DRM). You will find 32- and 64-bit Linux installers on the vendor's website.

GSView comes with a modern interface (Figure 4) and uses the MµPDF viewer framework for rendering content. Strikingly, the GSView displays even large documents with several hundred pages and large-format photos extremely quickly. In the *Options* | *AntiAlias* menu, you will find different edge-smoothing levels, which make text with small font sizes easier to read.

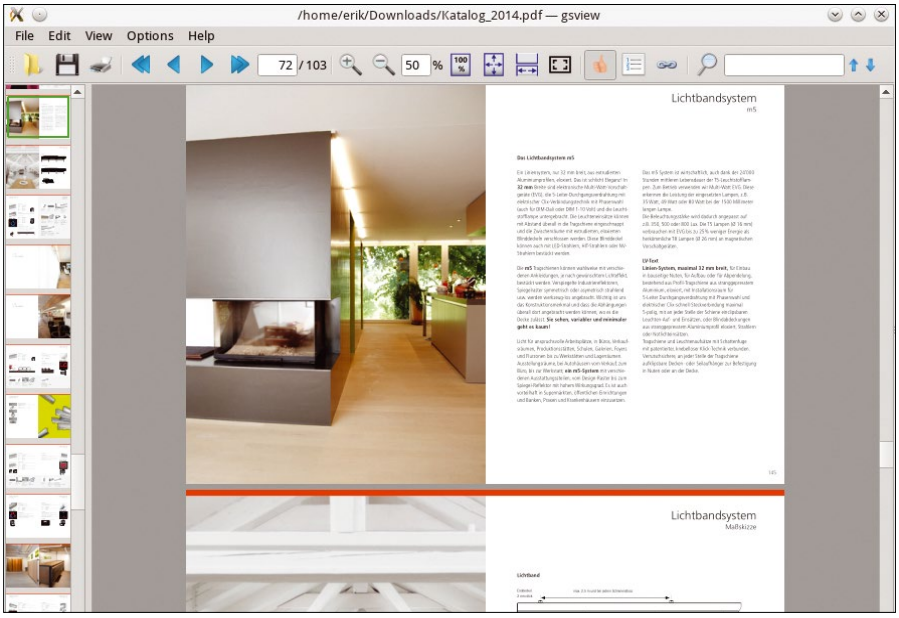


Figure 4: GSView impresses with its straightforward presentation.

A full-text search, navigation options to links, and individual page copies are possible. However, GSView lost points for the way it handles comments. The viewer displays existing annotations as symbols in the document but does not open them, and you cannot add your own comments. GSView cannot fill out forms, does not support bookmarks, and cannot handle attachments.

GV

The graphical Ghostscript front-end GNU GV [7] is developed by the Free Software Foundation. The reader is available for download on the project's FTP servers. After launch, the program catches the eye with its rustic appearance (Figure 5), and user guidance at first takes some getting used to, although it is quite easy to comprehend, thanks to self-explanatory symbols and buttons.

GV is a viewer only and has no special features except a presentation mode (which is hidden in the State menu). The program displays embedded comments with a colored symbol but cannot open them. In most cases, GV ignores graphical elements inserted by other PDF readers, such as text boxes, frames, and colored markers. The program cannot handle file attachments or forms.

The reader shows no weaknesses in terms of the display – GV even opens large documents quickly and without the delays typical of other viewers. Especially for large files with many embedded images, GV was one of the fastest candidates in the test.

Because the program can be controlled almost completely with the keyboard, GV has a speed advantage. A click on the magnifying glass displays marked areas in a separate window at different magnification levels.

MµPDF

The sixth candidate, MµPDF, also developed by Artifex Software, is a framework with a rendering engine for PDF and EPUB documents. The company has both commercial [8] and GNU Affero General Public License (AGPL) open source [9] versions. The developers focused on speed and low resource requirements. The lean program can be found in the repos of many distributions. You start the program in the terminal and enter the name of a PDF file. Called

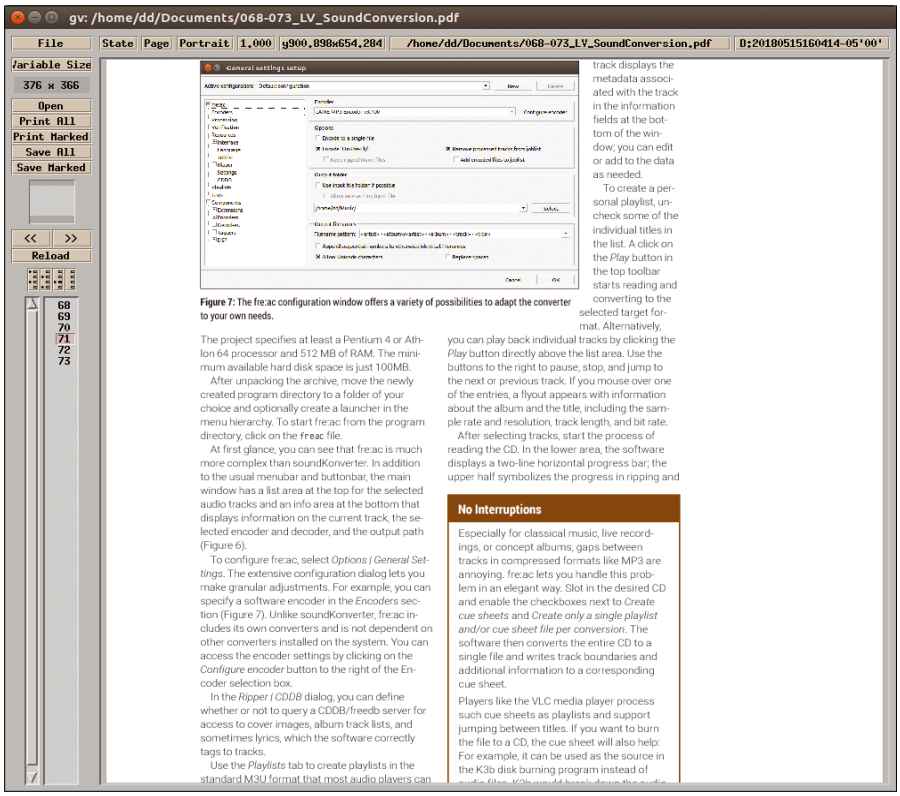


Figure 5: GNU GV looks outdated. However, the PDF reader by the Free Software Foundation impresses with its speed.

without parameters, the tool lists its options, including one for opening password-protected PDFs.

The minimalist interface appears without menus or toolbars (Figure 6). MµPDF is controlled with the keyboard, with keyboard shortcuts for rotating, enlarging, and reducing the display and for adding and jumping to bookmarks.

A comment function is missing. MµPDF displays comments made in other readers as balloons, but with no access to the content. Markup inserted with other readers was not always displayed by MµPDF in the test. It cannot fill in forms and ignores file attachments.

Okular

As an integral part of the KDE desktop environment, Okular [10] finds its way onto many Linux machines, but users of other work environments can also install the reader as a standalone package. In addition to PDF, it supports a

Figure 6: Focusing on the essentials: MµPDF is fast, but doesn't have menus or buttons.

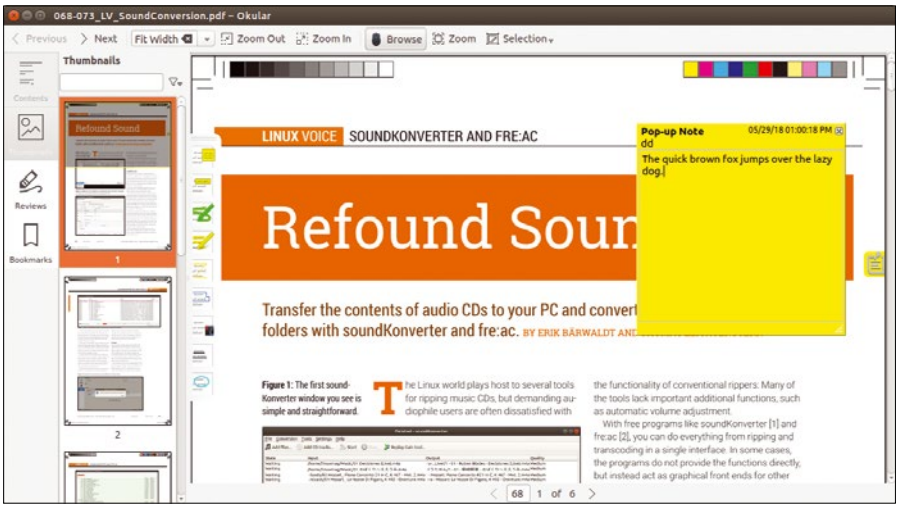


Figure 7: Comments are known as Reviews in Okular and can be accessed from the icon in the left sidebar. Apart from Foxit Reader, none of the test candidates offers better options in this category.

considerable number of other file formats, including EPUB, PS, ODF, and CHM.

Similar to Foxit Reader, the menu and toolbar are placed at the top; the left sidebar supports access to the table of contents, thumbnails, bookmarks, and comments (*Reviews*). Current versions of Okular can handle tabs, but this feature has to be switched on explicitly in a preferences dialog box.

If a document contains attachments, Okular displays a message with a link that opens a small dialog box with a list of attachments. The reader does not offer to display attached PDF files in the program: You first have to save the attachments to disk and then open them.

The comment function is reached by clicking *Reviews*, the Tools menu, or F6. Okular opens a vertical bar with various elements for marking or commenting. For example, the *Note* icon turns the mouse pointer into a cross, which you click in the document. The KDE program places a colored speech bubble there and opens an input window into which text is typed (Figure 7).

Only when the comments are saved with the PDF does Okular change the preview image column on the left: Instead of thumbnails, the reader now shows the *Reviews* category and the comments, page by page, in a table. Because the name of the commentator and the date and time are also visible, the function is suitable for collaborative work.

The test team liked the ability to create and manage bookmarks. A right-click on an entry opens a context menu for deleting and renaming entries. Because Okular numbers the bookmarks in ascending order by default, an individual designation – especially for large documents with many bookmarks – can be a real help.

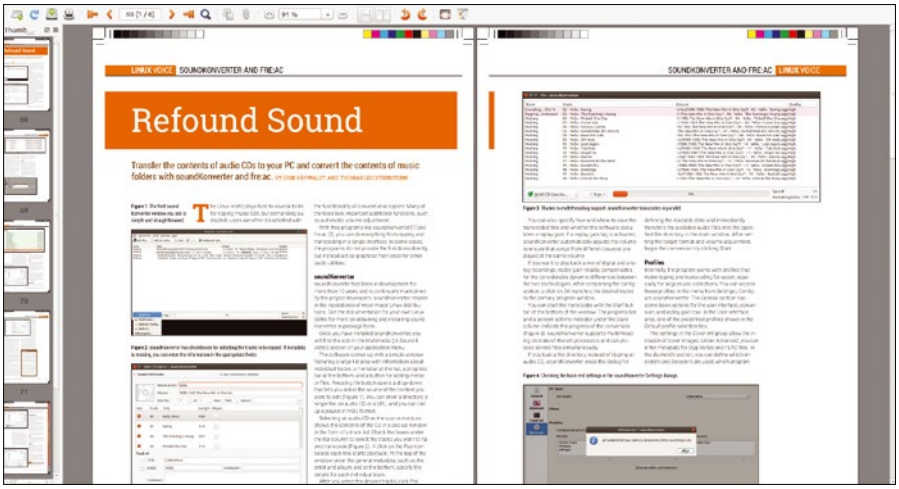


Figure 8: Qpdfview can display several pages side by side as you like.

Okular is the only candidate in the test to support speech output in conjunction with the text-to-speech function integrated in KDE. To make the reader barrier-free, the Speech Dispatcher service [11] must be installed, which controls voice output through the D-Bus interface (for KDE 4.x, the *kttsd* package; for KDE Plasma 5.x, the *jovie* package).

qpdfview

As the name "qpdfview" suggests, this PDF viewer is also based on the Qt graphics library. Qpdfview [12] is available in the package sources of most distributions.

The interface pleases with its simplicity and functionality: Besides the menu and toolbar, it offers an optional left sidebar; in the main window, the software displays documents in PDF, PostScript, and DjVu format. Qpdfview is one of the few Linux readers to handle tabs and offers an icon with a tab on the far left of the toolbar.

Another useful feature is the ability to design the look and feel of the program to suit yourself, which makes it possible to display further information from the View menu. You can choose from preview images, bookmarks, or even a table of contents, which you activate by ticking a box.

The toolbar is also flexible. If you open *View* | *Tool bars*, you can select or deselect specific areas. Qpdfview applies the changes without restarting. The View menu also includes options to enable continuous display of two or more pages in the display window (Figure 8). You can even rotate or invert documents.

Qpdfview does not support the ability to add attachments. However, the viewer has no problems with forms.

Xpdf

The Xpdf viewer [13] has been around for more than 20 years and is one of the dinosaurs among the test candidates. Besides the program for displaying documents, an installation of Xpdf comes with several other tools – for example, to convert to other formats or extract embedded images. Xpdf uses the Motif toolkit and therefore looks quite old-fashioned at first glance (Figure 9).

Contrary to current conventions, Xpdf has no menu; you control the reader with the buttons at the bottom of the window. Only a context menu is available, which opens when you right-click in the program window. This menu lets

Figure 9: Xpdf is visually the senior among the tested PDF viewers.

you open files, change the view (rotate, zoom in, zoom out), and enable continuous displaying of multipage documents.

Xpdf is quite spartan: You cannot fill out forms, and although you can see existing comments as a balloon icon, you cannot read them. The program also ignores underlined or highlighted text and attached files, nor can it add a file to a PDF.

Xpdf does not offer a presentation mode. However, you can switch to a full-screen view from the context menu.

Zathura

The last candidate in the test is also a minimalist. Zathura [14] can only be controlled with the keyboard. The shortcuts are similar to those you would use in Vim. If you compile the program yourself, you can choose between the PDF routines of Poppler and MµPDF. Zathura displays not only PDF documents but other formats, such as *zathura-cb* (Comic Book), *zathura-ps* (PS), or *zathura-djvu* (DjVu), for which additional packages have to be imported.

After startup, the lean viewer does not show a menu or toolbars. Only one status bar is visible at the bottom, leaving plenty of space for documents in the window (Figure 10). Scrolling is controlled by pressing *H*, *J*, *K*, and *L* (left, down, up, right, respectively); the + and - keys change the view. Entering *:q* terminates the program. If you are not familiar with Vim key combinations, you can read about them on the Zathura man page. If you want to set your own shortcuts, you can create a configuration file (`man zathurarc`).

Zathura does not allow you to fill out forms or mark or annotate anything in the document. To manage bookmarks, you can use the commands `:bmark` (define a bookmark), `:blist` (display all bookmarks), and `:bdelete` (delete one bookmark).

Well Prepared

None of the 10 test candidates showed weaknesses in their presentation. Most of the PDF viewers also support the latest PDF specification's special features. Those looking for a fast reader that doesn't suffer performance hits, even with large documents, should look at MµPDF or Zathura. GSView, Xpdf, and GNU GV are also recommended as lean viewers, but without many extras.

Atril, Evince, and qpdfview are solid all-rounders that also help reliably when filling out forms. Okular and Foxit Reader stand out from the competition: The KDE program is the first choice for those who need voice output. The proprietary Foxit Reader impresses above all with its appealing collaborative capabilities – as long as obtaining the required accounts in the manufacturer's cloud does not represent an obstacle. ■■■

Info

[1] PDF format: *https://en.wikipedia.org/wiki/Portable_Document_Format*
[2] Atril: *https://github.com/mate-desktop/atril*
[3] Mate applications: *https://mate-desktop.org*
[4] Evince: *https://wiki.gnome.org/Apps/Evince*
[5] Foxit Reader: *https://www.foxitsoftware.com/pdf-reader/*
[6] GSView: *https://www.gsview.com*
[7] GNU GV: *https://www.gnu.org/software/gv*
[8] Artifex MµPDF commercial license: *https://www.artifex.com/licensing/*
[9] Open source MµPDF: *http://www.mupdf.com*
[10] Okular: *https://okular.kde.org*
[11] Speech Dispatcher: *https://freebsoft.org/speechd*
[12] qpdfview: *https://launchpad.net/qpdfview*
[13] Xpdf: *https://www.xpdfreader.com*
[14] Zathura: *https://pwmt.org/projects/zathura/*

Figure 10: You control Zathura with the keyboard. The program handles tab completion and displays status messages at the bottom of the window.

Create system snapshots with CYA

System Saver

The CYA script helps you back up system files, which you can then restore quickly in case of an emergency. *By Ferdinand Thommes*

It is always surprising how many users skip regular backups. This negligence can hit you hard when least expected. Backup software generally backs up directories, partitions, or an entire installation.

Some backup tools are used to back up and restore the operating system itself, not the user data. CYA, for instance, is an 80KB Bash script based on rsync that lets the user keep a copy of the system close at hand. CYA stands for "Cover Your Ass(ets)" – I think we all get the gist. CYA does not try to overload you with a graphical interface; instead, it lets you start the backup with a single command – or even automatically.

Versatile

According to the developer, the CYA script, which was developed by Cyberweb Solutions, can create snapshots of any Linux or BSD distribution – and theoretically, even of any system that offers Bash and rsync. CYA understands how to handle most Linux filesystems, including ZFS and Btrfs. The software is licensed under a BSD license, so anyone can view and modify the code for their own use. You can run CYA automatically with systemd or cron or integrate it into scripts.

The way CYA works is similar to restore points in Microsoft's System Restore, which also only backs up the operating system itself without user data. You can reset the system to an earlier status using a restore point. CYA offers the same features. The home directory remains untouched – at least so far. In the latest version, the software saves the user's data separately with the `cya mydata` function.

By default, the program automatically or manually creates three snapshots; then, it overwrites the oldest one. You can also tell CYA to create snapshots that it does not overwrite or that do not even appear in the list of backups.

The software backs up without compression or a proprietary format by default, which means that you can easily restore not only the entire installation, but also individual files or directories if required. You do not actually need the `cya restore` function; in principle, a file manager or a tool for the terminal is sufficient.

What Installation?

Because the software is a script, you don't need to install CYA in the classic way. Just download it as a ZIP archive [1], which you then unpack, then copy the `cya` file to the `bin` directory under your home directory or the `usr/local/bin` directory. If you want to put the `cya` script somewhere else, add the path to the `$PATH` variable. Be sure to set the file permissions to make the file executable.

The script is now ready for use. The only other requirement is that the system is configured to use `sudo` [2]. Once you have done this, make sure you are in the home directory and execute the `cya` command (Figure 1). If nothing happens, log out and back on again – but this should not be necessary for most distributions.

As you can see, the program creates a new user named *cya*, as well as some directories and files. Snapshots will end up in `points/` in the future. The `cya.conf` file is still empty – you can store your own configuration options here. The `LAST_RUN` file tells you when the software last ran.

The `cya help` call reveals all the options and parameters you can use to control CYA (see Table 1). The default settings are fine for many use cases; however, you might want to familiarize yourself with the available options to have the right commands at hand when in doubt.

Configuration

Type `cya configfile` to configure. Examples of modifying the configuration are shown here. If you need a rotation setup

```
ft@cya-test:~/bin$ ./cya
☀ Cover Your Ass(ets) v2.2 ☀

tee: /home/cya/versionts: No such file or directory
Checking sudo permissions...
We need to create /home/cya ... done
Checking sudo permissions...
We need to create /home/cya/points ... done
Checking sudo permissions...
We need to create /home/cya/cya.conf ...done
Checking sudo permissions...
We need to create /home/cya/LAST_RUN ...done
ACTION ☒ Enter A Valid Command

Please enter a valid command.  To list commands enter:

cya help

Backup Name                                    Date & Time
```

Figure 1: During the first run, CYA creates the new user named *cya* and its infrastructure.

with more than three slots, you can change `MAX_SAVES=""` and write this statement and all the others to the empty `cya.conf` file in `/home/cya/`. You can then add or exclude files and directories until the result is what you want. The sequence is irrelevant.

Once you have completed the configuration, start the first backup. Simply enter `cya save` in the terminal (Figure 2). The script then saves the system except for `/home/`. CYA stores the snapshot in the `home` directory of the *cya* user created by the system under the `points/` directory.

If the home directory is on a separate partition and you use the automatic method, you must make sure there is enough free space for at least four backups. This number results from the fact that the program keeps three snapshots each for the automatic method and then rotates.

Enumerated

If you call cya list, you will see the first snapshot with its date and time under point 1. Other snapshots that you created automatically or by running `cya save` are numbered by the tool and rotated as needed (Figure 3).

The `cya keep` command lets you create manual snapshots that are not subject to rotation, which is a good way to take a snapshot immediately after installing the operating system under a name such as `basisinstallation`. Even if you are importing a large update or an extensive package, creating a snapshot up front gives you additional assurance.

If something goes wrong, you can roll back the system to a previous state. Although you will want to keep `basisinstallation` and even archive it with `cya archive`, you can dispose of snapshots created before updates with the `cya remove` command as soon as you no longer need them.

Table 1: CYA Commands

Command	Results
Create Snapshots	
`save`	Create snapshot (rotating)
`keep`	Create manual snapshot (no rotation)
`mydata`	Back up your home directory
`archive`	Manually archive snapshot
Recovery	
`restore`	Restore data
Other	
`configfile`	Help for manual configuration
`directories`	List directories to be backed up
`list`	List snapshots you have created
`remove`	Remove manual snapshots
`script`	Linux mount script
`uses`	Use cases for script integration
`version`	Update check

Manual and Scripted

After entering the `cya keep` command, the program prompts for a unique snapshot name, which must not contain spaces. Alternatively, enter a name with the command; if run in scripts, use:

```
cya keep name pre_upgrade
```

The rest of the process is the same as the automatic backups, except that the snapshot does not reside in a numbered directory. Instead, it ends up in a folder with the assigned name. So, if you stick to the above backup as an example right after installing the operating system, then the data could be found in `/home/cya/points/basicinstallation`.

If you want to use the software in a script that automatically creates a snapshot every month and overwrites the previous one, the command is:

```
cya keep name monthly overwrite
```

Here, the tool takes an incremental approach and only considers changes since the last snapshot.

```
ft@cya-test:~$ cya save
☀ Cover Your Ass(ets) v2.2 ☀

ACTION ☒ Standard Backup

Checking sudo permissions...
We need to create /home/cya/points/1 ... done
Backing up /bin/ ...  complete
Backing up /boot/ ...  complete
Backing up /etc/ ...  complete
Backing up /var/ ...  complete
Backing up /lib/ ...  complete
Backing up /lib64/ ...  complete
Backing up /opt/ ...  complete
Backing up /root/ ...  complete
Backing up /sbin/ ...  complete
Backing up /usr/ ...  complete
Backing up /initrd.img ...  complete
Backing up /initrd.img.old ...  complete
Backing up /vmlinuz ...  complete
Backing up /vmlinuz.old ...  complete
Write out date file ... complete
Update rotation file ... complete

ft@cya-test:~$
```

Figure 2: During the first snapshot, the software implements the configuration you have created. The backed up data is stored in the `points/` directory.

```
ft@cya-test:~$ cya list
✳ Cover Your Ass(ets) v2.2 ✳

ACTION ☒ List Backups

Backup Name                                    Date & Time
------------------------------------------------------------------------------
1                                              Fr 30 Mar 2018 10:38:13 CEST
------------------------------------------------------------------------------
2  <-- LAST RUN                                Fr 30 Mar 2018 10:48:21 CEST
------------------------------------------------------------------------------
after_upgrade                                  Fr 30 Mar 2018 19:14:30 CEST
------------------------------------------------------------------------------
before_upgrade                                 Fr 30 Mar 2018 11:07:18 CEST
------------------------------------------------------------------------------

ft@cya-test:~$
```

Figure 3: The `list` option displays two standard snapshots and two manually saved snapshots.

If you do not want to overwrite the backups, use

```
cya keep name monthly-DATE
```

instead; this tells the script to append the respective date to the name.

Recovery

Any backup or snapshot system is only as good as its data recovery function. It is therefore worth testing the restore before an emergency occurs. First, use the `cya script` command to create a recovery script.

Specify which directories the system must mount when restoring. You can make your own entries here if you want to include additional directories (Figure 4). You will want to create the script immediately after configuration and save it separately.

If data to be restored resides outside the default directories, you can manually enter the under `/home/cya/`. For example, if `/var/` is on an external hard disk, you need to add the mountpoint here. Then, copy the finished `recovery.sh` script to a USB stick, which you later mount in a Live system and use to restore the backups.

As already mentioned, you should test the process before an emergency occurs so that you are familiar with the procedure. If the hardware, partition layout, or CYA version changes, you will want to run `cya script` again to customize the script.

The recovery process requires a Live system. You start this from a USB stick, an optical medium such as a CD, or a Netboot image. If possible, use a version of the operating system that matches the installation to be restored.

In a field test, I deleted the `boot/` directory on a machine with:

```
sudo rm -rf /boot
```

This meant the computer was no longer bootable. CYA had to restore this directory to have a bootable system after the restore.

Recovery from a Live System

From a running Live session, first mount the USB stick with the recovery script, if this does not happen automatically, and switch to the console. Alternatively, you could mount the partition manually and specify the absolute path to the script in `home/cya/`. Then, start the script with the `./recovery.sh` command and follow the instructions (Figure 5).

First, select which snapshot you want to restore; then, you can take a new snapshot of the installed system. In the test case here, this didn't make sense because the system was no longer bootable, but it can help in the later diagnosis of a broken system.

```
ft@cya-test:~$ cat /home/cya/recovery.sh
#!/bin/bash

# Cover Your Ass(ets) Mounting Script v1.0
# This script attempts to ready a Linux system for cya by mounting the filesystem

# Create mount point
sudo mkdir -p /mnt/cya

# Mount root filesystem
sudo mount /dev/sda2 /mnt/cya/

# Add additional mount calls below this line - mount into /mnt/cya/CUSTOM_DIRECTORY
# Dont forget to add create directory before mounting if it doesnt already exist!
# Ex: sudo mkdir /mnt/cya/backups
# Ex: sudo mount /dev/sda4 /mnt/cya/backups/

echo "Run the cya recovery program by issuing the following command:"
echo "sudo /mnt/cya/home/cya/cya restore"

# Run restore
sudo /mnt/cya/home/cya/cya restore
ft@cya-test:~$
```

Figure 4: The `recovery` script includes commands to mount the necessary partitions to ensure easy recovery. You can adapt it to your individual needs.

```
iducer@siduction:/mnt$ cd /media/siducer/
iducer@siduction:/media/siducer$ ls
7b68cdf-245b-419c-827a-73f15a3762bd
iducer@siduction:/media/siducer$ cd e7b68cdf-245b-419c-827a-73f15a3762bd/
iducer@siduction:/media/siducer/e7b68cdf-245b-419c-827a-73f15a3762bd$ ./recovery.sh
un the cya recovery program by issuing the following command:
udo /mnt/cya/home/cya/cya restore
✳ Cover Your Ass(ets) v2.2 ✳

CTION ☒ Restore

((WARNING))) You are about to begin the restore process which WILL alter the system's operating files!!!
It is best practice to restore a system from a live boot environment using a CD, USB, or network image.

re you sure you want to continue this process? [y/N]
```

Figure 5: Starting the `recovery` script in the Live environment initiates the recovery. If you start it within the normal installation, it will refuse to work.

```
======================================================================
FILE CHANGES ABOUT TO OCCUR! VERIFY AND CONFIRM:
======================================================================

Restoring Backup Profile: before_upgrade
Backing Up Current State: Yes
Delete Files Not Found In Backup: No
Ask To Restore Directories Individually: No

* DATA CHANGES IMMINENT*
To start process type the word restore in all lower case, otherwise hit any other key to abort ...
```

Figure 6: CYA warns about the point from the process cannot be reversed. It then starts the recovery of the lost files.

```
First pick the name of a backup to restore. Press any key to continue...

Backup Name                                    Date & Time
------------------------------------------------------------------------------
1                                              Fr 30 Mar 2018 10:38:13 CEST
------------------------------------------------------------------------------
2  <-- LAST RUN                                Fr 30 Mar 2018 10:48:21 CEST
------------------------------------------------------------------------------
after_upgrade                                  Fr 30 Mar 2018 19:14:30 CEST
------------------------------------------------------------------------------
before_upgrade                                 Fr 30 Mar 2018 11:07:18 CEST
------------------------------------------------------------------------------

Enter backup name to restore: before_upgrade

Do you want to take a backup of current state before restoring?

Enter your choice: [y/N]
```

Figure 7: First you determine which backup you want to import. Then specify whether you also want to create a current backup of the installed system.

```
ft@cya-test:~$ cya list
* Cover Your Ass(ets) v2.2 *
ACTION ☒ List Backups

Backup Name                                    Date & Time
------------------------------------------------------------------------------
1                                              Fr 30 Mar 2018 10:38:13 CEST
------------------------------------------------------------------------------
2  <-- LAST RUN                                Fr 30 Mar 2018 10:48:21 CEST
------------------------------------------------------------------------------
after_upgrade                                  Fr 30 Mar 2018 19:14:30 CEST
------------------------------------------------------------------------------
before_upgrade                                 Fr 30 Mar 2018 11:07:18 CEST
------------------------------------------------------------------------------
RESTORE-1522487974                             Sa 31 Mar 2018 11:31:35 CEST
------------------------------------------------------------------------------
```

Figure 8: The `cya list` command shows the new backup created during the restore after restarting the restored system.

Next, decide whether the script removes files that are not in the backup from the installed system. The answer to the last question determines whether the recovery runs as a single pass or whether you want to confirm each directory of the root tree individually. In my example, the latter made sense because I knew that the system only lacked the boot directory. If you don't know what's broken, *no* would be the right answer.

Now, a final warning appears notifying you that the software will overwrite all data from this point onward. CYA shows the selected options and waits for you to enter the word `restore` to start the process (Figure 6). The script then processes the selected options (Figure 7).

Within minutes to half an hour, depending on the options selected and the hardware used, the process is complete and the software prompts you to restart the installed system (Figure 8).

Conclusions

There are many ways to create restore points for a system restore. CYA may seem complex at first glance, but that has a positive long-term effect. The script offers many possibilities for customization and is suitable for desktop use, as well as for use via SSH on a server (Figure 9).

As with any backup program, you should run through the process, including the restore, at least once. The project has documented the entire process in an exemplary manner on the website and in around 20 YouTube videos on the individual topics [3].

Once you have configured CYA and provided it with a cronjob or systemd timer, you do not need to do anything until an emergency occurs. If necessary, you can take manual snapshots in just a few seconds. CYA did not cause any problems in the test and will replace Clonezilla on my system, which I previously used for this purpose. ■■■

Info

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

[2] "sudo: *https://wiki.debian.org/sudo*

[3] Videos: *https://www.youtube.com/watch?v=fdH3Um4XNUs&list=PL7D-dW0jxJRrEQfKXeulzTWmPyomIp4k4Y*

```
ft@cya-test:~$ cya uses
* Cover Your Ass(ets) v2.2 *

ACTION ☒ Uses

This script has many uses including:

* Software Testing - clean up changes caused by testing software
* Configuration Changes - rollback system configuration changes
* Bad Upgrades - undo botched upgrades
* Foil Intruders - erase unauthorized system files and changes


ft@cya-test:~$
```

Figure 9: CYA supports many scenarios, some of which are listed in the `cya uses` command.

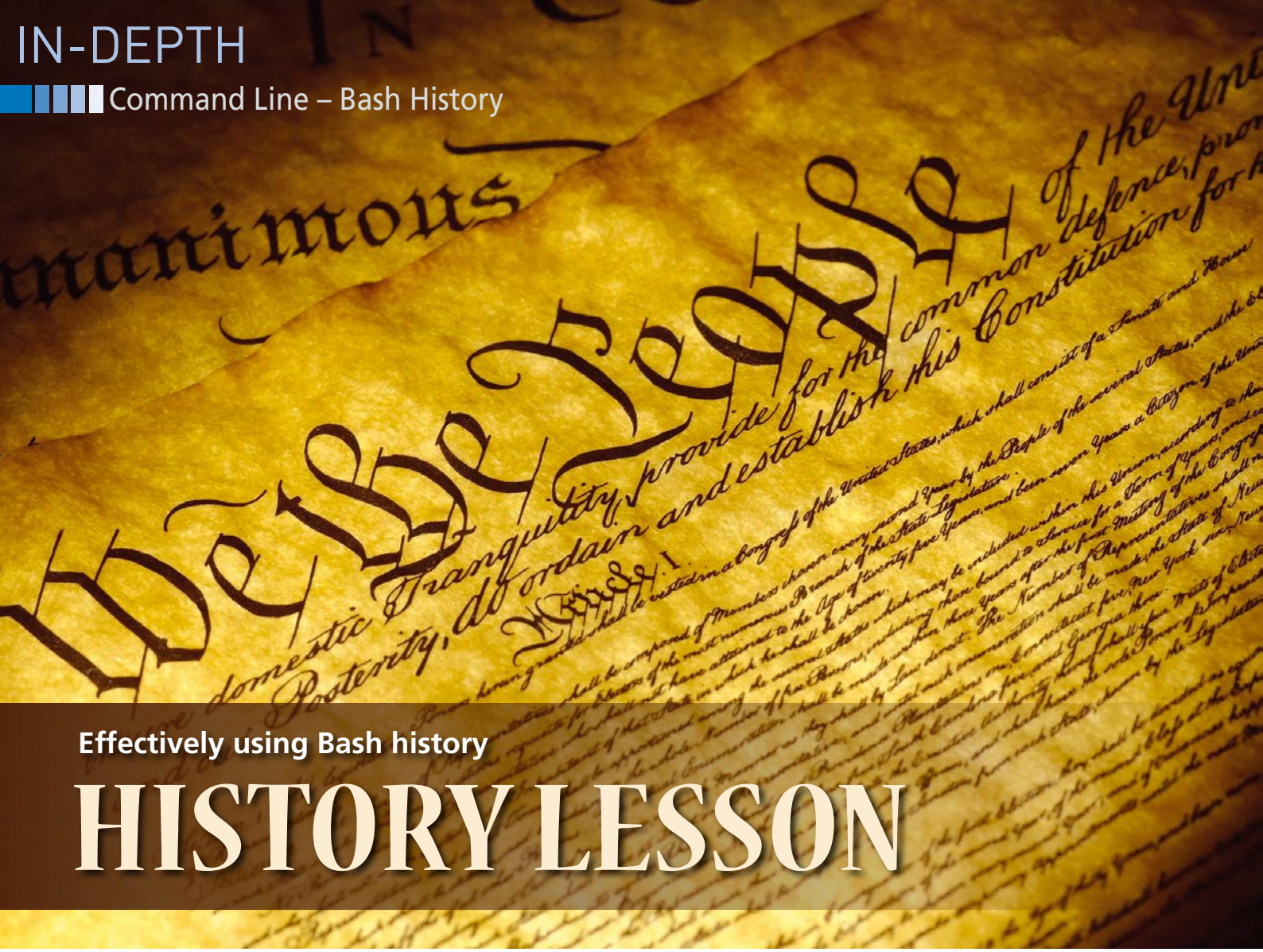
Complete Your Open Source Library with Archive DVDs!

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



Effectively using Bash history

HISTORY LESSON

You can do more with the Bash history command than just using the arrow keys. We show you how to use this command-line tool more efficiently. *By Bruce Byfield*

If you use the command line regularly, you probably use the Bash history file. Usually located in `~/.bash_history`, the history file stores hundreds of previous shell commands, enabling you to reuse them without typing them. If you are like most users, your use of Bash history is largely confined to using the arrow keys to scroll back one or two commands. Yet the `history` command itself offers dozens of ways to search and reuse the history more efficiently: the regular command options, history expansions, environment variables, and a couple of alternative search commands as well.

The limited use of Bash history may now be common partly because many users today have little command-line experience. Even more importantly, the information available in the `history` command's man and info pages is verbose and only partial [1]. At best, users pick up isolated tips from how-to articles, which give them little sense of the command as a whole.

Still another reason for not making full use of the `history` command is the uncertainty about all it can do. For one thing, the command is controlled by several automatically installed environmental variables that users may never see unless they open `~/.bashrc`. For another thing, by default, the file collects every entry at the command line, whether the entry is properly formed or not. Additionally, entries from every terminal opened after the first terminal are not collected, but not written, which means the history of all except the original terminal are lost when the window is closed. Such quirks are logical, but arcane enough that it quickly makes the `history` command a mystery to the average user. What follows is an attempt to systematically explain the different ways you can use history more exactly.

The Basic Command and Its Options

Issuing the basic `history` command displays all the commands in the history file one per line. Probably, you will want to pipe it through `less`, especially if your interest is in the most recent commands, which are listed at the top of the file. Each line begins with the line number in the history file. The displayed information helps you locate commands to

Author

Bruce Byfield is a computer journalist and a freelance writer and editor specializing in free and open source software. In addition to his writing projects, he also teaches live and e-learning courses. In his spare time, Bruce writes about Northwest Coast art. You can read more of his work at *http://brucebyfield.wordpress.com*

```
945  cd /mnt
946  ls
947  mount /dev/sdc /mnt
948  su -
949  man unoconv
950  man unocov
951  man unoconv
952  apt-cache search unoconv
953  su -
954  man unoconv
955  ls Pdf
956  cd ./Downloads
957  ls pdf
958  cd ./Download
959  ls pdf
960  ls
961  cd /home/bb
962  ls
963  cd /projects/creative
964  cd /projects/
965  pwd
966  ls ./projects
967  cd ./creative
```

Figure 1: The bare `history` command lists the contents of the history file.

```
bb@nanday:~$ history 4
 1942  history 3
 1943  man history
 1944  history |less
 1945  history 4
```

Figure 2: You can specify the number of history items to list after the bare command. Shown here are two listings, one for showing 3 entries and another for 4.

reuse, so, essentially, the unmodified command is a search feature (Figure 1).

Very quickly, you are likely to want to add an option. These options, you will notice, have changed so little since the days when Unix ruled that they are prefaced by a single hyphen and do not have the corresponding GNU-invented options that start with two hyphens.

Most of the options for the `history` command are housekeeping options. For example, you can use `-dNUMBER` to delete a specific entry for which you have no use for one reason or another. If your history is long and you see no use for most of the entries, you can even clear the entire history file with `-c`. Perhaps most importantly, if you want to reuse recent entries before the next time you log in, you can add them immediately with the `-a` option. To avoid duplications, you could use `-n` instead and only add commands from the current session that are not already in the history file. None of these options give any feedback when errors occur, so be certain you know what you are doing when applying them.

History Expansions

In addition to the options found in most commands, `history` also has what the man page describes as history expressions. History expressions can be thought of as the regular expressions of the `history` command that draw on the history file, but not always with the need to type `history`. They fall into three categories: event designators, word designators, and modifiers.

Event designators pinpoint an entry in the history file and begin with an exclamation mark (!). For example, if you have identified the line of the command you want to repeat, you can add `!NUMBER` to the basic command, while `!-NUMBER` selects a number of lines before the current one (Figure 2). Two exclamations points (!!) will repeat the previous command. More elaborately, `!COMMAND` will let you to reuse the last time the specifically named command was used. If you know the number for a specific command, you can reissue it by adding `!NUMBER` – all of which is usually considerably faster than scrolling with the arrow keys.

At other times, you might notice a command that was incorrectly entered, such as `sl` for `ls`. If it was the last command, you can save yourself some typing and add `COMMAND !*` to enter the proper spelling while using the same options and file paths (Figure 3). In other cases, an event designator works with a string of characters, so `!STRING` locates the most recent entry with the specified string. If, in the middle of typing, you realize that you have recently used the same command that you are inputting, `!#` will complete it with the last matching entry.

Word designators filter by character strings, with the position of words counted from the beginning of the line, the first word being `0` and the last word a number or `$`. Multiple words are separated by single spaces.

After a word designator, various modifiers can be added, each prefaced by a colon (:). An `h` will reproduce only a file name extension, and a `t` will remove it. The modifier `s/OLD/NEW` will allow you to edit an entry when you renew it. If you have made a substitution in the previous entry, `&` will repeat it. Such expansions can get confusing, so you may want to add `p`, so that the modified command is displayed, but not carried out.

These are only some of the most common history expansions available. For a complete list, consult the man or info page [1], both of which focus on history expansions to the exclusion of the rest of the `history` command's complexity. However, even the examples given are enough to show how much more there is to the command than scrolling with the arrow keys.

```
bb@nanday:~$ sl /home/bb/Downloads
bash: sl: command not found
bb@nanday:~$ ls !*
ls /home/bb/Downloads
2018_yvraf_release_final.pdf
```

Figure 3: Some event designators work without requiring the `history` command.

```
ob@nanday:~$ history |grep apt-get
 1042  apt-get remove tilix
 1070  apt-get remove tilix
 1498  apt-get install goto
 1966  history |grep apt-get
```

Figure 4: Sometimes, an ordinary `grep` on the history file can be useful.

```
(reverse-i-search)`.ogg': vorbisgain *.ogg
```

Figure 5: A reverse search is a type of autocomplete.

Environment Variables

Some aspects of the `history` command and file are set either temporarily or permanently with environmental variables. To set the variables for the current session, use the `export` command before using `history`. To set the variables permanently, use the `env` command. Permanently stored variables are saved to a user's `.bashrc` file, and variables for the entire system to `/etc/bashrc`. The history variables for the session can be overwritten using the command:

```
unset export VARIABLE=setting
```

Permanent variables can be overwritten by entering a new value.

Five history variables are supported:

- `HISTCONTROL=SETTING` can be completed in several ways, all of which are designed to prevent duplicates. The choice of `ignboth` prevents the history file from writing a command that is already present in the file. By contrast, `ignorespace` will not write a command to the file that starts with a blank space. If duplicates already exist, you can remove them with

```
export HISTCONTROL=erasedups
```

- `HISTCONTROL` may be supplemented by `HISTLSWORDS`, a colon-separated list of character strings that indicate which commands the history file should not store. For instance, since package installation is often a one-time operation, some users might ban the storage of any file that contains `apt-get`.
- `HISTIMEFORMAT` displays the time a command in the history was issued, giving you another means to locate it. The variable uses the standard format for date and time: `%d` for the day, `%m` for the month, `%yyyy` for the year, and so on.
- `HISTSIZE` is the number of entries stored at one time, 500 is usual in most distributions. When the number of entries is exceeded, the oldest entries are overwritten. Since memory is cheap these days, many history users set a much higher number for convenience. At the opposite end of the scale, you can turn off the history file by setting this variable to `0`.
- `HISTFILE` designates a file other than `.bash_history` for storing entries. `HISTFILE` is mostly useful for a common history file between different users or systems, although moving from the default may make the whole idea of history less useful.

Other Ways to Search

The history file can also be used in other ways. For example, you can pipe `history` through the `grep` command, searching for a string with `history | grep SEARCH` (Figure 4). Another use of the history file is sometimes called a reverse `-i` search. To use this function, press Ctrl + R and begin entering letters. As in tab completion, each letter entered narrows down the match. When you find the command you want, press Enter to activate it (Figure 5).

As this overview shows, the `history` command has a greater ability to search than many users realize. It will take practice to find what features fit into your work habits, but learning the ones that work best for you will add just enough sophistication to make working at the command line easier – especially when you have the environmental variables set up to your satisfaction. ■■■

Info

[1] Bash history man page: *https://ss64.com/bash/history.html*

What?!

I can get my issues SOONER?

Sign up for a digital subscription and learn the latest techniques for better network security, system management, troubleshooting, performance tuning, virtualization, and cloud computing on Windows, Linux, and popular varieties of Unix.

The sys admin's daily grind: Let's Encrypt wildcards

Encrypting on the Wild Side

The pleasure of owning a nice domain like *sensorenresidenz.de* is clouded by the requirement of an X.509 certificate for every subdomain that the admin wants or has. Columnist Charly can help boost the webmaster's spirits. *By Charly Kühnast*

Certificates from Let's Encrypt [1] are free and popular – supposedly half of all digital certificates used on the web originate there. However, anyone running web servers with many subdomains has had the displeasure of having to manage a bag full of certificates. A single wildcard certificate from one of the usual certification bodies solves the problem in principle, but it is expensive.

This changed in March 2018: All those who have access to their domain name's DNS server can now also obtain wildcard certificates from Let's Encrypt.

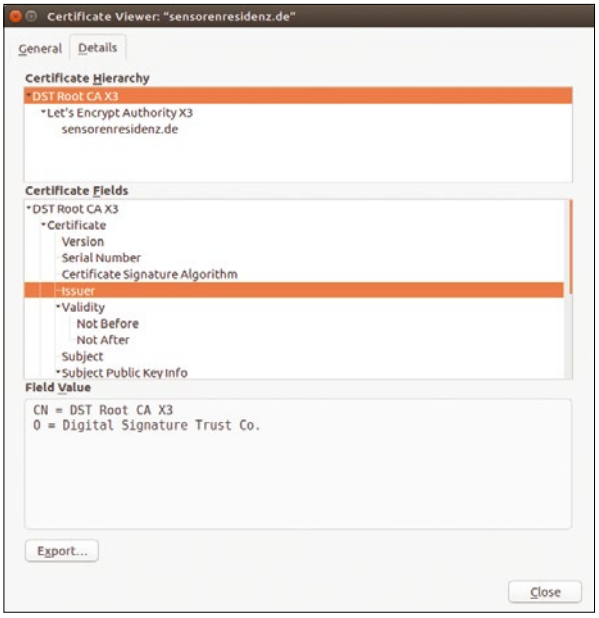


Figure 1: Browsers – here Firefox – accept wildcard certificates from Let's Encrypt without any restrictions.

If you have not yet installed the Let's Encrypt client, you can load it as follows:

```
git clone ➥
  https://github.com/➥
  letsencrypt/letsencrypt
```

After that, web admins usually have to add some Python packages. That's easily taken care of with

```
letsencrypt/➥
  letsencrypt-auto --help
```

which will ask you for your `sudo` password when launched. Let's Encrypt uses a DNS-based challenge-response method for authentication.

Let's Dance

The following command lets me request a certificate for the namespace `*.sensorenresidenz.de` (just don't ask why I registered the domain):

```
sudo /usr/local/letsencrypt/➥
  letsencrypt-auto certonly --manual ➥
  --preferred-challenges dns ➥
  --server https://acme-v02.api.lets➥
    encrypt.org/directory ➥
  --email charly@kuehnast.com ➥
  --domains *.sensorenresidenz.de
```

After a second of recollection, the Let's Encrypt client answers. It wants me to create a TXT record named `_acme-challenge.sensorenresidenz.de` in the DNS with a given random string as content:

```
Please deploy a DNS TXT record under the ➥
 name _acme-challenge.sensorenresidenz.de ➥
  with the following value:
ST8ehm-bKS6wRAxZk5vYDzUO9-OEr3Ne➥
  IpClFyaq-kA
Before continuing, verify the record ➥
  is deployed.
```

After I have created the entry and checked that it is available, I press Enter. Let's Encrypt retrieves my request and verifies it. Then it creates a certificate (`fullchain.pem`) and key (`privkey.pem`) and stores both under `/etc/letsencrypt/live/Name` – applause (Listing 1).

Integrating the certificate with the web server follows the usual steps. The browser is happy with the certificate that my servers now deliver (Figure 1).

Wildcard certificates from Let's Encrypt expire after 90 days like all other certificates from the issuer. To renew them, you simply call `letsencrypt-auto renew`. ■■■

Info

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

Listing 1: I did it!

```
Congratulations! Your certificate and chain have been saved at:
  /etc/letsencrypt/live/sensorenresidenz.de/fullchain.pem
  Your key file has been saved at:
  /etc/letsencrypt/live/sensorenresidenz.de/privkey.pem
  Your cert will expire on 2018-08-07. To obtain a new or tweaked version of this
  certificate in the future, simply run letsencrypt-auto again. To
  non-interactively renew *all* of your certificates, run "letsencrypt-auto
  renew".
```

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.

IT Highlights at a Glance

Too busy to wade through press releases and chatty tech news sites? Let us deliver the most relevant news, technical articles, and tool tips – straight to your Inbox. Subscribe today for our excellent newsletters:

ADMIN HPC • ADMIN Update • Linux Update

and keep your finger on the pulse of the IT industry.

Admin and HPC: www.admin-magazine.com/newsletter

Linux Update: www.linux-magazine.com/newsletter

Photo by Parker Byrd on Unsplash

Open source libraries for facial recognition

I Know That Face

It is not just Facebook – every Linux user can extract faces from photos and assign them to real people, thanks to free libraries. Mike Schilli shows you how to do it. *By Mike Schilli*

Facebook users already take it for granted that the social network recognizes people from their circle of friends by their faces on uploaded pictures. Some free libraries, which every Linux user can download from GitHub, also extract faces from photos and compare them with previously recognized ones, thus allowing the home user to recognize people (e.g., in their private vacation photo collection) and to mark the images accordingly.

Much goes on behind the scenes in automatic face recognition. First, an algorithm has to pick out a face-like object from the millions of pixels in a photo (Figure 1): Two round, slightly darker areas as the eyes; a protruding object in the middle as the nose; a horizontal line below it as the mouth; and another below it as the chin – that could be a face (Figure 2).

A good face recognition program not only recognizes full-screen faces on portrait photos, but also faces that only cover a few hundred pixels due to the subjects being further away or warped because they are looking into the camera from an angle.

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.

Assured After Training

Anything that could be a face needs to be extracted from the background noise of the image. Neural networks [1] are excellent for this purpose. They do not act on fixed pixel values, which wouldn't work, because even two photos of the same person differ enormously at the pixel level. Instead, the network is trained with millions of different faces during the learning phase and then recognizes everything that looks remotely similar with a small remaining margin of error.

Installation Made Easy

Hosted on GitHub, Adam Geitgey's `face_recognition` project [2] uses the popular `dlib` library to recognize faces in photos and videos. With

```
git clone https://github.com/ageitgey/↵
  face_recognition.git
```

This `git` command creates a clone of the GitHub repository in a local directory.

The `docker` file in the top directory handles the installation of the dependent projects. The command

```
docker build -t face .
```

in the project directory loads everything off the web, compiles `dlib`, and also loads a 100MB-heavy neural model that is already trained about faces into the container. A coffee break would be a good thing while this is going.

To call the script in Listing 1 for face recognition in the container, the user copies it to the `examples` directory of the GitHub repository's clone and calls

```
docker run -v `pwd`:/build ↵
  -it face bash -c "cd /build; ↵
  python3 face-box.py pic.jpg"
```

This binds the current directory (where both the Python script and the test image are located) to the `/build` directory in the container. The Bash com-

Figure 1: The author hiking in the Arizona desert.

Figure 2: The `face-box.py` script identified a face in the photo and framed it.

mand changes to that directory and calls `face-box.py` in the container where the face recognition tool is installed. The Bash command passes in the image name `pic.jpg` as a parameter to `face-box.py`; the photo shows yours truly in the hellishly hot desert of Arizona (Figure 1). The script picks out the face under the baseball cap, marks it by darkening the area, and then generates the file `pic-box.jpg` with the resulting data. Since the current directory is bound to the container, the box file also resides right there, even outside of the container directory after the `docker` command has been completed.

Listing 1: face-box.py

```
01 #!/usr/bin/python3
02 import face_recognition as fr
03 import sys
04 from PIL import Image, ImageDraw
05
06 try:
07   _, img_name = sys.argv
08 except:
09   raise SystemExit(
10      "usage: " + sys.argv[0] + " image")
11
12 img   = fr.load_image_file(img_name)
13 faces = fr.face_locations(img)
14
15 pil = Image.fromarray(img)
16 pil = pil.convert("RGBA")
17
18 tmp  = Image.new(
19         'RGBA', pil.size, (0,0,0,0))
20 draw = ImageDraw.Draw(tmp)
21
22 for (y0, x1, y1, x0) in faces:
23   draw.rectangle(((x0, y0), (x1, y1)),
24           fill=(30, 0, 0, 200))
25 del draw
26 pil = Image.alpha_composite(pil, tmp)
27 pil = pil.convert("RGB")
28
29 img_name = img_name.replace(".", "-box.")
30 pil.save(img_name)
```

Listing 1 [3] is quite lean, with only 30 lines, underlining the project's claim to being "The world's simplest facial recognition" [2]. The actual face recognition is executed using `face_locations()` (line 13) against a JPEG image loaded with `load_image_file()`. Back comes a list with rectangular coordinates of recognized faces, because the algorithm does not only search for one face in images with several persons, but finds all in one go. The `for` loop in line 22 iterates across all blocks of four of these coordinates and paints a semitransparent dark rectangle on the face coordinates using the `ImageDraw` class from the Python Imaging Library (PIL) treasure trove.

Not Opaque

For transparency in the dark overlay, the image needs a temporary canvas in `tmp` with an alpha channel (image mode RGBA) with a relatively high transparency value (200 of 255 in line 24). The `fill` parameter for the color to be used is set to light red (30,0,0). However, PIL cannot turn an image with an alpha channel into a JPEG, so line 27 turns it into an RGB image without an alpha channel before `pil.save()` stores the JPEG under a name with a `-box` extension. The procedure works relatively reliably, apart from a few blatant outliers, as shown in Figure 3 of a stalactite cave whose stalactites (Figure 4) the neural network thinks represent my facial features.

Not Perfect, but Not Bad

But `face_recognition` can do more than find faces in pictures; it can identify the person to whom the face belongs. To compare two faces found in different images, the algorithm again cannot simply match raw images pixel by

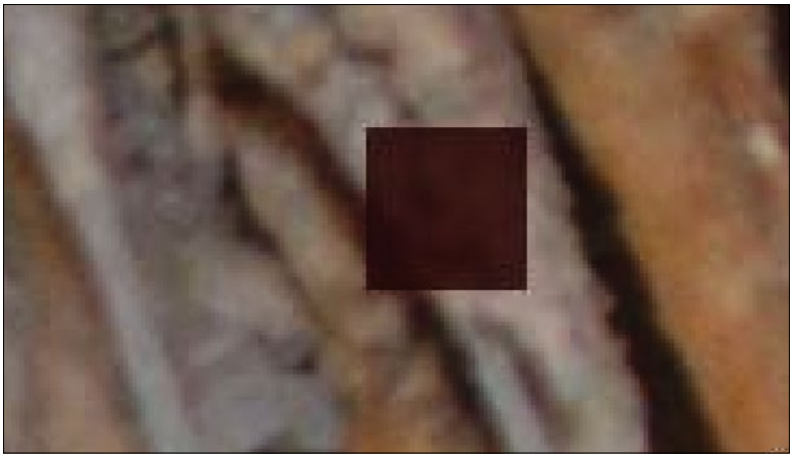


Figure 3: The artificial intelligence module thinks it has identified the author here …

Figure 4: … but it's seriously mistaken.

pixel. Rather, it has to normalize, equalize, and then extract a series of features.

A person might pay attention to the size of the nose, the color of the eyes, the forehead height, or the thickness of the eyebrows. The facial recognition algorithm, on the other hand, learns which features produce the most hits and the least false positives with millions of test images in the learning phase, based on matching and mismatching images. Afterward, however, the algorithm only consists of meaningless columns of numbers. As is usual in machine learning, no one knows which particular feature the algorithm uses to arrive at a particular decision.

Figure 5 shows the key data of the reference face extracted from Figure 1 provided by the `face_encodings()` function. The facial comparison algorithm in turn takes the key data from each recognized face and compares them to the reference. If two records approximately match, it is probably the same person.

With this tool, a script can extract a face from the reference image and compare the result with faces on other images. As a practical application, I have whipped up the script in Listing 3, which searches my own photo collection (containing an impressive 36,525 images) for images showing the person on the reference image – me.

The file hierarchy is based on what I call the shoe box principle, meaning new photos just get dumped into there without any extra archiving or indexing. Whittling down the collection by hand would be very labor intensive. But I can certainly show an artificial intelligence (AI) system the photo from Figure 1 and rattle through the image collection to see if the face in Figure 1 can also be detected on other photos.

Unpacking the Whole Collection

To do this, Listing 2 first defines an iterator for all JPEG photos below the `/photos` directory on my hard disk. It skips other formats and all entries in `.cache` directories where one of my image processing programs stores the thumbnails that I want to leave out of the face analysis action. The `photos()` iterator as of line 5 accepts the start directory and then runs through all the files it finds; the `yield()` operator returns them in line 12 bit by bit, when the main program asks for more.

In the main image finder in Listing 3, lines 8 to 12 check whether the user has specified both a reference image and the top search path for the photos at the command line. The first element of `sys.argv` contains

```
$ cat face-values
import face_recognition as fr
ref_img  = fr.load_image_file("me.jpg")
print(fr.face_encodings(ref_img)[0])

$ docker run -v `pwd`:/build -it face bash -c "cd /build; python3 face-values"
[-0.13315442 -0.00168471 -0.03589345 -0.09885187 -0.05589457 -0.01517804
 -0.01199222 -0.04692851  0.18923387 -0.05561652  0.16054742  0.02832796
 -0.25407898 -0.10978155  0.10285731  0.07996124 -0.16108523 -0.11483447
 -0.0959759  -0.14412737 -0.04713656  0.05746659 -0.04691344  0.02860871
 -0.21485221 -0.23155726 -0.11872631 -0.20538138  0.07044239 -0.09604892
  0.06296323  0.08331566 -0.15413234 -0.04659536  0.02606722  0.03737376
 -0.12634644 -0.02914073  0.19478095 -0.00208588 -0.11011767 -0.08763562
  0.02516873  0.25372446  0.2212179   0.00418672  0.0573585  -0.07569446
  0.17196316 -0.22886412  0.04987632  0.17908913  0.1044035   0.1524764
  0.12472308 -0.13007215  0.04473969  0.22061433 -0.25664231  0.10765717
  0.03402605 -0.08175991 -0.06325922 -0.03277334  0.19319049  0.04879164
 -0.12550919 -0.1290925   0.22415447 -0.16593254 -0.05105753  0.15125556
 -0.12739624 -0.23852426 -0.25480482  0.07302041  0.25291386  0.15024373
 -0.1028873   0.03625717 -0.03517125 -0.084273    0.05481945  0.04918346
 -0.04941802 -0.14561139 -0.00281725  0.03422378  0.20433143 -0.03039287
  0.08097647  0.1607566   0.00734734 -0.05263064 -0.02914657  0.05739972
 -0.09913284 -0.01379542 -0.02229789  0.01794208  0.05195126 -0.09492856
  0.03394579  0.08316693 -0.10980687  0.24001823  0.02950187 -0.0301931
 -0.08160857 -0.11022182 -0.02169624  0.01768621  0.18095559 -0.27287441
  0.25480813  0.14042208 -0.03049276  0.14340019 -0.06296419  0.08113439
 -0.08278649 -0.06418354 -0.20503788 -0.05525303  0.06264418 -0.02229233
 -0.02613626 -0.01446014]
```

Figure 5: The key data of the author's face from Figure 1.

Listing 2: photos.py

```
01 #!/usr/bin/python3
02 import os
03 import re
04
05 def photos(dir):
06   for root, dirs, files in os.walk(dir):
07     if re.search(r'\.cache', root):
08         continue
09     for file in files:
10       if re.search(r'jpg$', file,
11                    re.IGNORECASE):
12         yield(os.path.join(root, file))
13
14   # testing
15 if __name__ == "__main__":
16     for photo in photos("/photos"):
17         print(photo)
```

Listing 3: face-search.py

```
01 #!/usr/bin/python3
02 import face_recognition as fr
03 import dbm
04 import re
05 from photos import photos
06 import sys
07
08 try:
09   _, ref_img_name, search_path = sys.argv
10 except ValueError as e:
11   raise SystemExit("usage: " +
12     sys.argv[0] + " ref_img search_path")
13
14 cache = dbm.open('cache', 'c')
15
16 ref_img  = fr.load_image_file(ref_img_name)
17 ref_face = fr.face_encodings(ref_img)[0]
18
19 for photo in photos(search_path):
20   if photo in cache:
21     print(photo + " already seen")
22     continue
23   cache[photo] = "1"
24
25   try:
26     img = fr.load_image_file(photo)
27   except:
28     continue
29
30   for face in fr.face_encodings(img):
31     hits = \
32       fr.compare_faces([ref_face], face)
33     if any(hit for hit in hits):
34       print(photo)
```

the script name, which gets discarded in the underscore variable (_); then, line 16 loads the reference image. The next line extracts the reference face, shown under index `0` of the returned coordinate list because the reference image only has one face in it.

Later, line 19 runs through all JPEG files found by `photos.py`; for each one, the script calls the `compare_faces()` function from the `face_recognition` project with the face values previously obtained from the reference image. The

```
any(hit for hit in hits)
```

construct checks whether one of the faces detected on the current image matches the one on the reference image. If this happens, one of the elements in the `hits` list has a value of `True`, and line 34 prints the image file's path to standard output, where the surprised user can pick it up and inspect it with a photo viewer.

Listing 4 shows how the script calls into the Docker container and displays its output. I was amazed by the photos it dug up, some from ancient history, unveiling a more youthful edition of yours truly. Oh, the good old days!

So Many Nerds

However, the process is not perfect and occasionally makes downright laughable mistakes. For example, my collection had some pictures I took at open source conferences showing hundreds of young nerds, and the algorithm totally thought that one of them was me, which is impossible because I actually took the photos.

Since AI wastes lots of computing time, rummaging through the image tree and recognizing familiar faces takes a long time. If the script bombs out somewhere due to an error, it would be unfortunate to have to start over. Therefore, Listing 3 remembers results from all processed images in a persistent file named `cache`. Python closes it conveniently when the program terminates, so the script only has to open it at the beginning with the `c` flag (to create it for the first time if necessary). The script can subsequently access the `cache` dictionary to see whether it already contains the name of the file under investigation and skip it if so.

Brave New World

Since the process can extract and classify all the faces in an image, it opens up completely new possibilities for obtaining information. Potent methods of surveillance can be derived from collecting a large number of images, for example, using surveillance cameras positioned in public spaces. Armed with a reference photo of a person of interest, an AI system could easily find out which other people the suspect typically hangs out with in public – which could be put to good or bad use, depending on who's in charge of surveillance and for what purpose. ■■■

Info

[1] Machine learning: *https://medium.com/@ageitgey/machine-learning-is-fun-part-4-modern-face-recognition-with-deep-learning-c3cffc121d78*

[2] face_recognition: *https://github.com/ageitgey/face_recognition#face-recognition*

[3] Listings for this article: *ftp://ftp.linux-magazine.com/pub/listings/linux-magazine.com/214/*

Listing 4: run.sh

```
1 $ docker run -v /photos:/photos -v `pwd`:/build -it face bash
    -c "cd /build; python3 face-search.py me.jpg /photos"
2 /photos/2001/12/29/13:55:38.jpg
3 /photos/2001/07/22/11:47:27.jpg
4 /photos/2001/07/22/10:35:33.jpg
5 /photos/2001/07/22/15:43:23.jpg
6 [...]
```

ADMIN

Network & Security

COMPLETE ARCHIVE DVD

ISSUES 00-39

OVER 3,000 PAGES OF UP-CLOSE SYSTEM ADMINISTRATION

ISSUE 40/2017

While this ADMIN magazine disc has been tested and is to the best of our knowledge free of malicious software and defects, Admin 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.

DVD ROM

Smart and Practical Information for Network Professionals

Develop apps with React Native and NativeScript

Purely Native

The JavaScript frameworks React Native and NativeScript build a bridge between web app development and native app development. We look at the advantages of these native frameworks against the background of the classic app framework Meteor.

By Andreas Möller and Kristian Kißling

Photo by Jordan McQueen on Unsplash.com

Two open source JavaScript frameworks, React Native [1] and NativeScript [2], help programmers develop native apps for Android and iOS by using their IDEs with JavaScript. This approach has advantages over classical web app development in the style of, say, the JavaScript Meteor [3] framework, which relies on Apache Cordova [4] and WebView.

In the Beginning

Native apps for Android are traditionally programmed with Java or Kotlin; iOS uses Objective-C or Swift [5]. Because Android apps access native code internally with C or C++, developers rely on Java Native Interface (JNI) [6]. The Android Studio [7] and Xcode [8] development environments help in programming an application and bringing it to the corresponding app stores.

However, this approach has drawbacks: Installing Android Studio is quite complex, and Xcode runs exclusively on Apple's Mac OS X operating system. If you also want to adopt an Android app for iOS, you have to reprogram the code one to one under Objective-C or Swift. However, this significantly increases the development effort and creates additional sources of error through code redundancy.

Classic web apps use HTML, CSS, and JavaScript and run across platforms in a browser on any system. However, the platforms restrict access to the system with the help of the browser sandbox and only selectively release APIs, which limits the capabilities of web apps.

In 2009, PhoneGap (now Apache Cordova) was released, which allowed web apps to be connected to the system in a better way. Under Cordova, apps run in WebView on the system as in a browser. Plugins extend access to the system for the apps.

Meteor

The Meteor [3] JavaScript framework has evolved from a framework for developing real-time web apps [9] to an open source platform that claims to be the fastest way to launch apps. In fact, installing and creating a bootstrap app in the shell under Ubuntu 17.10 with

```
curl https://install.meteor.com/ | ↵
  sh meteor create example
```

is quite easy. However, an attempt to build the app for Android with the command `meteor add-platform android` aborts (Figure 1), because, internally, Meteor uses Cordova, which requires Java and Android Studio. Moreover, a build for iOS is not possible on Ubuntu.

```
pa@tarantos: ~/whatsapp
npm WARN excluding symbolic link node_modules/.bin/shjs -> ../shelljs/bin/shjs
Subproject Path: CordovaLib                    /
android: added platform

Your system does not yet seem to fulfill all requirements to build apps for
Android.

Please follow the installation instructions in the mobile guide:
http://guide.meteor.com/mobile.html#installing-prerequisites

Status of the individual requirements:
✗ Java JDK: Failed to run "javac -version", make sure that you have a JDK
  installed.
  You can get it from: http://www.oracle.com/technetwork/java/javase/downloads.

✗ Android SDK: Failed to find 'ANDROID_HOME' environment variable. Try setting
  it manually.
  Failed to find 'android' command in your 'PATH'. Try update your 'PATH' to
  include path to valid SDK directory.
✗ Android target: android: Command failed with exit code ENOENT
✗ Gradle: Could not find gradle wrapper within Android SDK. Could not find
  Android SDK directory.
  Might need to install Android SDK or set up 'ANDROID_HOME' env variable.
```

Figure 1: Cordova: Without Java and Android Studio, nothing works for Android developers under Ubuntu.

When creating an app, you can at least choose between Meteor's own JavaScript framework Blaze, AngularJS [10], and React [11]. However, Meteor offers less support for the programmer when it comes to deployment. Although `meteor build` packages versions for Android and iOS, it takes some manual work to get the apps into the respective app stores.

React Native

Native frameworks take a slightly different approach: With JavaScript help, they promise native apps for Android and iOS and thus build a bridge between web apps and platform-specific apps.

The React Native [1] framework leaves the underlying system to the JavaScript engine at run time. The code runs in its own thread and accesses the native APIs and native code of the system via a bridge [12] (Figure 2). The graphical user interface (GUI) is not created with the Web-View Document Object Model (DOM) tree, as under Meteor, but with calls to native UI widgets in Java or Objective-C.

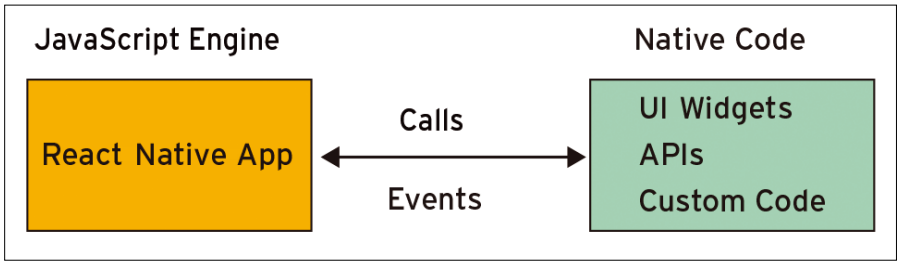


Figure 2: A bridge connects the JavaScript world with native code.

Listing 1: Installing Expo on Ubuntu 17.10

```
01 curl -sL https://deb.nodesource.com/setup_8.x |  sudo -E bash -
02 sudo apt-get install nodejs
03 sudo npm install exp --global
04 exp init react-test
05 cd react-test
06 exp start
```

Figure 3: Expo offers the app for uploading in the Expo client like a website.

When the user triggers events by using the rendered elements, the React Native system passes them back across the bridge and converts them into a JavaScript event. React Native counteracts performance losses caused by synchronizing both worlds with a virtual DOM, for which the JavaScript React framework provides the model. In practical tests, React Native's sample apps reacted more or less as slowly as native apps under an older iOS version.

If you want to design React Native apps, the Expo [13] project will help with a number of tools, including the Expo Development Environment (XDE), which also launches in the browser under the name Snack [14]. At the same time, it can be operated from the command line after executing the installation instructions in Listing 1.

Lines 1 and 2 install the current version of Node.js v8 with Debian's Apt package manager. The Node package manager, `npm`, then fetches Expo (`exp`), including React Native, and installs it on the system in line 3; line 4 then sets up the boilerplate application `react-test`. Lines 5 and 6 package the app for loading

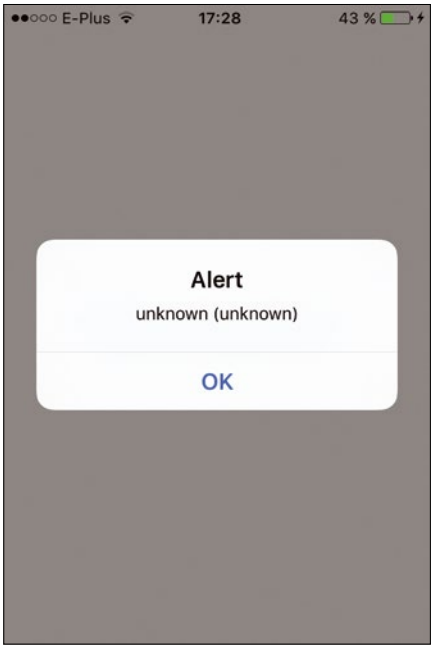


Figure 4: The app uses the `UIAlertController` class on iOS.

from a smartphone. Figure 3 shows the shell after executing the command chain, together with a QR code for installation.

To transfer the packaged app with the QR code to an Android device or iPhone, the user must first install on the device the Expo client of the same name from the corresponding app store by launching the client, scanning the QR code, and loading the boilerplate app (Figure 4).

If the requirements for an app exceed the limits of React Native and Expo, you can retrofit native code, which is then controlled remotely with JavaScript in the app. In this fallback scenario, however, you do need to install Android Studio and Xcode.

Example App

Listing 2 shows how developers can use React Native. The code stores the `App.js` file in the `react-test` directory. Like any application under React Native, the app in Listing 2 is based on the JavaScript React [11] framework, imported in line 1. Line 2 then retrieves the `StyleSheet`, `Button`, `View`, and `NetInfo` components from React Native.

React Native also forms the `App` base class (lines 4-15) as a component. Its render method (lines 5-11) uses the XML-like JSX code to create an area-specific user interface (lines 7-9). In line 8, the `View` component encapsulates a button [15]. Android implements this at run time with an instance of the `android.widget.Button` class [16]. iOS, however, uses `UIButton` [17].

Lines 12 to 14 store the callback function `getConnectionInfo()`, to which the value of the `onPress` attribute points (line 8). When the button is pressed, the button method determines the connection status of the mobile device in line 13. The lambda function responds to the asynchronous result by calling the `then()` method, which should report the determined connection type in an alert box. As Figure 4 shows, the connection status could not be determined in the field test for unknown reasons.

The `{styles.container}` code reference from the value of the `style` attribute binds the stylesheet information from the static `styles` JavaScript object to the `View` component in line 7. These details paint the background in a gray tone and place the button in the center of the display (lines 17-24). In addition to `NetInfo`, React Native offers a number of other system-related API objects that extend the SDK.

The `exp start` command in the last line of Listing 1 acts like a watch process. If it registers a change to the project directory, it informs the packager server, which has also been started. The server arranges for the updated app to be delivered to connected Expo clients.

As the sample app `react-test` [18] demonstrates, a programmer can publish an app free under their Expo account in the project's cloud store at any time. This is triggered by clicking the *Publish* button in XDE. If you also add `buildidentifier` and `package` to the `app.json` configuration file in the project directory, then `exp build:android` creates an Android app within the provider cloud.

With a URL under the Expo account, you can track the build log of the process live at the command line. Figure 5 shows the build log after successfully building the Android package. Behind the scenes, Expo uses instances of Android Studio or Xcode from its cloud. As with Meteor, however, the providers have not completely automated the last few steps in the app stores; some manual work is again needed.

NativeScript

The NativeScript [2] JavaScript framework also promises to launch native apps for Android and iOS. Like React Native, NativeScript leaves it to the respective JavaScript engine to execute the code. Android v8 and iOS JavaScriptCore (JSC) are used.

NativeScript also controls native code remotely via a bridge [19]. The runtime environment dynamically passes on calls to getter and setter methods of a JavaScript object to native objects.

Setting up the development environment for NativeScript proves to be particularly tricky on Linux. Builds of iOS apps fail for known reasons, anyway. If you are not limited by cost (see Table 1), you can create the apps in the provider's cloud as you would under Expo. At least the first 100 builds are free.

Listing 2: React Native Sample App

```
01 import React from 'react';
02 import { StyleSheet, Button, View, NetInfo } from 'react-native';
03
04 export default class App extends React.Component {
05   render() {
06     return (
07       <View style={styles.container}>
08         <Button title="Connection Info" onPress={this.getConnectionInfo}/>
09       </View>
10     );
11   }
12   getConnectionInfo() {
13     NetInfo.getConnectionInfo().then(info => alert(info.type+' ('+info.effectiveType+')'));
14   }
15 }
16
17 const styles = StyleSheet.create({
18   container: {
19     flex: 1,
20     backgroundColor: '#ddd,
21     alignItems: 'center',
22     justifyContent: 'center',
23   },
24 });
```

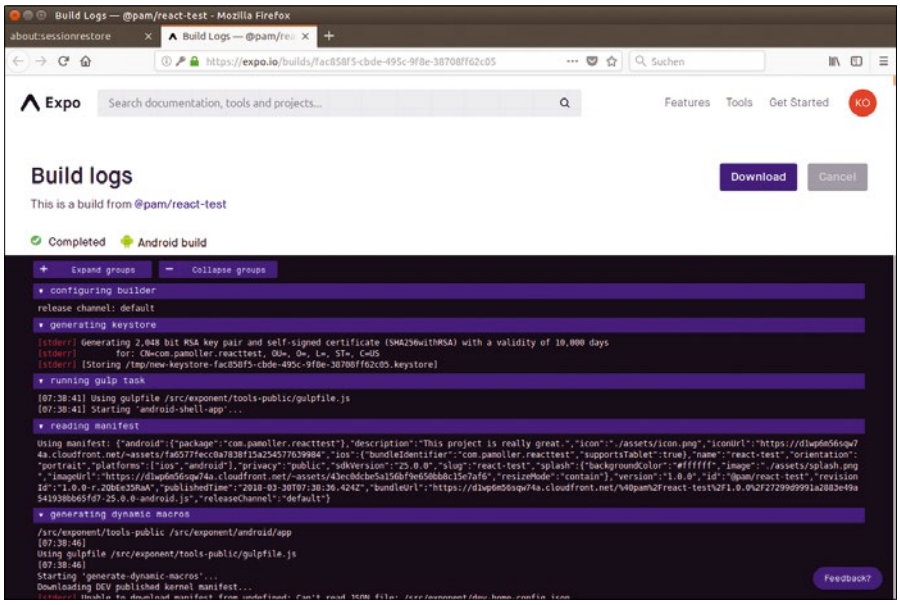


Figure 5: Expo provides insight into the logfiles of its cloud builds.

The command

```
sudo npm install -g nativescript
```

installs the JavaScript framework on Ubuntu. A suitable bootstrap app is then created:

```
tns create ns-test --template ➥
  nativescript-template-ng-tutorial
```

The Sidekick [20] graphical development environment installs Debian's `dpkg` package manager after downloading with:

```
sudo dpkg ➥
  -i NativeScriptSidekick-amd64.deb
```

Then, Sidekick launches with:

```
/opt/Native\ Script\ Sidekick/Native\ ➥
  Script\ Sidekick
```

Unlike React Native, NativeScript supports all kinds of scripting languages and frameworks: Besides AngularJS [10] and Vue.js [21], you can also use pure JavaScript or TypeScript [22].

Listing 3 returns to the sample app from Figure 4 and Listing 2 under AngularJS and NativeScript. The code ends up in the `app/app.component.ts` file in the `ns-test` directory under the previously created bootstrap app. Line 1 of Listing 3 imports the `Component` class from AngularJS; line 2 imports the `connectivity` object from NativeScript. The decorator in lines 4 to 7 converts the subsequent class definition of the basic `AppComponent` class (lines 8-22) into a derivative of the `Component` class.

Line 6 creates the area-specific UI using the `Button` component to generate a button, which calls the `getConnectionInfo()` callback function when tapped. The NativeScript code determines the connection type synchronously, and the `alert()` function reports the result to the user.

Figure 6 shows the app's successful Sidekick for Android cloud build. We did not tested the cloud build for iOS, because the required iOS developer certificate – in contrast to the documentation – is now commercial.

Listing 3: Sample App with NativeScript

```
01 import { Component } from "@angular/core";
02 import * as connectivity from "connectivity";
03
04 @Component({
05   selector: "my-app",
06   template: `<Button text="Connection Info" (tap)="getConnectionInfo()"></Button>`
07 })
08 export class AppComponent {
09   getConnectionInfo() {
10     switch(connectivity.getConnectionType()) {
11       case 0:
12         alert('none');
13         break;
14       case 1:
15         alert('WiFi');
16         break;
17       case 2:
18         alert('Mobile');
19         break;
20     }
21   }
22 }
```

Conclusions

Meteor apps are easy to set up and scale thanks to the Galaxy cloud service. Things only start to go wrong when creating mobile apps.

React Native and NativeScript offer equivalent solutions. They do without

Table 1: Frameworks Compared

Framework	IDE	Cloud Builds	Native UI	Native Modules	Native Packages	App Store Deployment
Meteor 1.6.1	No	No	No	Yes	Yes	No
React Native 52.0	XDE 2.22.1	Yes	Yes	Yes	Yes	No
NativeScript 3.4.3	Sidekick 1.5.1	Commercial	Yes	Yes	Yes	No

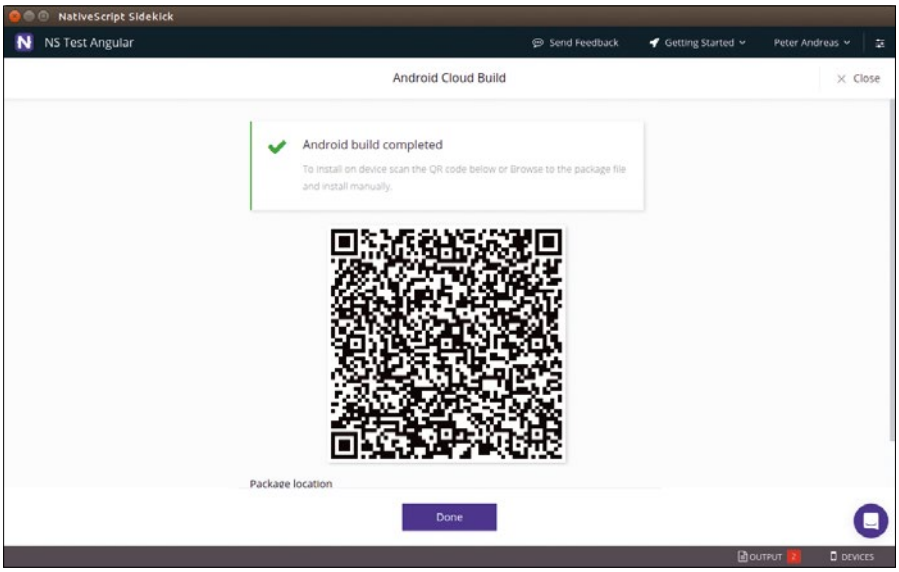


Figure 6: Sidekick after the cloud build for Android. The version for iOS is now commercial.

WebView and use native code for the iOS and Android app interfaces. Both native APIs and UI components use a bridge. For app developers, this feels like working with a browser and DOM, and the native UIs improve the user experience.

The Expo project complements React Native. With the Expo client, the web-based Snack IDE, and free cloud storage, Expo offers an ideal environment for new projects and their communities. Thanks to cloud builds, developers no longer have to go to the trouble of configuring and installing Android Studio and Xcode. ■■■

Info

[1] React Native: *https://facebook.github.io/react-native/*

[2] NativeScript: *https://www.nativescript.org*

[3] Meteor: *https://www.meteor.com*

[4] Apache Cordova: *https://cordova.apache.org*

[5] Swift: *https://developer.apple.com/library/content/documentation/Swift/Conceptual/Swift_Programming_Language/index.html*

[6] Java Native Interface: *https://developer.android.com/training/articles/perf-jni.html*

[7] Android Studio: *https://developer.android.com/studio/index.html*

[8] Xcode: *https://developer.apple.com/xcode/*

[9] "Real-Time Web Applications with the Meteor Framework" by Andreas Möller, *Linux Pro Magazine*, issue 161, April 2014, pg. 46, *http://www.linuxpromagazine.com/Issues/2014/161/Meteor*

[10] AngularJS: *https://angularjs.org*

[11] React: *https://reactjs.org*

[12] React Native bridge: *http://www.discoversdk.com/blog/how-react-native-works*

[13] Expo: *https://expo.io*

[14] Snack: *https://snack.expo.io*

[15] React Native button: *https://facebook.github.io/react-native/docs/button.html*

[16] Android button: *https://developer.android.com/reference/android/widget/Button.html*

[17] iOS button: *https://developer.apple.com/documentation/uikit/uibutton*

[18] Sample app `react-test`: *https://expo.io/@pam/react-test*

[19] NativeScript bridge: *https://developer.telerik.com/featured/nativescript-works/*

[20] Sidekick: *https://www.nativescript.org/nativescript-sidekick*

[21] Vue.js: *https://vuejs.org*

[22] "CoffeeScript, Dart, Elm, and TypeScript" by Tim Schürmann, *Linux Pro Magazine*, issue 202, September 2017, p. 42, *http://www.linuxpromagazine.com/Issues/2017/202/JavaScript-Alternatives*

Too Swamped to Surf?

Our ADMIN Online website offers additional news and technical articles you won't see in our print magazines.

Subscribe today to our free ADMIN Update newsletter and receive:

- Helpful updates on our best online features
- Timely discounts and special bonuses available only to newsletter readers
- Deep knowledge of the new IT

ADMIN Network & Security

www.admin-magazine.com/newsletter

MakerSpace

Arduino programming for open hardware projects

Getting Started

Whether you are building your own device or customizing one, Arduino technology often functions as the foundation for your open hardware project. Here's how to get started with Arduino products. *By Bruce Byfield*

Arduino single-board microcontrollers [1] were originally intended for educational and do-it-yourself projects. Increasingly, though, they are a foundation for Internet of Things (IoT) projects and open hardware. Sometimes, they are used only for prototyping, but, just as often today, they are found in commercial products ranging from keyboards to prosthetics. If you are involved in open hardware, knowing how to work with Arduino products is quickly becoming essential.

Arduino boards are influential for several reasons. To start, while limited in scope, Arduino boards are powerful enough for many dedicated purposes. If they are not, chaining them or using extension hardware known as shields can easily extend their usability. Just as importantly, Arduino boards can interact with Linux, OS X, or Windows, as well as other single-board microcontrollers like the Raspberry Pi.

Moreover, Arduino offers widespread support for all user levels. This support includes its own IDE and a programming language that is a variant of C, a Creative Commons license, and a community of thousands. With this support, users can begin to work without experience in electronics or programming. At the same time, Arduino is flexible enough to meet the needs of experts as well.

Regardless of whether you are building your own device or customizing a product that uses Arduino hardware, at the heart of working with an Arduino board is customizing the firmware for a board or shield. Exactly what you can do depends on the hardware with which you are working. Since nearly two dozen boards (Figures 1 and 2) and almost the same number of shields are currently available, detailed instructions would fill a book, if not several [2]. What follows is a general overview of how to get started, from preparing to work with the Arduino IDE to flashing the device's firmware.

Installing the Arduino IDE

Regardless of which Arduino board you are working with, the first step is to download and install the Arduino IDE. Different versions of the IDE are available for different operating systems. Especially if you are working with a newer board, you should install the latest version of the IDE, not the one available from your distribution's package repositories. Note that Linux has 32- and 64-bit versions, as well as an ARM version [3].

On Ubuntu, begin installation with the following commands:

```
cd ~/Downloads
tar xvf arduino-1.8.5-linux64.tar.xz
```

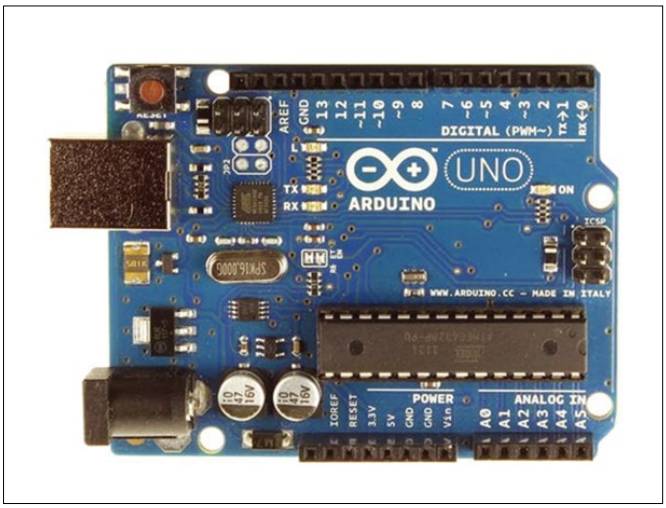


Figure 1: The UNO is a popular Arduino board for hobbyists.

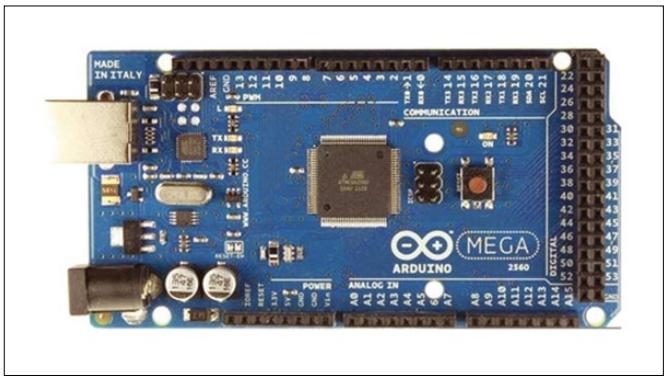


Figure 2: One of the more powerful Arduino boards, the MEGA 2560, is finding its way into commercial devices.

```
sudo mv arduino-1.8.5 /usr/local/arduino
cd /usr/local/arduino
sudo ./install.sh
```

Next, check that your regular account is in the `dialout` group by running the `group` command. If the username is not listed, add it with the commands:

```
sudo adduser $USER dialout
newgrp dialout
```

You will also need to add dependencies like GCC and Java, as well as make the `ARDUINO_PATH` part of your environment:

```
sudo apt-get install gcc-avr avr-libc
sudo apt-get install openjdk-6-jre
sudo update-alternatives --config java
export ARDUINO_PATH=/usr/local/arduino
```

Check for any additional instructions for the hardware and your Linux distribution. If you are using an Arduino clone – of which there are as many sold as true Arduino hardware – you may also need to install software for it as well.

To check that the IDE is ready for use, power up your Arduino board or device and plug it into a USB port that is not part of a USB hub without its own power supply, so that it can draw power from your computer. Use *Tools* | *Board* to make the IDE recognize the device, and, if necessary, *Tools* | *Port*. Then you can test that the IDE and the device are communicating by clicking *File* | *Examples* | *01.Basics* | *Blink* to make the LED on the Arduino board flash.

Using the Arduino IDE

The Arduino IDE has its own structure and jargon. In the `Arduino` folder created in your home directory the first time you run the IDE, code for the hardware is placed in the `hardware` directory. Files containing firmware source code are called sketches (`*.ino`) and each is installed in its own separate directory along with the resources required to work with it. Much of the programming can be done within the sketch, but lengthy modifications are sometimes stored in the `libraries` subfolders, together with general IDE resources.

You do not need to use the IDE to edit firmware. In fact, many users prefer to use their favorite editor instead. Nothing is wrong with using another editor, but to apply your modifications, be sure they are placed where the IDE can locate them. Moreover, the IDE is filled with useful tools and utilities, including:

- *File* | *Examples*: Small bits of code that show you what can be done with Arduino technology and that can be borrowed and modified as needed.

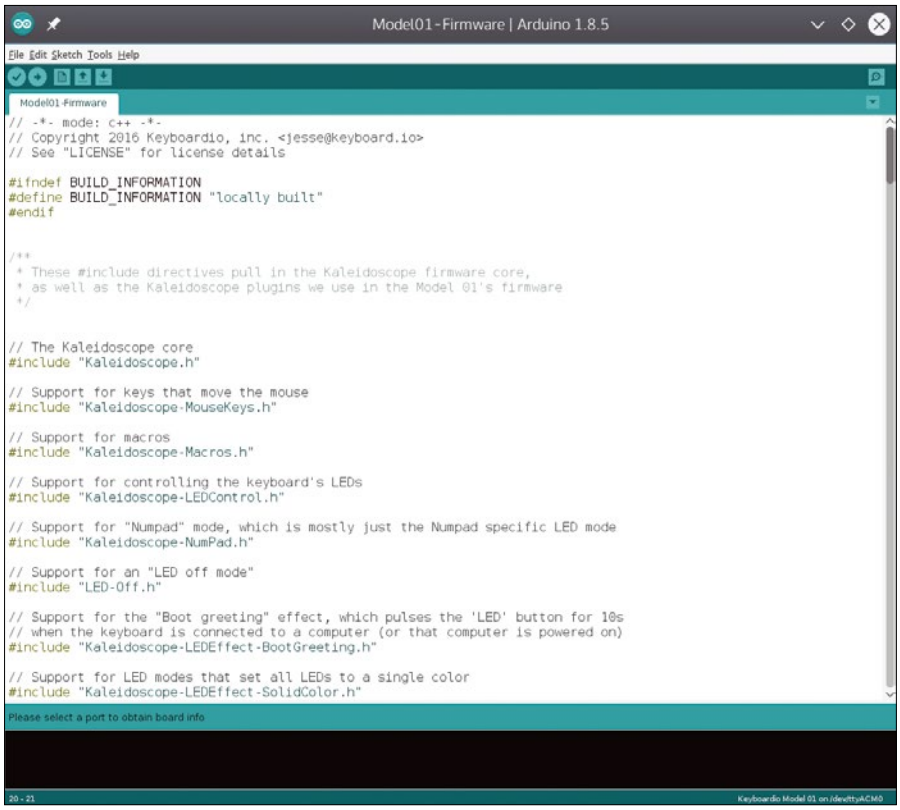


Figure 3: The Arduino IDE is both an editor and a collection of useful utilities.

```
bb@nanday:~/Arduino$ cd ./Model01-Firmware
bb@nanday:~/Arduino/Model01-Firmware$ make flash
BOARD_HARDWARE_PATH="/home/bb/Arduino/hardware" /home/bb/Arduino/hardware/keyboardio/a
vr/libraries/Kaleidoscope/bin//kaleidoscope-builder flash
Building output/Model01-Firmware/Model01-Firmware (0.0.0-gv1.22-7-gbf5d-dirty) ...
- Size: firmware/Model01-Firmware/Model01-Firmware-0.0.0-gv1.22-7-gbf5d-dirty.elf
  - Program:   20706 bytes (72.2% Full)
  - Data:       2039 bytes (79.6% Full)

Press ENTER when ready...
```

Figure 4: Before installing new firmware, the IDE compiles the source code.

- *Files* | *Preferences*: In addition to the general look and feel of the IDE, this dialog sets network connections and the verbosity displayed when code is run.
- *Tools* | *Archive Sketch*: Stores the current sketch, protecting it against being accidentally altered or overwritten.
- *Tools* | *Board Info*: Displays basic information about any board attached to the computer on which the IDE is installed. *Tools* | *Port* needs to be set before you can get information.

These tools are in addition to those found in the Editor menu for automating the writing of code and navigating within a sketch.

Programming

Arduino C, the programming language used to create Arduino firmware is a simplified version of C and C++. Unlike other versions of C, Arduino lacks the header section of a C program. Additionally, a sketch begins with a reserved routine called *setup*, which is executed only once and contains commands for initializing the hardware. The rest of a sketch is a routine called the *loop*, which is a block of statements that are repeated over and over until the hardware is disconnected or the sketch is overwritten. When you start a new file in the IDE, code for the setup and loop routines are added automatically. Otherwise, if you are familiar with C programming, you can probably adjust quickly to Arduino C (Figure 3).

If you are not familiar with C programming, you can do far more than you might imagine by modifying examples of the firmware provided with a particular device. You can begin by altering default settings set by the firmware and then experiment with code snippets on your own. You can click *Sketch* | *Verify/Compile* to check the validity of your experiments as you work. For those who prefer formal learning, the Arduino site has detailed information about Arduino C [4].

Installing Firmware

When a sketch and any plugins are ready, you can compile and flash the firmware. The process is similar to flashing the firmware on an Android device, the only difference being that you have full control of the process by default.

You can flash firmware either from the IDE or a command line. Within the IDE, open the sketch and select *Sketch* | *Upload*. From the command line, switch to the directory that contains the sketch to install and enter `make flash`. In both cases, the IDE compiles the firmware and often pauses before installing (Figure 4). This pause is to give you time to prepare, usually by pressing the reset button on the Arduino board, so that the bootload is bypassed for the firmware installation. On devices that run off an Arduino, like the Keyboardio Model 01, there may be a button to press, so that you do not have to unscrew the case to reset (Figure 5). While the firmware is flashed, LED lights will blink to show that the operation is happening. When the blinking stops and the process completes, unplug the board or the device and restart it. If the new firmware does not run the device properly, edit the firmware and try again. In a worst case scenario, you can restore an archived sketch that you know will work.

```
Looking for USB device with vid=1209 and pid=2301
Examining usb-Keyboardio_Model_01_CDkbio01-if00
  Found keyboard!
  ID_MM_DEVICE_IGNORE is set - good!
Looking for USB device with vid=1209 and pid=2300
Examining usb-Keyboardio_Model_01_kbio01-if00
  Found keyboard!
  ID_MM_DEVICE_IGNORE is set - good!
Connecting to programmer: .
Found programmer: Id = "CATERIN"; type = S
    Software Version = 1.0; No Hardware Version given.
Programmer supports auto addr increment.
Programmer supports buffered memory access with buffersize=128 bytes.

Programmer supports the following devices:
    Device code: 0x44

bb@nanday:~/Arduino/Model01-Firmware$
```

Figure 5: Installing the compiled firmware from the command line.

Next Steps

This article is a brief overview of working with Arduino technology. For any specific device, there may be additional steps required. Fortunately, the popularity of Arduino boards means that there is no shortage of additional, specialized resources. Begin with the Arduino Documentation [5] and branch out to other web resources, as well as the books available on Amazon. Just remember that Arduino technology is constantly evolving, so get the most recently written help available. Remember, too, that what is true of one board may not be true of others. You might want to buy a kit with detailed instructions to help familiarize yourself with some of the possibilities.

And if all the possibilities are too confusing, don't worry. One of the strengths of Arduino technology is its flexibility. Even without become an in-depth expert, you can still use Arduino technology to customize and improve devices to a far greater extent than you might imagine. ■■■

Info

[1] Arduino: *https://www.arduino.cc/*

[2] Boards and shields: *https://www.arduino.cc/en/Main/Products*

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

[4] Arduino C: *https://www.arduino.cc/reference/en/*

[5] Arduino Documentation: *https://www.arduino.cc/en/Main/Documentation*

MakerSpace

Minimally invasive beekeeping with a Raspberry Pi

Extend Your Feelers

Beekeepers can get to know their colonies better without continuously disturbing the industrious insects. Using a Raspberry Pi and various sensors, two hobby beekeepers monitor the temperature and humidity of their hives, with plans to monitor their weight. *By Lars Jurzik and Heike Jurzik*

The winters in Germany's Sauerland region are long, but above all, changeable. Sometimes it snows in October, and sometimes you experience a spring-like 15°C (59°F) over Christmas. The beekeepers in the area are prepared for this and prepare their bees for the winter in the early autumn after the last honey harvest by feeding several liters of sugar solution into the hives, depending on the size of the bee colony. When the outside temperatures drop, the response is prompt: Lock up, install mouse gratings in front of the entrance, and meet again in April.

Because beekeepers do not normally open the hives during the winter months, they cannot know if the population is thriving. Inspired by the Hiveeyes Project [1] and the Open Hive Monitoring System [2], we planned our own monitoring solution for our colonies. As hobby beekeepers, we first want to observe the temperature (see the "Test Setup" box) but, later, also connect our own hive scale.

Connecting the DHT11 Sensor

The DHT11 digital temperature and humidity sensor is available for a few dollars. It supplies the temperature in degrees Celsius and the relative humidity as a percentage. In the first trial, we connected the sensor directly to the RPi2B.

The *Python_DHT* sensor library [3] helps with the readout. After installing the packages *build-essential* and *python3-dev*, we checked out the sensor library from the GitHub repository and installed it on the computer:

```
$ git clone https://github.com/➥
  jugend-programmiert/Python_DHT
$ cd Python_DHT
$ sudo python3 setup.py install
```

To use the library in your own Python scripts, you use `import`. For example, Listing 1 reads sensor data and outputs it to the console. A test run on the console shows that the sensor and RPi2B are working together:

```
$ python3 dht11_simple.py
dht temperature=17.0,humidity=49.0
```

After saving the script to `/usr/local/bin`, the database and the collector were the next steps.

Setting Up the Database

The InfluxData package source [4] contributes both the InfluxDB database [5] and the Telegraf collector [6], which are added to `/etc/apt/sources.list.d/influxdb.list`; then, we added the repository's GnuPG key:

```
$ curl -sL ➥
  https://repos.influxdata.com/➥
  influxdb.key | sudo apt-key add -
```

After updating the package list (`apt update`) and importing the *influxdb* and *telegraf* packages, we configured the database to start automatically at boot

Test Setup

A Raspberry Pi 2 Model B (RPi2B) with Raspbian Stretch (based on Debian 9) was used, to which we connected a temperature and humidity sensor (DHT11). We also experimented with a DS18B20 temperature sensor and an active speaker connected to the RPi2B via a 40-pin GPIO extension board and a breadboard.

Listing 1: dht11_simple.py

```
01 import Python_DHT
02
03 sensor = Python_DHT.DHT11
04 pin = 4
05 humidity, temperature = Python_DHT.read_retry(sensor, pin)
06 print("dht temperature="+str(temperature)+",humidity="+str(humidity))
```

Listing 2: Querying the Database

```
$ influx
Connected to http://localhost:8086 version 1.5.1
InfluxDB shell version: 1.5.1
> show databases
name: databases
name
----
telegraf
_internal
> use telegraf
Using database telegraf
> show series
key
---
cpu,cpu=cpu-total,host=raspberrypi
[...]
dht,host=raspberrypi
[...]
system,host=raspberrypi
> select * from dht;
name: dht
time                host        humidity temperature
----------------------------
1522242851000000000 raspberrypi 54       19
1522242911000000000 raspberrypi 54       19
1522242971000000000 raspberrypi 53       19
1522243031000000000 raspberrypi 53       19
[...]
```

time, initialized the server, and started the command-line client:

```
$ sudo systemctl enable influxdb
$ sudo systemctl start influxdb
$ sudo influx
Connected to http://localhost:8086 ➥
  version 1.5.1
InfluxDB shell version: 1.5.1
```

When the client started, we created a new `admin` account and a new database named `telegraf`:

```
> CREATE USER admin ➥
  WITH PASSWORD '****' ➥
  WITH ALL PRIVILEGES
> CREATE DATABASE telegraf
> exit
```

In the InfluxDB configuration file `/etc/influxdb/influxdb.conf`, you need to enable the web server under `[http]`:

```
[http]
  enabled = true
  bind-address = ":8086"
```

After restarting the service by typing

```
systemctl restart influxdb
```

the Telegraf configuration continues.

Well Acquired

The `telegraf` account must be a member of the `gpio` group for the collector to read values from the GPIO pin:

```
usermod -a -G gpio telegraf
```

The `/etc/telegraf/telegraf.conf` file contains the database information in the `OUTPUT PLUGINS` section:

```
[[outputs.influxdb]]
  timeout = "5s"
  username = "admin"
  password = "****"
```

The `INPUT PLUGINS` area also has a space for your Python script, which will run once a minute:

```
[[inputs.exec]]
    commands = ➥
     ["python3 /usr/local/bin/➥
       dht11_simple.py"]
    interval ="60s"
    data_format = "influx"
```

If you want to check whether the communication between Telegraf and InfluxDB is working, you can launch the `influx` client. Listing 2 shows how we query the `telegraf` database on the test system.

Illustrated

Grafana [7] visualizes the acquired data. The software supports numerous data sources, including InfluxDB databases. Because the official Raspbian repositories contain a relatively old Grafana version, we used a current package from GitHub [8].

The Grafana service also needs to be configured to launch automatically after boot with `systemctl enable`; the

```
systemctl start grafana-server
```

command calls the service, which users can reach from web interface port 3000 (username and password are *admin*). First, we used *Add data source* to add the `telegraf` database; then, we set up the dashboard to visualize the metrics (Figure 1).

We moved the RPi2B, at first provisionally protected with plastic bags and tape, into the beehive, which is located next to the house wall and can therefore use the existing wireless network (Figure 2).

To make room for the RPi2B and sensor, we took one slat out of the top floor of the

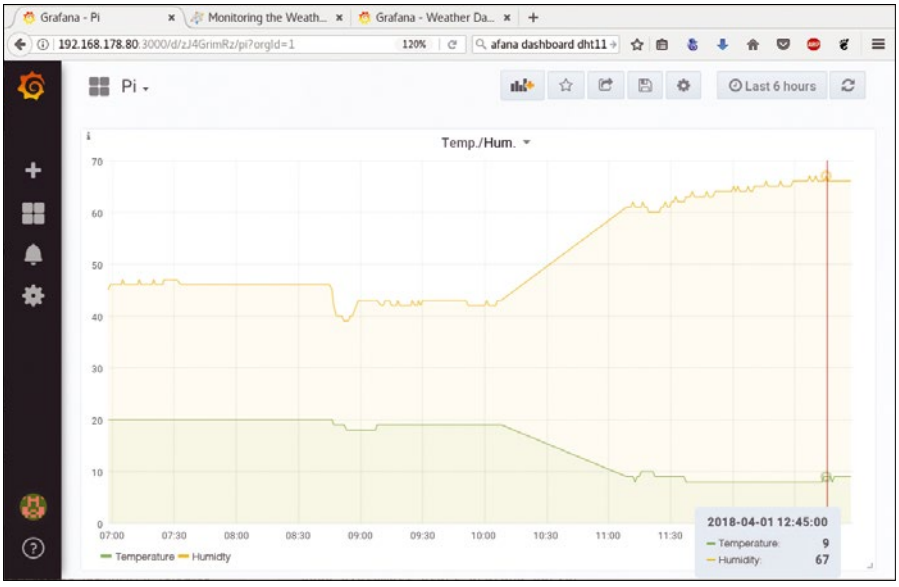


Figure 1: The sensor measures temperatures between 8°C and 9°C (ca. 46°F and 48°F) at the upper edge of the lower hive frame shortly after 10am, when the RPi2B was hung in the hive.

Figure 2: The RPi2B is still in a temporary housing. A waterproof housing and a solar module for wireless power are on the shopping list.

two-frame hive. The DHT11 currently measures the temperature at the upper edge of the lower frame. At an outside temperature of 1°C (ca. 34°F), the average Easter weekend temperature was 8°C.

Close to the Bees

The temperatures are higher where the queen lives and where the bees breed (about 35°C, or 95°F). Other beekeepers have three or four sensors in the hives and measure the temperature at the entrance hole, under the lid, and in the lowest frame. You can check out the sensor results of the Hiveeyes project online [9].

With the information from a SunFounder tutorial [10], we experimented with a DS18B20 temperature sensor and a speaker module. To communicate through the GPIO extension board, we installed the Wiring Pi library [11], which provides the `gpio` tool (Figure 3).

pi@raspberrypi: ~

```
pi@raspberrypi:~ $ gpio readall
 +-----+-----+---------+------+---+---Pi 2---+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 |     |     |    3.3v |      |   |  1 || 2  |   |      | 5v      |     |     |
 |   2 |   8 |   SDA.1 |   IN | 1 |  3 || 4  |   |      | 5v      |     |     |
 |   3 |   9 |   SCL.1 |   IN | 1 |  5 || 6  |   |      | 0v      |     |     |
 |   4 |   7 | GPIO. 7 |   IN | 1 |  7 || 8  | 1 | ALT0 | TxD     | 15  | 14  |
 |     |     |      0v |      |   |  9 || 10 | 1 | ALT0 | RxD     | 16  | 15  |
 |  17 |   0 | GPIO. 0 |   IN | 0 | 11 || 12 | 0 | IN   | GPIO. 1 | 1   | 18  |
 |  27 |   2 | GPIO. 2 |   IN | 0 | 13 || 14 |   |      | 0v      |     |     |
 |  22 |   3 | GPIO. 3 |   IN | 0 | 15 || 16 | 0 | IN   | GPIO. 4 | 4   | 23  |
 |     |     |    3.3v |      |   | 17 || 18 | 0 | IN   | GPIO. 5 | 5   | 24  |
 |  10 |  12 |    MOSI |   IN | 0 | 19 || 20 |   |      | 0v      |     |     |
 |   9 |  13 |    MISO |   IN | 0 | 21 || 22 | 0 | IN   | GPIO. 6 | 6   | 25  |
 |  11 |  14 |    SCLK |   IN | 0 | 23 || 24 | 1 | IN   | CE0     | 10  | 8   |
 |     |     |      0v |      |   | 25 || 26 | 1 | IN   | CE1     | 11  | 7   |
 |   0 |  30 |   SDA.0 |   IN | 1 | 27 || 28 | 1 | IN   | SCL.0   | 31  | 1   |
 |   5 |  21 | GPIO.21 |   IN | 1 | 29 || 30 |   |      | 0v      |     |     |
 |   6 |  22 | GPIO.22 |   IN | 1 | 31 || 32 | 0 | IN   | GPIO.26 | 26  | 12  |
 |  13 |  23 | GPIO.23 |   IN | 0 | 33 || 34 |   |      | 0v      |     |     |
 |  19 |  24 | GPIO.24 |   IN | 0 | 35 || 36 | 0 | IN   | GPIO.27 | 27  | 16  |
 |  26 |  25 | GPIO.25 |   IN | 0 | 37 || 38 | 0 | IN   | GPIO.28 | 28  | 20  |
 |     |     |      0v |      |   | 39 || 40 | 0 | IN   | GPIO.29 | 29  | 21  |
 +-----+-----+---------+------+---+----++----+---+------+---------+-----+-----+
 | BCM | wPi |   Name  | Mode | V | Physical | V | Mode | Name    | wPi | BCM |
 +-----+-----+---------+------+---+---Pi 2---+---+------+---------+-----+-----+
pi@raspberrypi:~ $
```

Figure 3: The GPIO layouts.

The idea was to create a measuring device that emits an audible signal when a certain temperature is exceeded. Such a device could be interesting for transporting colonies of bees. If longer distances are planned, a transport grid on top of the hive ensures a sufficient air supply. For shorter distances, however, many beekeepers simply secure the hive with a strap.

When it gets too warm, the bees start flapping their wings to cool down, and when the hive is closed, everything heats up even more – the insects can produce such high temperatures that the combs and honey melt and the bees perish. A signal tone warns the beekeeper during transport and prevents the colonies from buzzing themselves to death while cooling the beehive.

Do Not Miscalculate!

A hive scale is planned for the summer to monitor the weight of the beehives. The beekeepers association in Nettetal, Germany, implemented such a project with an Arduino [12]; the data is presented by the Hiveeyes project. A scale is not only interesting in the summer to observe honey production, it can also be used in winter to better assess whether sufficient food is available. If it remains very cold for a long time, the bees will need more food – you can then see in time whether or not nutrition is becoming scarce. ■■■

Info

[1] The Hiveeyes Project: *https://www.hiveeyes.org*

[2] Open Hive: *http://open-hive.org*

[3] Python_DHT sensor library: *https://github.com/coding-world/Python_DHT* (README in German)

[4] InfluxDB and Telegraf repository: *https://repos.influxdata.com/debian*

[5] InfluxDB: *https://www.influxdata.com/time-series-platform/influxdb*

[6] Telegraf: *https://www.influxdata.com/time-series-platform/telegraf*

[7] Grafana: *https://grafana.com*

[8] Grafana on GitHub: *https://github.com/grafana/grafana*

[9] Hiveeyes dashboards: *https://swarm.hiveeyes.org/grafana/dashboards*

[10] SunFounder tutorials: *https://www.sunfounder.com/learn*

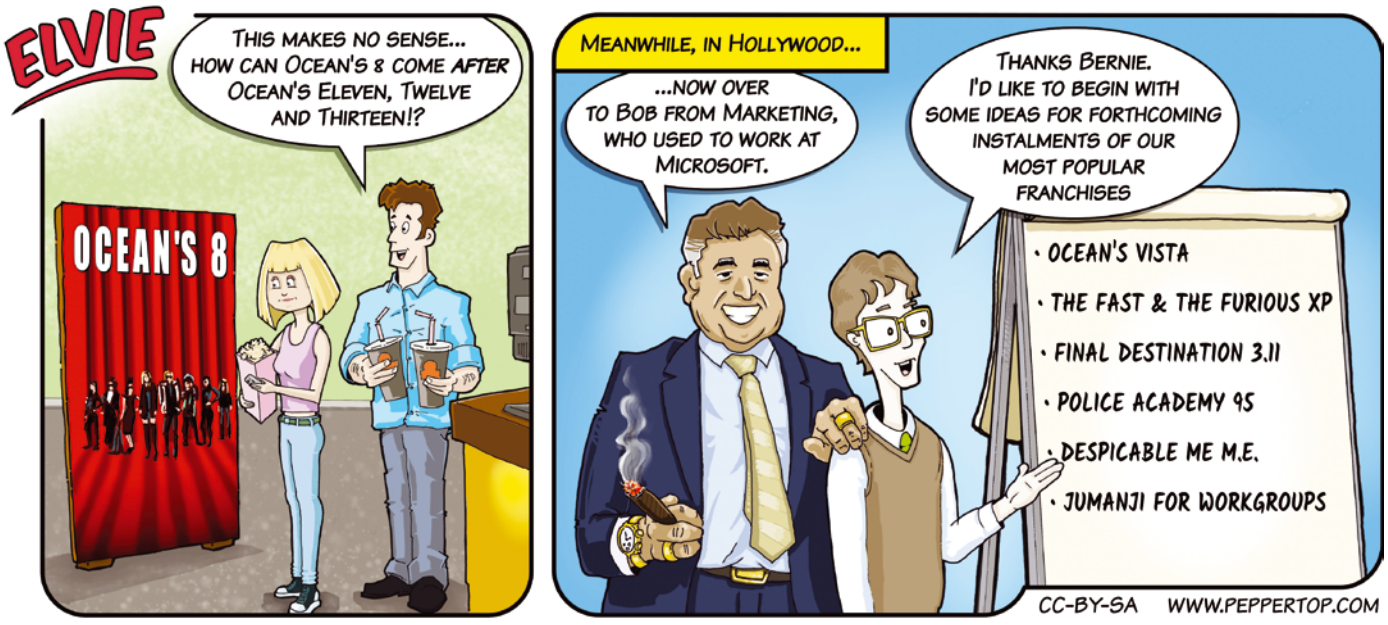
[11] Wiring Pi: *http://wiringpi.com*

[12] Arduino bee scale: *https://www.imker-nettetal.de/esp8266-beescale-erste-eindruecke* (in German)

Some science experiments require massive amounts of computer analysis. Does that mean you need a massive computer just to crunch the numbers? Although many scientists would love to have their own supercomputer, most lab budgets recommend a different approach. The Berkeley Open Infrastructure for Network Computing (BOINC) is a framework that lets you break a problem into small bits and distribute the pieces to computers running all over the world. This month we show you how to use BOINC to render proteins and help with the search for extraterrestrial life. Also in this issue, you'll learn how to look for missing files with DocFetcher and manage bloat with Ncdu.

Image © Olexandr Moroz, 123RF.com

LINUXVOICE



A Webzine for High-Performance Computing Specialists

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

ADMIN Network & Security

admin-magazine.com/hpc

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®.

Adopting FOSS firmware can keep your hardware out of the landfill. BY JON "MADDOG" HALL

Designing for reuse

Recently, an article by Bruce Schneier, CTO of IBM Resilient, came across my desk discussing some of the issues around home broadband routers. Apparently, on May 25th, the FBI asked people to "reboot their routers" as various pieces of malware were running in these routers and causing havoc with the Internet. The malware was very sophisticated, to the point where it had (more or less) a software backplane and could install new "plugins" to create even more mischief. Mr. Schneier stated that the malware was probably not the creation of your classic "basement cracker" but was instead a product of an intense program by a government.

Unfortunately, rebooting the router did not really fix the problem, as the malware was still infecting the lower levels of the router firmware, so at a minimum, the user would have to "reset to factory settings" (destroying the sometimes complex set of configuration and passwords) or (more likely) install new firmware, which is completely outside the abilities of most residential users of broadband routers, even assuming that the makers of the router are still updating the firmware or are even in business.

Mr. Schneier therefore recommended that, if the router does not have firmware updates, the best course of action is to throw away the old router and buy a new one, which is what several of my less computer-savvy friends did.

Part of this issue is that these problems have been known for a long time. The reason that the FBI wanted people to reboot their router was that a rebooted router would call to the mother ship of the crackers and try to download malware again. The FBI (who had intercepted the mother ship) wanted to see how many routers were affected. Apparently there were a lot. The list of known routers that might be affected can be found online [1].

There were several other security-related articles that flashed across my computer screen over the past two weeks, one of which had to do with some of the Internet of Things (IoT) devices we are now deploying in our homes. A technique known as *DNS Rebinding*, which has been known for at least a decade, can affect modern devices such as Roku streaming devices, Sonos wireless speakers, smart home thermostats, Google Home, and Chromecast (the last two I have in my home). These issues keep surfacing because people bring out new devices and, in the interest of making them as "self-configuring" and easy to set up as possible, overlook the techniques used to allow these devices to be taken over by crackers.

Many of these devices do not use strong encryption and authentication to make sure that the software talking to them should have the right to interact with them. People who program them assume that, since they are installed behind secure broadband router gateways (cough), the security on these streaming devices can be relaxed.

Now imagine that there is not only ONE of these devices in your home, but dozens, or hundreds. I have twelve Google Minis, a Google Home, and a Google Max in my house. Yes, I know that "Google can listen to me." I have much fewer issues with Google listening to me than I have with crackers trying to steal my identity or credit card information. Fortunately, my router is up to date with its firmware, and I have configured it well.

Which brings about the issues of FOSS firmware on routers and IoT devices.

Sometimes, I think I am sounding like a broken record (people who have never heard a vinyl record may not understand, understand, understand…) when I say that not buying hardware that can run FOSS is just asking to help build the junkyard of old and forgotten electronics in the future. Eventually the company or solution provider who last built the software for the device will either go out of business or lose interest in updating the software.

Caninos Loucos [2], the LSITEC project in which OptDyn is collaborating, has a long-term plan of making 100% open hardware and software, even to the point of allowing others to make their own CPU (following an open architecture, of course) so they can make sure there is no malware in the system.

Caninos Loucos is designing a sensor computer the size of a dime, with the goal of complete openness. Their other "dog" computers, at this point the Labrador and Bankhar (whose design as the Subutai Broadband Router was contributed to the project) are also as open as modern components allow. The design focus is to document the hardware interfaces so the FOSS communities can support them forever, or at least make the hardware useful even if the upper levels of functionality change.

Unless we want an electronic mountain higher than Everest, we need to design for repurpose and reuse. ■■■

Info

[1] Affected routers: *https://www.symantec.com/blogs/threat-intelligence/vpnfilter-iot-malware*

[2] Caninos Loucos: *caninosloucos.org*

Weighing in with Ncdu

Ncdu adds some GUI-like features to the classic du command. BY FERDINAND THOMMES

The `du` (for "disk usage") utility is a useful tool for estimating file size and cleaning up a hard drive. `du` summarizes the disk usage for each file in the directory and even looks down into subdirectories.

`du` worked well in the old days of printed output, but in today's interactive environments, it has some limitations. Like many old-school command-line tools, `du` prints the output on the screen by default, and the terminal screen is nowhere near large enough to see all the files in a large directory system. Of course, you could print to a file and examine the file with a text editor, or you could filter and refine the `du` results using standard command-line piping and sorting techniques. But sometimes what you really want to do is move around in the output as if perusing a printed page, progressing interactively through subdirectories as you would with a GUI-based file manager.

The Ncdu utility [1] is a souped-up version of du that has some GUI-like features. Ncdu uses the ncurses API [2], which is a common tool for building GUI features into command-line programs. (See the box entitled "Why Work in the Terminal?")

Ncdu, which has been under development since 2007, is written in C and works on all POSIX-compliant systems. Packages are available for many Linux distributions, as well as for BSD, Solaris, Cygwin, and Mac OS.

Ncdu comes preinstalled on many Linux distributions, and if you can't find it on your system now, you can easily install it using your local package manager. The Ncdu utility is only a few kilobytes in size, and it is released under an MIT license. The easiest way to control Ncdu is to use the arrow keys, the Enter key, and a few letters.

Easy to Use

To start Ncdu, type `ncdu`. If you only want to examine hard disk usage, you can simply start the application as a user. If you want to delete data outside of your home directory, you will need root privileges. The `ncdu` command supports a number of command-line options (see the box entitled "Command Line Options.")

Ncdu sorts directories and files in descending order by default (Figure 2). The letter *S* reverses the sort order and shows the smallest folders and files at the start. Pressing *G* lets you toggle through the different modes for displaying the relative size. We found figures in percent format the most meaningful (Figure 3). All told, the application offers a handful of options and a good dozen keyboard shortcuts to manipulate the output. The most common operating options are listed in Table 1.

The four arrow keys are all you actually need to navigate Ncdu. Use up and down arrow keys to move through the directory tree. The right arrow key takes you down into the selected directory, and the left arrow key back up again in the directory tree. Instead of the right arrow key, the Enter key also works in both directions. Alternatively, you can move up and down the directory tree with the *K* and *J* keys. Alternatively,

Why Work in the Terminal?

Some users prefer to work in a terminal window all the time – if you're good at it, you can work very quickly and efficiently without ever reaching for a mouse or touch screen. But even if you're accustomed to a desktop environment, you might have the occasional need for a command-line disk usage tool.

Desktop environments include several graphical tools for examining file size and disk usage, including QDirStat [3], Filelight [4], and Baobab [5]. But if the disk is completely full due to an error, or if another problem somewhere in the system caused a logfile to expand and lock up the disk with error messages, the graphical interface might not be accessible, and you need to use a text-based command to search out and delete bloated files so you can restart the system.

A tool like Ncdu is also useful for remote terminal sessions with SSH, in cases where a remote desktop session is impractical or prohibited.

you can open a directory with the *L*; the < key closes it again.

Customizing the Default Output

The default output format for Ncdu shows the size of the largest directory or file in the first row on the left. On the right is the file or directory name, where a forward slash marks a directory. In the middle column, the relative size of the element is shown with a graph of hashtags. Toggle the size display by pressing the *G* key. It shows additional information, or only percentages, or no information at all on the relative size.

If you want to scan the whole filesystem, enter `ncdu /`. To exclude other filesystems from the search, add `-x`.

Delete with Care

Once you have found the element you are looking for, and assuming you want to remove disk

Command-Line Options

The `ncdu` command supports some common command-line options, such as `-h` (Figure 1) for help and `-v` for displaying the version number. If you start Ncdu in quiet mode with the `-q` option, the screen will update less frequently, which will save bandwidth if you are working through an SSH connection.

The `-x` option tells Ncdu not to analyze all directories in the path but only those with the same filesystem as the source directory. This means that if the entire system is scanned, other mounted filesystems are not included. `-X` lets you exclude files and directories from the scan.

```
$ ncdu -h
ncdu <options> <directory>

  -h                        This help message
  -q                        Quiet mode, refresh interval 2 seconds
  -v                        Print version
  -x                        Same filesystem
  -r                        Read only
  -o FILE                   Export scanned directory to FILE
  -f FILE                   Import scanned directory from FILE
  -0,-1,-2                  UI to use when scanning (0=none,2=full ncurses)
  --si                      Use base 10 (SI) prefixes instead of base 2
  --exclude PATTERN         Exclude files that match PATTERN
  -X, --exclude-from FILE   Exclude files that match any pattern in FILE
  --exclude-caches          Exclude directories containing CACHEDIR.TAG
  --confirm-quit            Confirm quitting ncdu
```

Figure 1: Use the `-h` option to display an incomplete summary of command-line options. View the manpage by entering `man ncdu` to see a complete list.

```
ncdu 1.12 ~ Use the arrow keys to navigate, press ? for help
--- /home/devil ------------------------------------------------------------
  188.9 GiB [##########] /Downloads
   28.1 GiB [#         ] /Pictures
   19.6 GiB [#         ] /ownCloud
    4.9 GiB [          ] /.thunderbird.old
    4.4 GiB [          ] /.cache
    3.2 GiB [          ] /.config
    3.1 GiB [          ] /.icedove
    1.7 GiB [          ] /Calibre Bibliothek
    1.6 GiB [          ] /.thunderbird.old2
.   1.5 GiB [          ] /.local
    1.2 GiB [          ] /2015
  901.0 MiB [          ] /Workshots
  710.9 MiB [          ] /work
  475.5 MiB [          ] /PDF
  376.6 MiB [          ]  .xsession-errors
  249.0 MiB [          ] /Notebooks
  219.5 MiB [          ] /devil-official
  134.2 MiB [          ] /.var
  115.6 MiB [          ] /Rezepte
  113.9 MiB [          ] /.mozilla
   99.6 MiB [          ] /.mediathek3
   99.0 MiB [          ] /thunderbird
   79.6 MiB [          ] /siduction
   67 7 MiB [          ] /USA
   58.7 MiB [          ] /sdcard0
   48.9 MiB [          ] /.opera
   47.5 MiB [          ]  thunderbird-52.5.0.tar.bz2
   46.7 MiB [          ] /calamares
   34.4 MiB [          ] /.kde
 Total disk usage: 262.2 GiB  Apparent size: 262.1 GiB  Items: 404067
```

Figure 2: In the standard output, Ncdu shows the relative size of elements with hashtags.

Table 1: Ncdu Command Reference

Navigation	
Key	**Function**
Up/K	Up in the directory tree
Down/J	Down in the directory tree
Right/L/Enter	Open directory
Left/</H/Enter	Close directory
Manipulation	
N	Sort by file name
S	Sort by file size
A	Toggle between effective size and occupied space for a file
D	Delete the selected file from the directory
T	Sort directories before files
G	Toggle between size display by graph, percent, both, or none
E	Don't show hidden files
I	Information about the selected element
R	Recalculate selected element
Q	Quit Ncdu

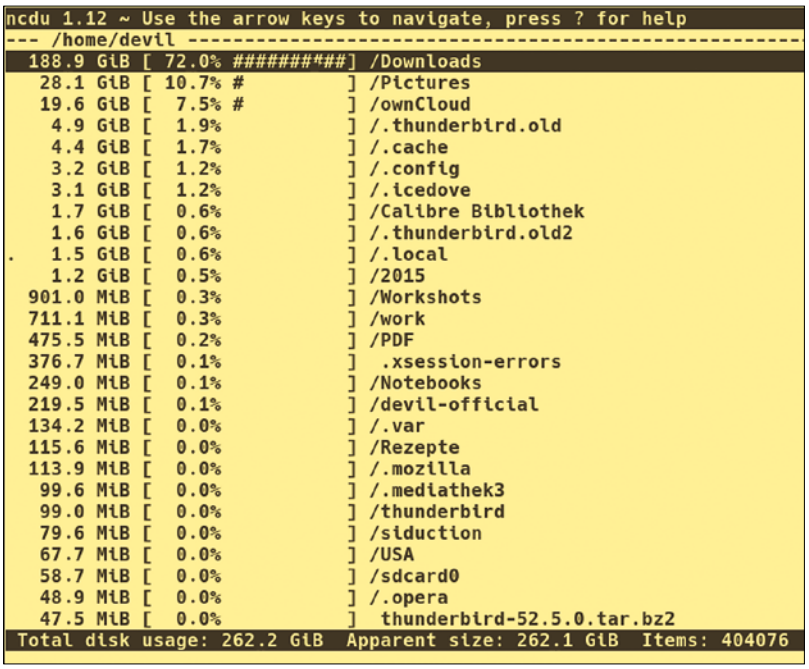


Figure 3: The *G* key lets you toggle to a mode where the relative sizes are expressed in percent format.

hogs that you no longer need, press the *D* key to delete the selected element. Be careful, because the process is irreversible; Ncdu uses the `rm` command in the background. To protect yourself from accidental deletion, start Ncdu with the `-r` option, which disables the delete feature (Figure 4).

Ncdu to Go

Ncdu can also redirect the output to a file for later analysis, as follows:

```
ncdu /home/Downloads -o Downloads-txtcommand
```

The following command:

```
ncdu -1xo- /home/Downloads | gzip >Downloads.gz
```

dumps the output into an archive format. To view the archive later on, enter:

```
zcat Downloads.gz | ncdu -f-
```

To view the unpacked text version, use:

```
ncdu -f Downloads.txt
```

Conclusions

Ncdu is a useful tool for analyzing the available disk space and locating and removing the biggest space-wasting files. In addition to the other benefits of working at the command line, Ncdu is faster than its GUI-enabled colleagues. You can also include Ncdu in scripts or run it as a cronjob. ■■■

Info

[1] Ncdu: *https://dev.yorhel.nl/ncdu*

[2] ncurses: *https://en.wikipedia.org/wiki/Ncurses*

[3] QDirStat: *https://github.com/shundhammer/qdirstat/releases*

[4] Filelight: *https://userbase.kde.org/Filelight*

[5] Baobab: *https://wiki.gnome.org/action/show/Apps/DiskUsageAnalyzer?action=show&redirect=Apps%2FBaobab*

Figure 4: To prevent accidental deletion, start Ncdu with the `-r` option for read only.

COMPLETE YOUR LIBRARY

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

ORDER YOURS TODAY!
shop.linuxnewmedia.com

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

Cycles for Science

Want to participate in the scientific revolution? BOINC lets you lend your spare computer cycles to data analysis efforts for NASA and other science institutions.

BY MATS TAGE AXELSSON

Scientists collect massive amounts of data. In fact, the biggest challenge of science is sifting through all the data to come to correct and fascinating conclusions. This is where you come in: If you want to help with the march of science, you could give your life to NASA or a large university, or you could just contribute some cycles from your home computer. One of the easiest ways to contribute to science is to let your computer crunch the numbers for ongoing scientific experiments.

It all started with the search for extraterrestrial intelligence (SETI). SETI scientists built a radio telescope to scan the skies for radio signals that could come from civilizations like ours. The big problem quickly became obvious; analyzing the data takes immense amounts of computing power. To address this problem, the team adapted the software they use to analyze the signal to run on a PC. Then they came up with a brilliant idea: make small work packages that anyone can compute in any order. Users can download the packages and analyze the data off-line. This client was then shared on their website, the best part: they designed it so users could use the software as a screen saver.

The popularity of the project exceeded their expectations, and the team quickly realized that the system could work for other projects also. However, other projects such as medical research and astronomy needed other mathematics. The solution was Berkeley Open Infrastructure for Network Computing, BOINC [1].

BOINC is a framework for crowd-sourced scientific analysis projects. Volunteers can download software and then choose which project they want to support. Institutions can even write their own code and add it to the list of supported projects.

The BOINC Client

Installing the BOINC client on most distributions requires only your favorite package manager. You'll need to install two packages. The first

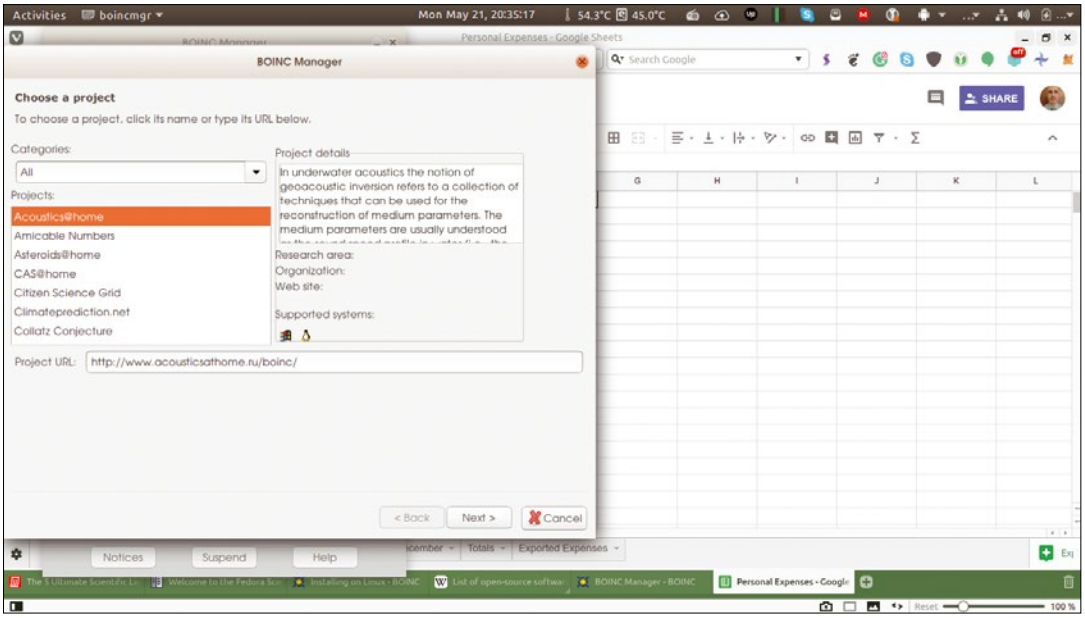


Figure 1: Choose your projects using the graphical BOINC manager.

The Author

Mats Tage Axelsson is still struggling to make the most of his Linux computer. All upgrades lead to more applications instead of better performance.

package is the client that does all the work. The other package is the GUI where you define which projects you wish to support. Without the GUI, you will have to configure your projects at the command line.

If you prefer to compile the BOINC client yourself, all the source code is available on GitHub [2]. Cloning the files to your machine is one way to get started; the git repository has the source for both the client, the manager, and the API. Once the client is installed, you can start crunching data, host a project, or even create your own project for others to help you. If you create your own project, you can save work by creating a VirtualBox image to run it in.

To install BOINC in Ubuntu, use the `apt` command.

```
sudo apt install boinc
```

The BOINC package is a meta package that installs the client and the GUI manager. It does not install boinctui, which is curses based and can be useful for servers where only ssh connections available. Other packages you need to consider are the ones that let BOINC access the GPU. The packages will vary depending on your hardware; for Nvidia, you can use CUDA. BOINC has its own package for utilizing these libraries; boinc-client-nvidia-cuda is a meta package that takes care of installing everything necessary.

If you wish to use BOINC to start your own project, it is a good idea to use a virtual machine, because you'll need many special packages, and you might not want to make all these changes to your production system.

You can run BOINC with the command-line tool called boinccmd, which is useful if you want to create your own script. If, however, you just want to add a project or two, use boincmgr. The boincmgr manager tool is a GUI that lets you choose to start, suspend, and stop projects (Figure 1).

BOINC Projects

Once BOINC is up and running, you can choose to support many projects with one client. SETI is fascinating, but what about helping with other research? BOINC supports a number of medical research projects. Denis@home simulates electrical heart activity. Rosetta@home simulates protein – the results are useful for research on HIV, malaria, cancer, and dementia. FERMI lab has also weighed in to get help analyzing data from the Large Hadron Collider. Other physics projects are also available. One noteable example is Einstein@home, which analyses LIGO data to search for gravitational waves. If you want to explore our galaxy, join Milkyway@home. To go further, join Universe@home to help create a simulation of the entire universe.

Several projects even let you collect data yourself, which requires the purchase of hardware. The hardware is usually cheap though. For instance, the radioactive@home project sells their sensor for € 25 with a rebate for large orders.

The Folding@home project comes as two different files: a client and a viewer. The client does the number crunching, and the viewer shows off a graph of a protein (Figure 2). To fetch the software, pick up the deb or rpm file at the Folding@home website [3]. Install with

```
sudo dpkg -i fahclient_7.5.1_amd64.deb
```

or

```
fahviewer-7.5.1-1.x86_64.rpm
```

Figure 2: Folding@home viewer showing a rendering of a protein.

Space Race

Many research projects have not found a good way of classifying their data. The Galaxy Zoo project has invited volunteers to classify pictures of different galaxies. Programs for these classifications has proven notoriously difficult to write, so in this case, you will be donating your personal brain power.

NASA also has a software package for analyzing images from their missions; you can download the package from the US Geological Survey website [4].

The package is for Ubuntu 14.04, which means your libraries are probably too new, which could cause problems with a more current release.

Use a virtual machine to get started; *osboxes.org* has 14.04 appliances. They also support Fedora 25 and Debian 8; the source is available on GitHub [5].

The complete data set is 130GB; if you think that is a lot, you need to choose specific missions to work with. If you use the command line, the server expects the `rsync` command. If you want to follow an active mission, you can just rerun rsync on a regular basis, and the command will download only the difference.

The following command downloads the data from the Cassini mission.

```
rsync -azv --delete --partial ➥
isisdist.astrogeology.usgs.gov::➥
isis3data/data/cassini data/
```

The Cassini data weighs in at around 14GB on disk. The data comes as pictures and other files that define where the spacecraft was at the time the picture was taken. To get accurate results for things like the height of a mountain, you need the position and angle of the camera.

With the data size in mind, consider using a SPICE server. Your software can use the SPICE protocol to pick up data on demand, instead of downloading it all in advance. Add `-exclude='kernels'` to your `rsync` command and run the following command:

```
spiceinit -GUI
```

You will now use `spiceinit` to create a specialized data file for your particular needs (Figure 3). If Cassini is not enough, you also have 9 missions of the European Space Agency.

Conclusion

BOINC won't win you a Nobel prize, but it does offer a means to participate in real research projects. The best part about setting up a project in BOINC is that, along the way, you learn a little about science and gain some insight into how scientists search for answers. ■■■

Info

[1] BOINC: *https://boinc.berkeley.edu/*

[2] BOINC on GitHub: *https://github.com/BOINC/boinc*

[3] Folding at Home: *https://foldingathome.org/start-folding/*

[4] ISIS Planetary Image Processing: *https://isis.astrogeology.usgs.gov/documents/InstallGuide/assets/isisInstall.sh*

[5] ISIS Source Code: *https://github.com/USGS-Astrogeology/ISIS3*

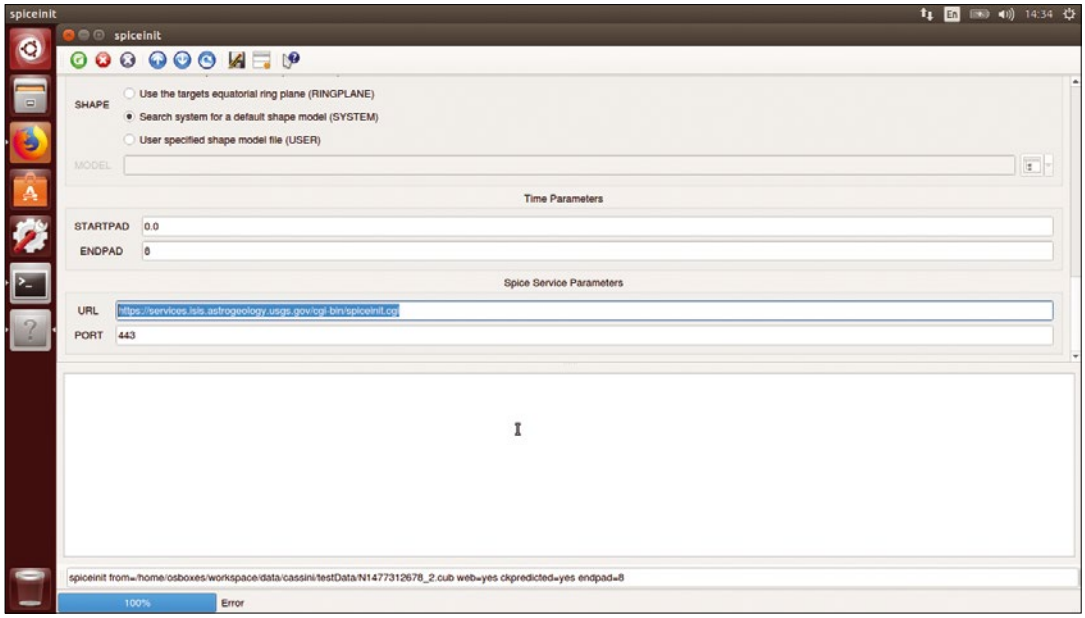


Figure 3: Using spiceinit to create a data set.

Bloodhound

DocFetcher is a practical local search tool that is easy to configure and use – even for large data collections. BY ERIK BÄRWALDT

Modern operating systems take up several Gigabytes of space just for the many application programs, and they sometimes contain up to several hundred thousand individual files. If you add your extensive music or photo collection, you can quickly lose track.

Modern desktop environments offer indexing and search applications for existing data, and the Linux environment includes several special search programs. However, many of these programs are not very intuitive, and some even expect you to install a database as a backend. In addition, many of the tools often do not support full-text searches. If you are looking for a lean, practical, and powerful search tool for your workstation, DocFetcher is a very interesting alternative.

You can download the Java application from the project page, where you will also find installation instructions [1]. As a prerequisite, you need a reasonably up-to-date Java runtime environment; DocFetcher harmonizes perfectly with the current OpenJDK environments, which you can usually install directly from your distribution's software repositories.

Listing 1: Unzipping DocFetcher

```
$ unzip docfetcher-1.1.19-portable.zip
$ sudo mv DocFetcher-1.1.19/ /usr/local/bin/docfetcher
```

Listing 2: Creating a Menu Entry

```
Version=1.0
Name=DocFetcher
GenericName=Document Index and Search
X-GNOME-FullName=DocFetcher Document Index and Search
Comment=Index and Search your computer
Type=Application
Categories=System;Utility;FileTools;Java;
Exec=/usr/local/bin/docfetcher/DocFetcher-GTK3.sh
Terminal=false
StartupNotify=true
Icon=/usr/local/bin/docfetcher/img/docfetcher128.png
```

Unpack the downloaded ZIP archive with the DocFetcher files using a tool like Ark, File Roller, or Xarchiver. You can then move the subdirectory you created to a directory of your choice. To start the program from a desktop menu, however, you need to manually create a menu entry (see the box entitled "Installation").

Start Your Engines

When you first launch DocFetcher, some systems start with a dialog where you can change the keyboard shortcut from the default ([Ctrl]+[F8]). If the shortcut is already mapped, a message asks you to confirm by pressing *OK*. The program window, which is divided into five panes, then appears. In the top-left corner, you will find an input field for the minimum and maximum file size that DocFetcher should consider for the search.

Select the file types you want DocFetcher to find from a dropdown list; the program enables all sup-

Installation

Many Linux distributions do not include DocFetcher in their package sources. Ubuntu, for example, does not yet include a package for DocFetcher. It is thus often necessary to install DocFetcher manually.

Listing 1 shows how to unpack the ZIP archive downloaded from the project page into the `/usr/local/bin/` directory. In Listing 2, you will find the content for `/usr/share/applications/docfetcher.desktop` to help you create a matching entry in the *Start* menu of the desktop environment.

Adjust the version number in the commands if necessary. If you prefer a location other than `/usr/local/bin/docfetcher/`, remember to change the paths appropriately. If you are still using a system without GTK3 libraries, you also need to swap `DocFetcher-GTK3.sh` for `DocFetcher-GTK2.sh` in the `Exec` line.

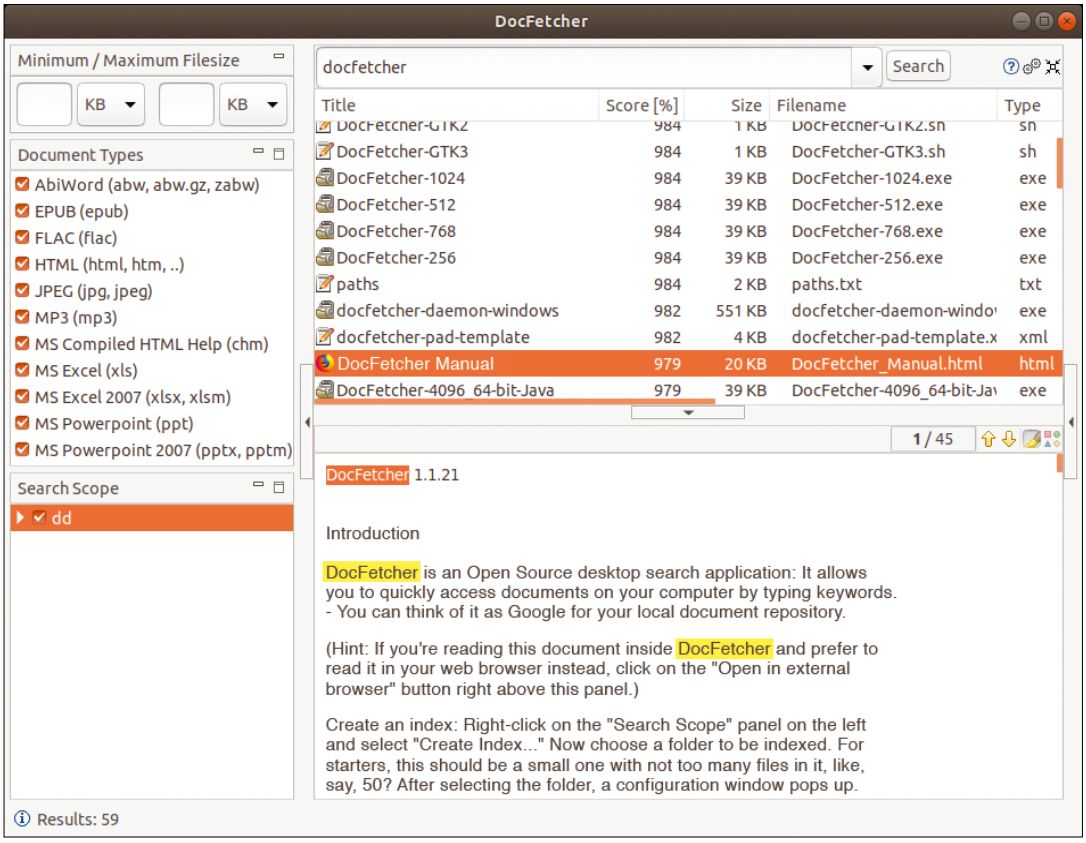


Figure 1: The DocFetcher interface shows all the important information.

ported formats by default. Below is the search area, and top-right is an input line for the search terms. Below this area, the software lists the results with information on match relevance and file size; an area in the bottom right displays the contents of the selected file (Figure 1).

DocFetcher needs to index the contents of the mounted storage media in order to search reliably and quickly even in large data sets. You can trigger this indexing from the *Create Index From* dialog, which you can access by right-clicking in the search area in the bottom-left of the main window. Then select either a folder or an archive file. In Microsoft environments, DocFetcher supports indexing of PST files containing messages, contacts, tasks, or appointments.

To limit the size of files that the program should consider, enter the minimum and maximum values in the boxes in the upper-left corner. The process of indexing the data collection, which relies on Apache Lucene [2], takes some time during the first run, but this step will significantly speed up searching in these folders (Figure 2).

After indexing is complete, you will find the indexed directories and archives in the *Search Scope* pane. Enter the desired search terms in the search box. After you press the *Search* button, DocFetcher searches through the indexed data and lists the locations. Files containing the search term appear together with information such as the file size. Below you will find the text passages where the search term appears. DocFetcher highlights the term in yellow (refer to Figure 1).

Multiple Terms

In addition to the simple keyword search, DocFetcher also offers simultaneous searching for several keywords. You can also search for word sequences or specify terms to exclude from the search. If you want to search for two

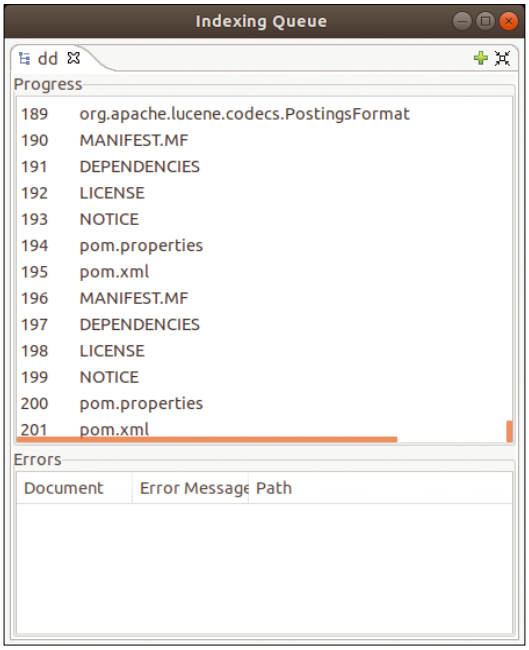


Figure 2: DocFetcher keeps you in the know during indexing.

terms, enter the two terms with the `AND` operand. DocFetcher searches for files in which both terms occur together, although they can occur at any location in the text. If you want the application to find an exact word order, you need to put the words in quotes.

You can exclude a term from the search by prefixing it with a minus sign. For a wildcard search, use a question mark or asterisk. The question mark replaces exactly one character in a search term; the asterisk replaces several characters. Especially when searching for compound nouns and technical terms, the asterisk is most helpful.

The search sometimes reveals results that are not needed at all. With the option to exclude unneeded formats, you can quickly thin out the list of hits. Uncheck the boxes to the left of the individual file formats in the *Document Types* window segment. Alternatively, use the *Search Scope* pane to limit the search to the relevant directory trees.

In the results display, you can scroll through the terms found page by page by clicking on the arrows to the left or right above the search display. The matches are shown with a yellow background. The up/down arrow buttons are used to navigate from match to match; DocFetcher highlights the search key in green.

Updates

As soon as you store new data in the directory hierarchies integrated by DocFetcher, you have to update the index to include all files in later searches. To update the index, right-click on the index in the search area and select the *Update…* option from the context menu. DocFetcher now integrates the new files and directories into the index in a process that is far faster than the initial indexing.

You can use the same context menu to list the documents in a folder without searching through them. Select the *List Documents* option. The software then displays the individual files in the results display top right in the program window. You can only apply this function to a single directory, not to higher-level directories that only contain subdirectories themselves.

To remove individual files from the folder, right-click the file and select *Open Parent Folder* from the context menu. The file manager opens, listing the files in the parent folder. Alternatively, you can display the folder contents by right-clicking on the directory in the lower-left corner of the search area and selecting *Open Folder* from the context menu.

Problems

DocFetcher supports the PDF format by default but shows its limits when displaying visually sophisticated documents with many illustrations and tables in the text display: Usually only the simple text appears without special formatting. Since DocFetcher displays PDF documents in the ASCII character set, the program does not correctly display special layout forms, such as texts with a multi-column layout. To eliminate this shortcoming, double-click on the match in the upper part of the window to start the PDF viewer installed on the operating system; the viewer then loads the document. In the document viewer, however, you then need to search for the desired term once again.

Another special feature is the handling of HTML content in DocFetcher. Because the software uses a lightweight, integrated web browser to display HTML content, when you click on an HTML file in the search results, its contents appears as a web page in the preview window.

You do need be careful when browsing web pages: Due to a lack of an ad blocker or a pop-up blocker in the integrated web browser, annoying advertisements could open; in our lab, a pop-up ad actually caused the entire program to crash. And vulnerabilities in the integrated browser cannot be ruled out. Do not use DocFetcher to retrieve unknown links.

Portable

DocFetcher lets users maintain portable repositories. These portable repositories can be stored on external data carriers such as DVDs or USB sticks, which you can carry with you. You can carry a complete repository of your documents, including the index files. To create a repository, you also need to save the program directory. Repositories archived in this way support full-text searching in the entire data collection.

To ensure portability when you create a new index, check the box to enable the *Store Relative Paths* option in the *Miscellaneous* settings area of the *Indexing Queue* settings dialog. This option ensures that the index is not rendered unusable with incorrect path specifications – the mount points may change from system to system.

Conclusions

I recommend DocFetcher as a search and indexing program for the desktop. The software is fast and stable, and it supports very simple and intuitive use; you will not need to spend your energy on time-consuming installations with database backends. One of DocFetcher's few shortcomings is the lack of support for some free formats, especially when it comes to multimedia. However, if you work primarily with ODF documents or PDF files, you should get along well with DocFetcher. ■■■

Info

[1] DocFetcher: *http://docfetcher.sourceforge.net/en/index.html*

[2] Apache Lucene: *https://lucene.apache.org/core*

FOSSPicks

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

Graham tears himself away from updating Arch Linux to search for the best new free software. BY GRAHAM MORRISON

Visual disassembler

Cutter

Reverse engineering, outside of those who actually do it, has a slightly bad name, because it's often associated with stealing algorithms from proprietary software or enabling game piracy. But it's actually a huge and legitimate area of research, as it enables old software to carry on running, old hardware to be emulated, or old bugs to be circumvented and performance investigated. It's also absolutely fascinating and a great way to learn about how your operating system actually works without the training wheels of an API or hardware abstraction. Unfortunately, the days when you could load a disassembler into your CPU's MMU are gone, and you can no longer realistically splice `nop` commands into an execution stack.

But there are tools that make the almost infinite complexity of modern operating systems and CPUs manageable, and one of these is the command-line tool radare2. Radare2 will take your Linux binary and turn it into a stream of assembler, letting you work out exactly what happens when a binary is executed. Of course, you need to understand assembler, your hardware, and the various subsystems a binary takes to produce output, but that's where radare2 will try and help, splitting the assembler into functions, libraries, and symbols to help you make better sense of what's happening.

The only problem with radare2 is that it's only slightly less complex than the infinite complexity of modern operating systems and CPUs. Fortunately, radare2 has just found itself with a shiny new GUI called Cutter, to help bring it's 1337 skills to the great unwashed mouse clickers. After launch, you simply select a binary and you're done. Something locally compiled will reveal the most useful information, but you can try anything. Cutter will then spend a little time analyzing the binary, the results of which are shown in a "sections" ring chart that brilliantly shows the components detected within the binary. A similar graphical representation of the binary is also shown as a horizontal bar chart at the top of the window, with each slice showing the use of sequential memory. But the main part of the Qt-configurable window is the disassembler itself, showing the assembly mnemonics, usually helpfully annotated to show loops and jumps, the names of registers, and function names. This is where you can follow the track of execution through your binary. Cutter helps with this, too, as you can click on jump or branch instructions to see where the code runs next.

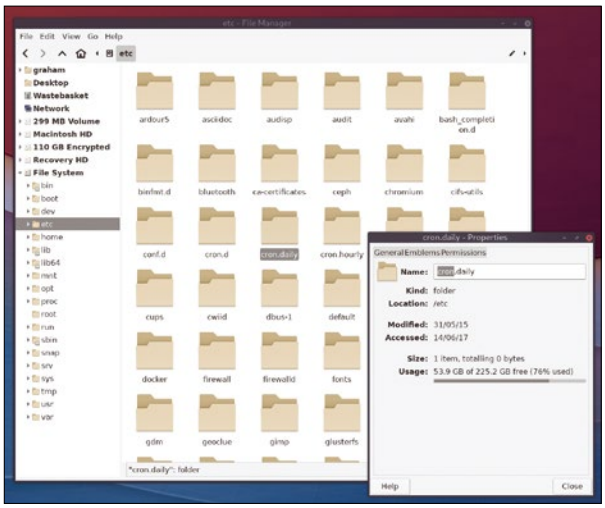
The disassembler is augmented with various other tabs to help you make sense of the code. There's a hex viewer, for instance, so you can see the raw binary contents of the file. There's also an excellent pseudo-code generator that will turn the assembler in a function into something where loops and structures are demarcated by curly brackets – it feels a lot like C, where the keywords have been replaced with assembler. Strings, entry points, and imports can be investigated in the same way, and you can search directly for either a flag name or an address. It's still complicated, of course, but even if you have no prior reverse engineering experience, Cutter still helps you get some usable knowledge out of almost any binary, even if it's just a greater understanding of how many subsystems there are in Linux and how many CPU cycles are potentially wasted drawing drop shadows.

1 Functions: Cutter can detect when a chunk of code is independent and therefore a function. **2 Binary map:** See where things are in the binary file you're viewing. **3 Search:** Look through the binary for both flags and a specific address. **4 Overview:** See the details for where you are and where you're going next. **5 Ring chart:** Analysis breaks up the executable into parts; this chart shows how each part is proportionately used. **6 Details info:** Create pseudo code, see a visual representation of the current function, and view the hex. **7 Sections:** The annotated version of the ring chart. **8 Disassembly:** Here's the code being run by your binary.

Project Website
https://github.com/radareorg/cutter

File manager

Thunar 1.8.0



Even on KDE, Thunar makes a great file manager utility, because it's quick and uses few resources.

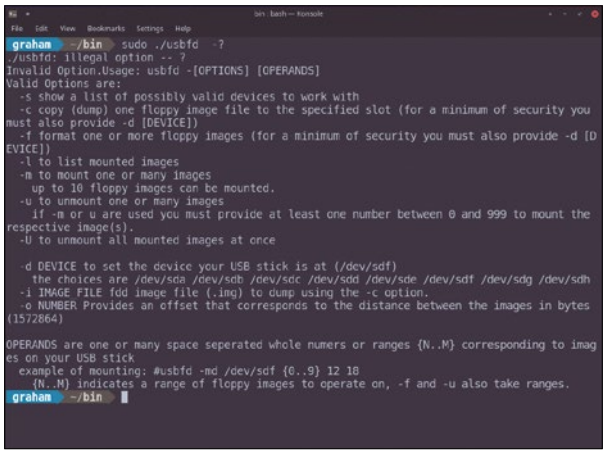
Xfce is often a little neglected as the world rages on about the current state of the Gnome and KDE desktops, but it has remained a quietly brilliant desktop environment for many years. The application at the heart of its effectiveness is the humble file manager, Thunar. What's great about Thunar is that it works and looks just like file managers of old, and it has what many people consider the default file manager functionality and appearance. The device list is on the left, network places beneath, and a main view on the right for viewing your files and folders. This view can quickly be changed from icons, to a details list, to a compact list, just like the file manager in a certain proprietary filesystem. But it works, and it's exactly what you need when you want some intuitive multifile selection that you can't easily do on the command line.

This update is a major release for a few reasons. The main reason is that Thunar is now using Gtk+3 rather than the ancient Gtk+2. This toolkit revision is happening across the entire suite of Xfce tools, with Thunar being perhaps the largest component. As with the GnuCash migration to Gtk+3 we wrote about previously, the long process of updating to the new toolkit "future proofs" Thunar and Xfce, as well as helps the UI look slicker whilst retaining its retro feel. But alongside that update, there's also a new and revised path bar at the top, showing breadcrumb navigation for where you are in the filesystem, alongside new *Next*, *Previous*, *Up*, and *Home* buttons. Small fixes add file sizes in bytes, user-configurable actions for remote locations, and refreshed icons for file transfers.

Project Website
https://git.xfce.org/xfce/thunar/

Virtual floppy writer

usbfd



As well as letting you copy files to virtual floppy disks, usbfd also lets you write entire images.

Floppy disk drives are now a distant memory, occasionally revived by the *Save* icon or the discovery of a stash of disks from storage. But there are old machines that still need to use them. For these, a great solution is to replace the drive itself with a device that emulates a floppy drive via a USB stick. These devices, such as those manufactured by companies like Gotek, are drop-in replacements for the old devices, often plugging into the same old ribbon cable and power supply. They enable old hardware with floppy drives, such as a Commodore Amiga, an old synthesizer, or a variety of scientific equipment, to be brought into the modern era of no moving parts. Even a 1GB thumb drive can store hundreds of floppy disks worth of data.

To store all these disks on a single USB requires a specific and peculiar partitioning regime. The tools for generating this schema are generally old and for Microsoft Windows. Fortunately, the floppy drive emulators can format a USB stick themselves by powering on when their two buttons are depressed. But this doesn't help you if you want to read and write to this USB storage from your Linux computer, which is the whole point of trying to move ancient hardware into the Internet era. Fortunately, this brilliant little tool will save the day. Consisting of no more than a single command, it sidesteps the horrible issue of these devices being formatted as both VFAT and FAT16 with 1,000 separate partitions and lets you simply mount or unmount individual "virtual floppys" from the command line. They're then mounted beneath the `/media` folder. From there, you can read and write files to the corresponding floppy folder – but remember, 1.4MB really doesn't go far!

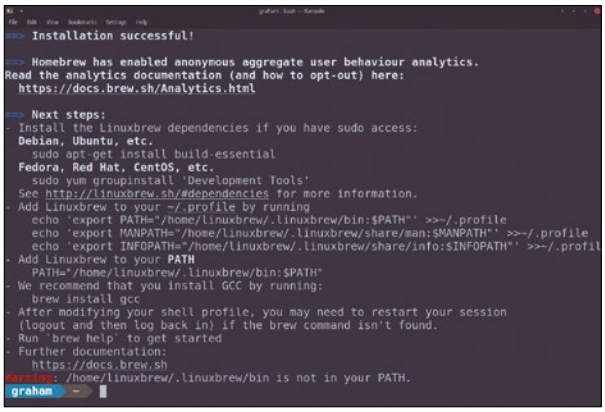
Project Website
https://github.com/dennisMe2/usbfd

Package manager

Linuxbrew

If you've used a Mac OS machine recently, you can't help but have noticed that many Mac OS installations seem to be filling up with open source software. The reason why this is happening now is because, while many parts of Mac OS are open source, it previously lacked the same binary packages Linux distributions build upon when providing new packages. That meant, if you wanted to install Gimp or Inkscape on Mac OS and there wasn't a native binary, you'd have to compile and install every dependency alongside the application itself. This problem was solved on Linux by package managers, and it's now been solved on Mac OS with a tool called Homebrew, or `brew`, as it's called on the command line. It works just like a package manager, pulling either available binaries that match your installation, or orchestrating a build environment so that your target can be installed. It may take longer, but it requires very few brain cells, and it works. This is why there's now so much open source on Mac systems.

Linuxbrew is Homebrew for Linux. But, why? There are four main reasons: The most compelling is that packages installed with Linuxbrew are installed into a home directory and not into any system-wide location. This means you don't need `sudo`, which helps if you're using a server, but it also means you don't need to trust the packages as much. The install script creates the new location or installs into your current home folder. With the binary added to your path, you'll be able to install packages that aren't yet available for your distribution, even on an old distribution. If you use a Mac, you can install the same software from the same sources on both. Package installation is as simple as typing `brew install` and the name of the package, just as you would with `apt` or `rpm`.



Like Wine, Brew uses bottles to keep package installations within their own filesystem containers beneath your home directory.

Project Website

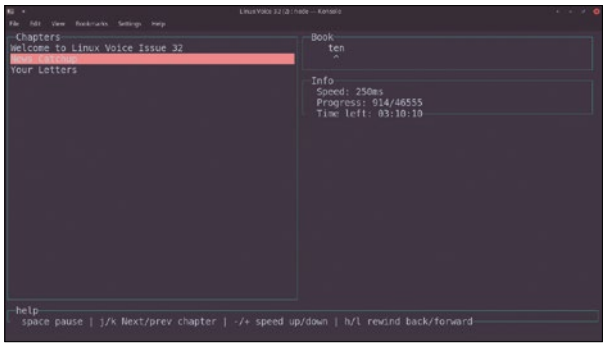
http://linuxbrew.sh/

Speed reader

Uniread

Despite the number of people reading books being reportedly fewer than before we all started looking at screens, we're all (probably) reading many more words than ever. And it seems neither screens nor books, at least in their default configuration, are ideal for reading. This is because you can dramatically boost your reading speed and even your comprehension by seriously reducing the amount of clutter that surrounds the sections of text you're trying to read. You can prove this to yourself with an online service called Spritz, which cuts down big text into single words flashed in front of your eyes with one letter colored red. This red letter is intended to be your focal point, and keeping this central allows you to read the words without your eyes ever having to jump around. The results can be remarkable, and even the Spritz demo, designed to hook you into the process, effortlessly feeds you words at 250 words per minute.

Uniread performs a similar trick, only on the command line, without any online privacy issues and with your own digital book copies. Installed easily via `npm`, it takes the location of your EPUB file as its single argument. With that done, you get a simple curses GUI that shows the chapters in your book on the left and the reading pane in the top right. Beneath this, a small Info window updates to show you your reading rate, progress, and estimated time left in the book. You then simply press the space bar to start reading. The words appear at a rate calculated from their complexity, although this can be changed with the cursor keys. Spritz's red focal point is replaced by an arrow underneath where you should focus your vision, but the effect is the same. It can be like reading with a super power, and there's little doubt you can speed read your way through the text. This can detract from the enjoyment of your favorite novels, but it's perfect if you want to hack through your obligation to this month's book club.



Speed read your way through a new novel from the comfort of the command line.

Project Website

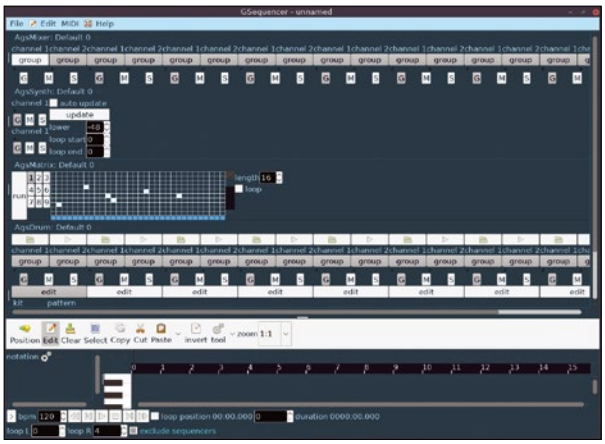
https://github.com/nemanjan00/uniread

Audio processor

Advanced GTK+ Streamer

GSequencer is a dive into the subsystem world of Linux audio, letting you tie all kinds of different elements together, such as step sequencers, plugin effects, note editors, and drum machines, and use these to generate audio or export a file. Rather than abstracting audio across layers, though, GSequencer plays it old school by targeting Advance Linux Sound Architecture (ALSA) directly. Consequently, you'll have to disable PulseAudio on any reasonably modern distribution. This is important because GSequencer aims to be a low-level routing and processing powerhouse, letting audio geeks manipulate audio streams via a visual interface without resorting to a fully fledged stack like PulseAudio, which also needs to cater for YouTube playback from Firefox. But you can still use GSequencer alongside PulseAudio if you suspend Pulse, as you might already do when running JACK, using the command `pasuspender -- gsequencer`. With that done, GSequencer can run unhindered, although it can take a while to run when first launched because it scans your system for any compatible audio plugins it can use, including LADSPA, DSSI, and LV2 plugin formats.

With the scanning out of the way, you can then access any of those plugins from the *Edit* | *Add* menu, which is where you add all the various elements you can use to process audio. There are so many different kinds of effects and sound generators, that explaining what GSequencer is capable of is almost impossible – it entirely depends on what plugins you have installed and the kind of sounds you want to process. Like a modular synthesizer, GSequencer enables you to line these up sequentially in any way you like, whether that's for reducing the noise in a podcast or for generating a sequence of randomized notes in a specific scale. The interface makes it all feel old school and primitive, but it's powerful and can make a refreshing change from all-powerful applications like Ardour or the commercial Bitwig Studio.



To boost its old-school credentials, GSequencer includes a pattern-based sequencer and drum machine.

Project Website

https://nongnu.org/gsequencer/

Process killer

fkill

There are as many ways to kill a process in Linux as there are to start them, and everyone has their favorite. Mine is typing `ps aux | grep <process name>` and then killing that process off directly with a cast of the `kill -9 <process id>` command, although I also resort to killing via `top` if the process is stealing CPU cycles and is difficult to track. Both of these methods have risks associated with them, especially `top`, which will let you type *k* on whatever process happens to be at the top of the CPU stack and kill it with a simple press of the Return key. `fkill` is a better alternative that also runs from the command line. What makes it better is that it's interactive. Running it with no arguments will list all the processes you currently have running, and you can cursor through those.

More like a menu, your selected process will attempt to stay in the middle of the selection as you scroll the list up and down. You can also start typing, and `fkill` will search for running processes containing your text. When you find the process you're looking for, such as PulseAudio, a simple press of Return is all it takes to kill the process (if you have the permissions), so you still need to be careful, but you can see clearly what you're doing and what's going to happen. One neat feature is that you can enter a process name or ID from the command line to kill that process without the menu. You can also kill a process bound to a specific port by prefixing the port with a colon. It's simple, but it works very well and is well worth replacing your current kill regime, because killing an editor containing your latest novel edits is never fun.

Turn your process list into a menu of tasks to be destroyed.

Project Website

https://github.com/sindresorhus/fkill-cli

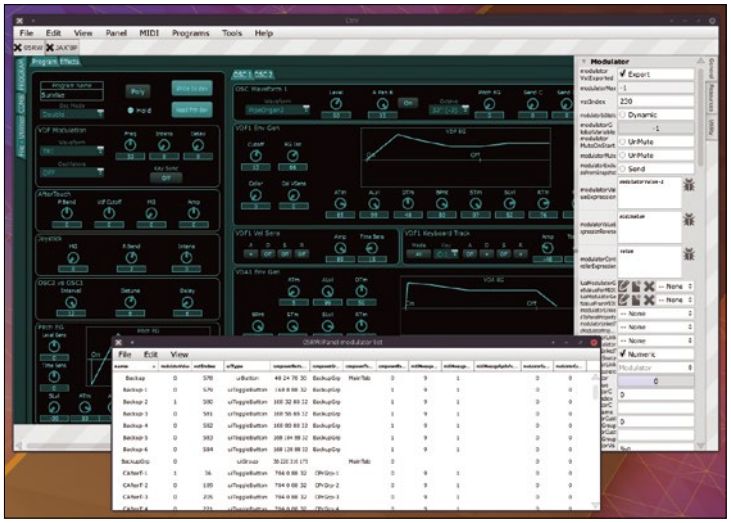
MIDI editor

Ctrlr

After spending the 1990s and the early 2000s becoming boring black boxes that either played back samples or prosaically emulated classic sounds from the 1970s, synthesizers have become exciting again. You can now buy genuine recreations of those old 1970s synths, built from new versions of old chips. Every week new synths are released that push the boundaries of audio design and performance, but in the decades since the Minimoog, one thing hasn't changed for synth players and noodlers – synths never have enough controls. Whether they've got five sliders or 50 knobs, there's always some aspect of the sound engine that you want to control but can't.

Third-party software editors have been around since synths started to talk MIDI in the early 1980s. These would often let you access these secret parameters, as well as store and manage your sounds, but you needed a different editor for every synth, and many would be left without updates, never making it further than 1987 and the Atari ST. The 21st century offers something different, however, and that's Ctrlr. More than a simple editor for a single device, it's an open source design platform for creating synth editors and sound managers that can be used on Mac OS, Windows, and Linux, either standalone or within your favorite audio environment. By removing the complexity of building an audio or MIDI application, removing the requirement to come up with your own graphical toolkit and a way of parsing the messages that need to be sent and received, as well as creating an editor to make all this accessible to everyone, Ctrlr has made MIDI editing cool again.

Ctrlr can be used in several ways. The standard way is to launch the application and then load a "panel." These panels are the software interfaces to your hardware synthesizers, and there are dozens for popular and unpopular synths alike. They look and feel just like you're controlling a software synthesizer, with sliders for parameters, X/Y pads to control pairs of values, knobs, and value output fields. You can snapshot the current setting to save a patch and save batches of patches to use later. As you change the parameters within the panel, Ctrlr sends the control or SysEx messages via MIDI to your synth, and it will equally receive parameters, too, if feedback is involved. If your synth has just a couple of knobs, this is a huge upgrade, but it also helps to integrate your external hardware with your internal software processes.



Breathe new life into your old MIDI synthesizer with an open source software panel to access its many secret parameters.

Since Ctrlr can send and receive any MIDI data, it doesn't have to be just for synths. You can also use it to create weird and wonderful performance interfaces.

Behind all this is an editor that lets you create your own panels, and this is way easier to achieve in Ctrlr than in other applications. You need only your synth's reference MIDI specification or a synth that transmits the values you want to automate, and you can design any kind of interface you wish. You can also edit any panel you have access to, tweaking its design or adding elements that are missing, so a panel won't hopefully become abandonware when the original designer moves on to something new. Because Ctrlr is so good at manipulating and managing MIDI data, you can use it to create performance GUIs that control many synths together, or none, in any way you choose, making it brilliant for avant-garde performance, too.

Project Website
http://ctrlr.org/

Dune 2 game engine

OpenDUNE

Despite its name, OpenDUNE is a recreation of the Dune 2 engine, rather than the earlier Dune (1).

If you're of a certain age, the memory of the game Dune 2 likely still burns with deep nostalgia. It was first released in 1992 and was popular on both the burgeoning PC platform and the then fading Amiga. What made it different from the average game/movie/book tie-in was that the game only borrowed the characters, the circumstances, and situation from the Dune books. Everything else, including the gameplay and your goals as a player, was entirely created by the game engine itself, and ultimately, Westwood Studios. This is important because the game engine was one of the first to implement real-time strategy, where you mine for resources, invest in infrastructure, and explore the terrain, eventually trying to outwit your neighbors in a race to dominate the terrain – in this case, the entire planet of Dune. Westwood would eventually transplant this entire engine into its hugely successful Command & Conquer franchise, a game series that started out almost exactly like a rethemed Dune 2.

There are numerous ways to play Dune 2 today. DOSBox is a great option for playing the original PC files, for instance, but there's also OpenDUNE – an open source implementation of that original game engine. You do still need the game data files, and you need to create a configuration file to help the binary find everything – and Linux sound isn't working – but the game does work. Seeing Dune 2 running on modern hardware is definitely worth it. It also ensures Dune 2 can continue to run long into the future, even on different architectures and platforms. That's why these creations are so important. It reportedly even works on an Atari ST running with a 68030 processor, which takes it all back to the beginning.

Project Website
https://github.com/OpenDUNE/OpenDUNE

Windows emulator

Snapped Wine Games

Huge thanks in particular to Martin Wimpress for snapping TrackMania Nations Forever into a Wine container to install from a single `snap` command.

The super convenient containerized snap packager has been gaining momentum recently, thanks to applications like Spotify, LibreOffice, Skype, and Bitwarden all becoming instantly installable from any of the distributions that support snap packages. Not all of these applications are open source, but the underlying snap technology is completely open. Anyone with a little spare time can roll their own and start publishing them, just as you can with Flatpak, DEBs, or even RPMs. But a recent weekend project that experimented with both Wine and snap packaging has repeated real benefits for gamers. That's because it's now feasible for some Windows games that work well with Wine to be packaged up within a snap, making them as easy to install as any other snap package: just `snap install <game>`, for instance.

This is a great development because Wine, the open source platform for running Windows software, often requires a very specific set of configuration options, often unique to each piece of software you want to run. Wine uses bottles to keep these configurations from touching one another, and it's why projects like PlayOnLinux are so popular. These take the guessing out of the configuration by bundling templates for trusted configurations to run a game or application. But now, in limited cases, you get all of this from the snap command line, and it starts with the rather excellent TrackMania Nations Forever. Type `snap install tmnationsforever` on a recent Ubuntu version, for example, and the Windows version of the game will be installed and configured to run in experimental versions of Wine without asking any further questions. It's still very much all in the testing phase, but it works and hints at a powerful future for prepackaged Windows binaries via snaps.

Project Website
https://snapcraft.io/

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*.

SREcon

Date: August 29-31, 2018

Location: Dusseldorf, Germany

Website: *https://www.usenix.org/conference/srecon18europe*

SREcon18 Europe/Middle East/Africa is a gathering of engineers who care deeply about site reliability, systems engineering, and working with complex distributed systems at scale. SREcon strives to challenge both those new to the profession as well as those who have been involved in it for decades. The conference has a culture of critical thought, deep technical insights, continuous improvement, and innovation.

Drupal Europe

Date: September 10-14, 2018

Location: Darmstadt, Germany

Website: *https://www.drupaleurope.org/*

Drupal Europe is both a technology conference and a family reunion for the Drupal community. Eleven industry tracks focus on real life case studies and success stories alongside emerging new best practices. Drupal Europe is put on by a group of community volunteers in collaboration with the German Drupal Association and the Drupal Europe Foundation.

Storage Developer Conference

Date: September 24-27

Location: Santa Clara, California

Website: *https://www.snia.org/events/storage-developer*

SDC18, brought to you by SNIA, is an essential event for storage software and hardware developers, storage product and solution architects, storage software engineers, product managers, storage product quality assurance engineers, product line CTOs, storage product customer support engineers, and in-house IT development staff.

Events

Akademy 2018	August 11–17, 2018	Vienna, Austria	https://akademy.kde.org/2018
Nextcloud Conference 2018	August 23-30, 2018	Berlin, Germany	https://nextcloud.com/conf/
Linux Security Summit	August 27-28, 2018	Vancouver, Canada	https://events.linuxfoundation.org/events/linux-security-summit-north-america-2018/
Open Source Summit North America	August 29-31, 2018	Vancouver, Canada	https://events.linuxfoundation.org/events/open-source-summit-north-america-2018/
SREcon18_Europe/Middle East/Africa	August 29-31, 2018	Dusseldorf, Germany	https://www.usenix.org/conference/srecon18europe
Atlassian Summit Europe	September 3–5, 2018	Barcelona, Spain	https://www.atlassian.com/company/events/summit-europe
Drupal Europe	September 10-14, 2018	Darmstadt, Germany	https://www.drupaleurope.org/
DevOpsDays Berlin	September 12–13, 2018	Berlin, Germany	https://www.devopsdays.org/events/2018-berlin/welcome/
The Linux Foundation Legal Summit	September 12–14, 2018	San Francisco, California	https://events.linuxfoundation.org/events/lf-member-legal-summit-2018/
Open Source Firmware Conf.	September 12–15, 2018	Erlangen, Germany	https://osfc.io/
Storage Developer Conf.	September 24–27, 2018	Santa Clara, California	https://www.snia.org/events/storage-developer
Open Networking Summit Europe	September 25–27, 2018	Amsterdam, Netherlands	https://events.linuxfoundation.org/events/open-networking-summit-europe-2018/
Open Source Backup Conference 2018	September 26–27, 2018	Cologne, Germany	https://upcoming.org/event/open-source-backup-conference-2018-0xlw0zogn5
All Systems Go	September 28-30, 2018	Berlin, Germany	https://all-systems-go.io/
All Things Open	October 21-23, 2018	Raleigh, North Carolina	https://allthingsopen.org/

CALL FOR PAPERS

We are always looking for good articles on Linux and the tools of the Linux environment. Although we will consider any topic, the following themes are of special interest:

- System administration
- Useful tips and tools
- Security, both news and techniques
- Product reviews, especially from real-world experience
- Community news and projects

If you have an idea, send a proposal with an outline, an estimate of the length, a description of your background, and contact information to *edit@linux-magazine.com*.

The technical level of the article should be consistent with what you normally read in *Linux Magazine*. Remember that *Linux Magazine* is read in many countries, and your article may be translated into one of our sister publications. Therefore, it is best to avoid using slang and idioms that might not be understood by all readers.

Be careful when referring to dates or events in the future. Many weeks could pass between your manuscript submission and the final copy reaching the reader's hands. When submitting proposals or manuscripts, please use a subject line in your email message that helps us identify your message as an article proposal. Screenshots and other supporting materials are always welcome.

Additional information is available at:
http://www.linux-magazine.com/contact/write_for_us.

NOW PRINTED ON recycled paper from 100% post-consumer waste; no chlorine bleach is used in the production process.

Authors

Contact Info

Editor in Chief
Joe Casad, jcasad@linux-magazine.com

Copy Editor
Amy Pettle

News Editor
Swapnil Bhartiya

Editor Emerita Nomadica
Rita L Sooby

Localization & Translation
Ian Travis

Layout
Dena Friesen, Lori White

Cover Design
Lori White

Cover Image
© lightwise, 123RF.com

Advertising
Brian Osborn, bosborn@linuxnewmedia.com
phone +49 89 3090 5128

Marketing Communications
Gwen Clark, gclark@linuxnewmedia.com
Linux New Media USA, LLC
2721 W 6th St, Ste D
Lawrence, KS 66049 USA

Publisher
Brian Osborn

Customer Service / Subscription
For USA and Canada:
Email: cs@linuxpromagazine.com
Phone: 1-866-247-2802
(Toll Free from the US and Canada)

For all other countries:
Email: subs@linux-magazine.com

www.linuxpromagazine.com – North America
www.linux-magazine.com – Worldwide

Printed in Nuremberg, Germany by hofmann infocom GmbH on recycled paper from 100% post-consumer waste; no chlorine bleach is used in the production process.

Distributed by Seymour Distribution Ltd, United Kingdom

LINUX PRO MAGAZINE (ISSN 1752-9050) is published monthly by Linux New Media USA, LLC, 2721 W 6th St, Ste D, Lawrence, KS, 66049, USA. Periodicals Postage paid at Lawrence, KS and additional mailing offices. Ride-Along Enclosed. POSTMASTER: Please send address changes to Linux Pro Magazine, 2721 W 6th St, Ste D, Lawrence, KS 66049, USA.

Published monthly in Europe as Linux Magazine (ISSN 1471-5678) by: Sparkhaus Media GmbH, Zieblandstr. 1, 80799 Munich, Germany.

Approximate On Sale Date

UK / Europe	Sep 01
USA / Canada	Sep 28
Australia	Oct 29

Issue 215 / Month 2018

Privacy

Next month we study some tools that help protect your privacy from the prying-eyes of cyber-snoopers and data miners.

Along with our usual mix of timely technical articles, issue 215 also comes with the latest release of the Complete *Linux Magazine* Archive DVD. This exclusive DVD includes the entire history of *Linux Magazine* – every article from all 214 previous issues. This limited release is available with the *Linux Magazine* print edition, or you can purchase the DVD directly through the shop. Secure your copy by subscribing now to the DVD edition at: http://shop.linuxnewmedia.com.

Preview Newsletter

The Linux Magazine Preview is a monthly email newsletter that gives you a sneak peek at the next issue, including links to articles posted online.

Sign up at: ***www.linux-magazine.com/newsletter***

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

openSUSE